Dublin City University

School of Computer Applications

A Grounded Theory of Requirements Documentation in the Practice of Software Development

Norah M Power

May 2002

Supervisor Prof Tony Moynihan

A thesis submitted for the degree of Ph D

Declaration:

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Norah Power
Candidate

ID Number: 96970162

Date: 25 July 2002

To the memory of my father

It is no linguistic accident that 'building,' 'construction,' and 'work,' designate both a process and its finished product Without the meaning of the verb, that of the noun remains blank

John Dewey, *Art as Experience,* 1934

Acknowledgements

This thesis would not have been written without the gentle guidance and wisdom of my supervisor, Tony Moynihan Thank you, Tony, for your seemingly endless patience and your faith in me

I was very fortunate, also, to have had the assistance of so many reflective practitioners who were willing to share their insights on requirements documentation with me I hope I have woven these into a recognisable reality Thank you all, for giving your time so readily, for sharing your experiences and 'theories of action' with me, and to many of you for your continued interest in my work

To my colleagues in the CSIS department at UL, I am deeply obliged for all kinds of help and support over the years Donncha O Maidin, for giving me the time to concentrate on this, Ita Richardson for helping me to get started, introducing me to people, and for always encouraging me, and Annette McElligott, for so much moral support, but also practical help when I most needed it Thanks also to Tony Cahill and Liam Bannon, for helpful discussions, and also to Brian Fitzgerald, who was a great help in preparing for the viva

Michael Breen helped in many ways, more than he will ever know Mike Keegan is another person whose help I could not have done without Thanks, also, to all the people who suggested and introduced me to the practitioners I had the privilege to interview

Thanks are due to John O'Donoghue, Eamonn Murphy, and the members of the Ph D Forum at the National Centre for Quality Management, who provided a supportive environment in which to present, explore, and discuss this type of research

There was one who appointed and dedicated himself to the role of "encourager-in-chief" of this endeavour Thanks George, for sticking with it, and thanks to all the other people who encouraged and sometimes nagged me over the years

Finally, I take this opportunity, also, to thank Ollie Ryan for twenty years of loving association, and for occasionally taking my mind off my work

Contents

List of Appendices

List of Figures

List of Tables

Abstract

This thesis is concerned with the concept of a "good enough" requirements document It takes the position, based on empirical observations, that standard prescriptive approaches have failed to identify the necessary and sufficient characteristics of a good requirements document, because what is good enough in one situation may not be desirable or acceptable in another Therefore, no single set of criteria can define "a good requirements document "

The thesis presents a grounded theory which attempts to explain the diversity of styles of requirements documents found in practice, in relation to the variety of situations in which software products and systems are developed It identifies the factors that might be useful to categorise situations from the point of view of requirements documentation

Requirements documents are widely used in software development, an activity typically carried out in an organisational context Organisational theory suggests that the best approach in any situation depends on the factors that affect that situation

In the research, it was found that experienced practitioners employ a wide variety of constituent elements, structures, and styles when documenting requirements This is in contrast with much of the literature on requirements engineering

The contribution of this research is in three parts (a) an analysis of requirements documents as texts, (b) a scheme for classifying system development situations with respect to the requirements documentation process, and (c) a framework matching typical requirements documents with the types of situations identified in (a)

As a grounded theory, it is the result of a detailed and systematic investigation into the role of requirements documents in the practice of software development Its status as a theory implies that it is tentative and provisional An outline of how the theory might be validated for its usefulness, applicability, and generality is presented in the concluding chapter

Chapter 1 The Focus of the Enquiry

1 1 What is a good requirements document?

This thesis is concerned with the notion of quality in a software requirements document, and especially with how this might vary in different situations Many authors have tried to define the concept of quality in a requirements document in the same way that software quality has been defined, by listing the essential attributes of a good requirements document Many of these lists derive from the *IEEE Recommended Practice for Software Requirements Specifications* which lists the desirable qualities of a requirements specification According to this, a Software Requirements Specification (SRS) should be

a) Correct,
b) Unambiguous,
c) Complete,
d) Consistent,
e) Ranked for importance and/or stability,
f) Verifiable,
g) Modifiable,
h) Traceable (IEEE, 1998)

These characteristics are often quoted as absolute desiderata, as if the more a specification has of each of these qualities, the better it will be However, some of these qualities can conflict with each other Sometimes you can have too much of something that would be desirable in smaller measure For example, if requirements statements are broken down too much, in order to make them traceable or modifiable, they may become inconsistent or difficult to verify Attempts to formalise a requirements document, to make it unambiguous, may lead to making it impossible for the people who are in a position to tell if it is factually correct to understand what is says Too much detail in a document has the same effect

Instead of asking what are the desirable characteristics of a requirements document, or whether a particular document conforms to an absolute standard of quality, a more appropriate set of questions might be

- What is a good enough requirements document?
- Is this a good enough requirements document for the purposes for which it is used?

Quality has been defined as fitness for purpose, so it is pertinent to consider the different purposes for which a requirements document may be used

1 2 Uses of requirements documents

It is widely agreed that requirements documents have a number of useful purposes They are a means of communication between the end users and the developers of the system They form the basis of a contract between the developers and their clients They should allow the developers to assess the feasibility of the system and they should form the foundation of the design and implementation of the system The IEEE standard for requirements documents lists the following purposes

1 *Establish the basis for agreement between the customers and the suppliers on what the software product is to do*
2 *Reduce the development effort by revealing omissions, misunderstandings, and inconsistencies early in the development*
3 *Provide a basis for estimating costs and schedules*
4 *Provide a baseline for validation and verification*
5 *Facilitate transfer of the system to new users or other parts of the organisation*
6 *Serve as a basis for enhancement* (IEEE, 1998)

The first of these, "the basis for agreement" is very important, since two different parties are involved, the ones who have the requirements, often called the customers, and the ones who are going to satisfy them, often called the suppliers, so there is a need for agreement and therefore communication between them Most of the other purposes are related to the systematic coordination and control of the system development project One of the most important aspects of this is the need for an orderly approach, and the requirements document is often seen as the baseline on which everything else proceeds

The term "requirements document" is used throughout this thesis to cover any written document, whether formal or informal, that forms the basis of the agreement between the parties in a software development project An alternative term, the "(software) requirements specification" is often used in the literature, but the word "specification" has particular connotations, and, as shall be reported in Chapter 4, several other terms are also used in practice, for this type of document, so the broad term "requirements document" is employed

1 3 Users of the requirements document

The users of a requirements document are often divided into the customers and the suppliers, a division which reflects the fact that the document is the basis of a contract However, there are other ways of categorising the different kinds of people who read and write requirements documents Gotel and Finkelstein (1994) identified three distinct roles in the writing of requirements

1 The principal, or person who motivates the document, and is responsible for the consequences

2 The author, who chooses the content and structure, and is responsible for the semantics

3 The documentor, who records/transcribes the content, and is responsible for the appearance of the document

These roles may be played by the same person or different people, depending on the circumstances (Gotel and Finkelstein, 1994)

A much wider set of people and roles are involved in reading and reviewing a document Depending on the situation, the audience or circulation list for a requirements document often includes marketing personnel, designers, programmers and other developers, testers, other analysts, project managers, product managers, as well as end-users, their representatives and their managers Each of these has a different perspective on the requirements document, as well as different ways of regarding its usefulness, and consequently, its desirable properties

The people involved in a project are often called stakeholders because they have an interest in the outcome of the project This is a role that is played in relation to a particular development project The same person may play two or more stakeholder roles at once, including sponsor, customer and user (Wieringa, 1996) The sponsor is the organisation or person that pays for the development organisation and its process The customer pays for the developed system In many cases, these are the same, but in others, they are different

1 4 A means to an end

A requirements document is a means to an end, but it is often regarded as an end in itself This is reflected in statements such as "the purpose of requirements engineering is to produce a complete and consistent statement of requirements," and is due to a tendency in software engineering to focus exclusively on the products of the system development process Christiane Floyd (1987) identified this dominant product-oriented paradigm of software engineering and contrasted it with the emerging process-centred perspective, in which system development is seen as a learning process for the developers and users alike, becoming part of an overall ongoing process of organisational change In this perspective, the system is never completed, being subject to an iterative evolutionary process interleaving use and development The requirements document is instrumental in this process, but is similarly never complete (Floyd, 1987)

A similar, though more technical, perspective on the evolution of software is evident in the work of M M Lehman, who made the distinction between E-type and S-type programs

(Lehman, 1984, 1989, Lehman and Belady, 1985) From empirical observation of several large systems, he identified E-type systems that were embedded in a real-world context and needed to evolve in order to remain useful in those environments He distinguished such systems from S-type programs which are simply required to satisfy a given specification He argued that the majority of software in use is E-type software and that the software life cycle is an inherently iterative process, with the entire development phase simply initiating this process of evolution A requirements document, in this type of situation, is not a product, but a heuristic tool in the initiation phase of the life-cycle process of an E-type system

1 5 A contingency approach

The term "contingency theory" was coined by Lawrence and Lorsch in 1967 in the report of a study they did on organisational structure, in which they concluded that the structure of a successful organisation is closely related to the environment in which it operates (Lawrence and Lorsch, 1967) This principle is often stated as *"There is no one best way to manage It depends on the situation "* (Scott, 1987)

Similarly, it could be said, there is no "one best way" to document requirements Many different kinds of situations exist in practice, and many different approaches to requirements documentation are used How is a practitioner, therefore, supposed to know what is the best way to document requirements in any particular situation, if all situations are different?

The answer may lie in identifying the situational factors that influence the use of requirements and the role of the requirements document in different kinds of situations The possible factors include the organisational context of use, the organisational context of development, the application domain, the system architecture, the contractual arrangements, and the relationship between the developer and the customer or client The question which is addressed in this thesis revolves around these and other situational factors, and how they might be used to explain observed differences in the ways that requirements for systems and software are documented in practice

1 6 The research question

So far, the question has been raised, what is a good enough requirements document, and an answer has been proposed, that it depends on the situation, because the document is a means to an end, a tool that must be useful in the situation One way in which system development situations differ is in the variety of relationships that can exist between the developers and the people or organisations who will buy and use the software These relationships are explored in the thesis Another probable source of variation is the problem situation, because software is used in many different application domains Other sources of variation will be introduced later, as they arise in the findings of the thesis

1 6 1 Customers, developers, and other stakeholders

The terms customer and client are often used interchangeably, both in the literature and in practice Customers are defined as the people or organisations who pay for and deploy the software, and clients are defined as the people or organisations who pay for the development (Robertson and Robertson, 1999) Users are people who will use the system, and may be important stakeholders in the development project, depending on the situation Other stakeholders may include product managers, marketing people, testers, and technical writers

Some situations are market-oriented, which means that the relationship between the developers and the customers is not as close as it would be in other situations, which are more solution-oriented Some situations involve a formal contract between the developers and their clients, which affects the nature and content of the requirements documents while other situations are based on a much less formal relationship between the developers and their clients

1 6 2 Different problem situations

Another area of variation which might influence variations in requirements documentation is the application domain Software applications can be applied to the solution of many different types of problems, from medical devices that diagnose and cure disease to lethal weapons, and systems that intercept and neutralise such devices Software is used for the communication and dissemination of information, or to protect privacy and unauthorised access to buildings or information Information systems themselves form a large proportion of all software-based systems, from transaction processing, through office automation, to decision-support, online analytical processing and data mining

Different attempts have been made to divide up application domains For example, real-time, event-driven systems can be distinguished from data intensive systems, but that is not a very satisfactory distinction Some real-time systems are embedded in specialised hardware, while others simply run on real-time operating systems

Glass and Vesey (1995) have compiled a collection of different taxonomies of applications These include different industrial taxonomies from software or solution vendors, as well as academic and research classification schemes They point out that different taxonomies have different uses, which lead to different organising principles None of these lends itself to matching development methods to problem domains, which is what the authors had in mind They conclude that a systematic approach to the classification of problem domains has yet to be done

The collection of industrial taxonomies surveyed in (Glass and Vessey, 1995) is extremely comprehensive and extensive, listing hundreds of applications However, it is doubtful that each application needs a dedicated approach to development Apart from the fact that it would not be practical, it would ignore the basic similarities underlying different applications

Experienced practitioners often report that similar types of problem tend to occur in different application areas For example, one developer found that his earlier experience of developing invoicing systems was useful later on when he was working on the problem of issuing parking fines This insight is congruous with Michael Jackson's recognition of the problem frame as a starting point in analysis (Jackson, 1995)

Different problem frames have been identified which suggest that certain problems share certain basic similarities which can help with the task of solving them, by outlining what needs to be known in order to adequately describe the problem Problem frames also suggest that the adequacy of a given description to describe a problem is related to the problem at hand, rather than being independent of it

1 6 3 Specifying requirements

The first part of the research question has been introduced, namely what are the different situational factors or variables that influence the variation in requirements documentation Stakeholder relationships and problem situations are two probable areas that might be involved How requirements are specified in the document is another area of variation which will serve as the dependent variable in this research question This is an area of requirements theory and practice where very little agreement exists

This is reflected in the wide range of requirements specification techniques and notations that are used for specifying requirements, but none is as widely used as plain English, or natural language Many notations and modelling techniques have been proposed, but in many cases, requirements documents are written either mainly or entirely in natural language The obvious drawbacks and criticisms of natural language as a means of specifying requirements have often been reiterated A typical example by Jackson (1983) lists redundancy, lack of precision, difficulty of evaluating completeness, inability to evaluate internal consistency, inability to reveal the implications of requirements among the disadvantages of "narrative requirements " Nevertheless, most requirements documents rely on natural language for understanding, for flexibility, and for supplementing and tying together whatever is specified in the more formal notations

Another reason may be that software documents are essentially textual documents, because software is essentially a textual medium

"The most distinctive feature of software development is that it consists entirely of textual manipulation From the first verbalization of an application concept, whenever a system or part of it is developed, fixed, enhanced, adapted or extended, text is added, changed and/or eliminated to achieve the desired result " (Lehman, 1989)

1 7 Research, theory, and practice

This thesis is concerned with trying to explain (or structure an explanation of) the diverse ways that software requirements are documented in practice To date, this diversity has not been examined or analysed by any empirical study or theoretical framework This is reflected in the literature where most approaches to requirements specification seem to be presented as application-independent, or context-independent, as if they were universally applicable This is not the case in practice

There seems to be a large gap between the research and the practice of software engineering in general Many authors, for example, (Fitzgerald, 1996), have identified a gap between theory and practice in the fields of information systems and software engineering This has led to a debate in the information systems literature now known as the debate on Rigour versus Relevance But the real gap is between research and practice, not theory and practice Researchers do not have a monopoly on theory, because it can be argued that most people, including practitioners, engage in theory building as part of their work

The act of theorising is often considered to be a part of everyday thinking, and this has sometimes been used in explaining the genesis of more formal theory Gilbert Ryle in *The Concept of Mind*, presents theory building as an activity which all people of any educational level engage in at some time (Ryle, 1949) Peter Naur in writing about programming as theory building espoused this sense of the word "theory" (Naur, 1992) Towards the end of his life, Albert Einstein put it the other way around *'The whole of science is nothing more than the refinement of everyday thinking "*

George Kelly's personal construct elicitation techniques (Kelly, 1955) are based on the idea that everybody is engaged in constructing their own theories to make sense of the world Chris Argyris and Donald Schon (1978) coined the term "theories of action" (to cover espoused theories and theories in use) Schon later observed that reflective practitioners have their own theories of action that guide their professional behaviour (Schon, 1983) Organisations also engage in theory building or sensemaking, as it is called (Weick, 1979, 1984a)

This thesis sets out to examine the conjecture that the variety of approaches to requirements documentation used in practice is related to the situation of use, and therefore can be understood by looking at the different situations and asking what approaches have been found to work in practice in those situations The answers are expressed in a theory which is expressed as a conceptual framework which seeks to explain the variety of documentation practices in relation to the variety of situations The research, therefore, begins with practice, and ends with theory

1 8 Structure of the thesis

The thesis is divided into four parts This chapter has introduced the research question that will be addressed in the remainder of the thesis Part One contains two further chapters Chapter 2 deals with the method of enquiry, which is inductive rather than deductive, and positions it within the range of alternative research methods Chapter 3 outlines how the research was carried out, and explains how the findings were reached

Part Two contains the main findings of the thesis Chapters 4, 5, 6, and 7 present the proposed theoretical framework, which analyses and seeks to explain the variety of different types of requirements documents used in practice, in terms of their situation of use

Chapter 4 sets forth a taxonomic analysis of the different textual elements or components of requirements documents This explores the variety of different kinds of requirements statements that are used in documenting requirements, and reflects the wide variety of uses to which they are applied Seven different categories of requirements statements are distinguished, and these are grounded in the interview data

Chapter 5 presents an ontological analysis of requirements concerns, the substantive issues that influence and give rise to requirements themselves These concerns are grounded in the interview data, and the analysis is presented as a graph showing the precedence relationships among the different concerns that may exist in different situations It identifies eight root concerns and their dependent concerns The analysis is supported with quotations from the interview data

Chapter 6 proposes a classification of requirements situations which is based on the root concerns identified in Chapter 5 Seven typical situation profiles are presented, illustrated and supported by suitable quotations from the interview data

Chapter 7 is a situation related classification scheme for requirements documentation This consists of profiles of prototypical requirements documents based on the documentation model of Chapter 4 and the situation types presented in Chapter 6

Part Three has two chapters that give theoretical support for the framework presented in Part Two Chapter 8 is a review of established organisational theories and perspectives on the way people working together in organisations use documents as resources for action, as reflected in the situations identified in Part Two of the thesis Chapter 9 looks at established concepts and current research on representation and modelling in the field of requirements engineering, and how these are reflected in the different types of requirements statements found in the empirical study

Part Four is an attempt to assess the theoretical framework as a contribution to the field Chapter 10 compares the framework to previous and current literature on requirements documentation, and Chapter 11 summarises the work, looks at what needs to be done with it, and presents some conclusions on the contribution and relevance of it

Chapter 2 The Method of Investigation, and its Justification

2 1 Empirical research in software development

Software development or software engineering is by nature an empirical activity Software practitioners build products and test them to see if they work as expected Software engineering researchers have therefore typically followed a similar approach, sometimes called applied research, in which the goal of the research project is to produce a new product or prototype or proof of concept, usually with a suitable acronym An accepted part of any conference on software engineering is the session of research demonstrations, in which researchers exhibit the latest batch of newly developed tools and environments and claims are made for their advantages over earlier tools This phenomenon has been called "build and boast," mainly because of the lack of formal evaluation that is applied to such tools *vis a vis* their competitors, or in the context in which they are intended to be used But it is not the only way to do research in software development and, in recent years, many researchers have begun to adopt a more formal approach to the evaluation and assessment of software engineering tools and techniques This more formal approach is what is now called empirical research in software engineering (Basili, 1996, Perry et al , 2000) and it is basically of two types laboratory experiments and field research

2 2 Laboratory experiments

Empirical research is often mistakenly equated with experimental laboratory research There is a long tradition of experimental research in the cognitive aspects of programming (Gray, 1996, Olson, 1987) These studies have concentrated on how novice programmers learn to program, or studies of program comprehension, often based on cognitive models of programming Typically these studies are carried out in a closed environment under controlled conditions with carefully selected subjects and control groups, and are based on simple tasks which can be completed in a given space of time, so that the outcomes and differences between subjects can be measured accurately

A rare, early, example of applying laboratory experiments in the wider area of systems development was Vitalari and Dickson's (1983) study of the problem-solving behaviour of systems analysts While the results of that study have often been quoted, there has not been a strong tradition of doing laboratory research in systems analysis This is possibly because of the difficulty of finding suitable subjects, but the main reason is probably the context-dependent nature of systems analysis and design

Another advantage of experiments is that they can be replicated, although this is not commonly done However, some notable experiments have been carried out and replicated in the area of testing and inspection, including the inspection of requirements documents (Basili, 1996, Perry et al , 2000, Seaman and Basili, 1997, 1998)

2 3 Field experiments

The pitfalls of experimental studies are well-known, e g the problem of biased samples, unrepresentative subjects, etc but the main weakness of laboratory experiments is that they take place in artificial settings, removed from the real world context and conditions of software development Therefore, some researchers have adopted the practice of what is called "Industry as Laboratory" whereby real-world experiments would take place in natural settings But it is hard to find suitable sites for such studies, better known as field experiments and in principle they would be impossible to replicate, for it would be impossible to control all the various factors that might interfere with the outcome

Another well-known obstacle to the success of field experiments is the Hawthorne effect, so-called after a series of productivity improvement experiments in the Hawthorne telephone relay plant in the early twentieth century The subjects seemed to improve their productivity no matter what experimental treatment was applied, thereby invalidating any conclusion that a specific treatment would improve productivity This led to the recognition for less obtrusive methods of observation, with the consequent ethical issues intrinsic to such methods

However, despite the difficulties, some attempts have been made at establishing the pursuit of field experiments in software engineering Some of these are aimed at formally evaluating the outcomes of applied system development practices (Fenton et al , 1994, Potts, 1993) The field experiments that are increasingly carried out in software engineering belong to the general methodology of field research, which already has a strong tradition in the discipline of Information Systems and includes a broad variety of different types of studies action research, case studies, longitudinal studies, etc as well as the more conventional survey research

2 4 Survey research

A common approach to field research in information system development is to design a specific enquiry such as a questionnaire study, asking a sample of people to respond to structured set of questions, either orally or in writing This approach suffers from two serious drawbacks, the problem of choosing an appropriate sample, and the problem of asking the right questions

Because it is a practical impossibility to question everybody who might be able to respond, it is normal to choose from the target population a representative sample of respondents, and hope that they will provide an adequate response rate This is similar in purpose to sampling for a laboratory experiment, but much harder to control The inevitable problems of postal and email survey research are well known and include low response rates, self-selection by the respondents, and the need for follow-up contact

Another perennial problem of this kind of research is the problem of questionnaire design You cannot be sure that your respondents will understand the questions and interpret them in a uniform way For example, it was reported that in a regional survey of ICT (Information and Communications technology) companies, several respondents working in ICT companies did not understand or recognise the term ICT, or realise that it applied to their company, when in fact the researcher considered that it did, and had chosen them as such In my own investigation, at an early stage, I found my interview respondents used the term "formal methods" (a term I avoided in questioning them) quite loosely compared to my own more strict academic interpretation of the term This confirmed my decision to adopt a more exploratory approach to the investigation

A properly conducted field survey can be very useful in testing hypotheses which have been generated by a good theory of the phenomenon being studied, based on variables and concepts which are well understood, and generally applicable It is not suitable for investigations which seek to identify such variables, or create such theories, or studies where interpretation and context are important factors

2 5 Qualitative research

This research into requirements documents is concerned not just with people's perceptions and intentions, but also with the work they do over extended periods of time, and the people they work with, all of which are situated in specific contexts that cannot be replicated in laboratory experiments Nor can it be effectively addressed by traditional survey-based methods, such as questionnaires based on pre-determined questions It requires a research instrument that is sensitive to the perceptions and intentions of the actors involved, and qualitative field research methods seem to be the only ones that can fulfil this need (Lee et al, 1997)

Qualitative research is best explained by distinguishing it from quantitative research Qualitative research is exploratory research As Seaman and Basili, in their study of the way that organisational factors influence software inspection meetings, put it

"*The aim of this study is not to test or validate hypotheses about relationships between these variables, but to explore what relationships might exist and try to explain those relationships* " (Seaman and Basili, 1998)

Very often, the idea is to identify the variables themselves, as well as the relationships between them Also, many types of qualitative research offer the possibility of systematically generating new theories (theory building) rather than simply the means to test existing ones There is an undoubted need for both types of enquiry because they are complementary to each other

The best example that I have come across of this type of research applied to software requirements is called "*A Field Study of the Software Design Process for Large Systems*" (Curtis et al , 1988) Despite the title, this study deals with the requirements process in seventeen large projects Project personnel were interviewed in order to find out their perspectives on the process, focusing on how requirements and architectural design decisions were "made, represented, communicated and changed" as well as how these decision processes affected productivity and quality

The authors of this field study call their approach "ecological," not qualitative But, they contrast it with quantitative and experimental methods, which they judge to be insufficient for providing the insights needed in what they call "problem-driven" research Comparing their approach to the field research methods of sociology and anthropology, they say

"*The need for expedient results dictated the short, intensive study of a broad cross-section of projects, rather than the longitudinal study of a single project* " (Curtis et al , 1988)

2 6 Varieties of qualitative research

Qualitative research, which is sometimes called intensive research, covers a range of different types of strategies, varying in their level of intensity or involvement with the subjects they study Case study research is one of the most intensive, being the study of a single instance or a small number of instances of a phenomenon, in a natural setting, with a view to understanding, description and explanation, hypothesis generation, but not testing Sometimes the results of a case study may be used to challenge or question existing theories, but that is not their purpose Case study researchers usually employ a variety of means of data collection, because the number of sites is limited, the cases must be looked at from as many points of view as possible, so interviews, video recordings, observation, document analysis, or any other technique may be used The results are in the form of a rich description of the phenomenon being studied Comparisons and contrasts may be made between cases, but statistical reports are not used

Action research is similarly intensive, because the researcher is actively involved in a given situation, introducing some new technique or practice, with a view to improving the work carried out in that situation and evaluating its effects in that situation Its advantages and disadvantages are similar to those of case study research, with the added problem of the possibility of bias on the part of the researcher Therefore, it is difficult to generalise the results of action research, a more serious objection in these studies than in case studies generally

A longitudinal study is a case study that takes place over an extended period of time, with a view to understanding the changes and the processes that take place over time in a given situation The researcher is engaged with the situation even more intensively than in an ordinary case study

The typical case study focuses on an organisation, a project, a site, or a particular group of people But qualitative research is not always quite so intensive More extensive qualitative enquiries are carried out which aim to gather data more widely, in order to systematically study many different cases of a phenomenon This approach may be called multi-case qualitative research, or a qualitative survey In this vein, investigations into system development methods, tools, techniques, practices, and organisational structures have been undertaken in which qualitative field data is systematically collected from multiple sites Such studies are carried out with a view to covering as many different cases as possible, but without necessarily applying quantitative analysis to those cases Examples already referred to in this chapter include Curtis et al , 1988, Seaman and Basili, 1997 and Seaman and Basili, 1998 Multi-case qualitative approaches contrast with in-depth case studies in targeting a variety of cases rather than deep engagement with a single case They contrast with traditional quantitative surveys in terms of both the data collection methods used and the interpretation of the data

Qualitative studies are sometimes called interpretative studies, in recognition of the fact that they deal with data that must be interpreted The writers on qualitative studies often contrast this "interpretivist" stance with the logical positivist stance which is more appropriate to survey-based quantitative research, but different methods and types of qualitative research vary in this regard Researchers using qualitative methods may adopt different philosophical positions What unites qualitative researchers is that they are dealing with data that is textual rather than numerical, rich and varied rather than regular and stratified, context sensitive rather than context-free A qualitative approach is open to the varieties of meaning and interpretation that may be present in a situation, while adopting either a positivist view of the situation or a more interpretative epistemological position

2 7 Assessing the validity of qualitative research

Perhaps the most critical distinction between qualitative and other types of research is how the results are evaluated/validated Quantitative research is always validated according to the criteria of scientific method

Internal validity is concerned with the conduct of the experiment or survey, the reduction of the possibility of bias or error, etc (Are there any other explanations of the behaviour than the hypothesis being tested?)

External validity refers to the potential to generalise the results of the enquiry to the whole population (Are these results applicable outside the sample studied?)

These are important questions because quantitative research is always concerned with hypothesis testing But qualitative research is not (or not necessarily) Lincoln and Guba (1985), in their influential book, *Naturalistic Enquiry*, have laid down an alternative set of criteria for assessing the trustworthiness or validity of qualitative research These criteria are

1 Credibility (which corresponds to internal validity, but has different techniques for establishing it They suggest five techniques)
2 Dependability (which corresponds to reliability)
3 Confirmability (corresponding to objectivity) and
4 Transferability (corresponding to external validity, but very different from it) The transferability of any piece of qualitative research must be established, not by the researcher who produced it, but by anyone who seeks to apply it to some other context (Lincoln and Guba, 1985)

The case for the qualitative approach to research may be summarised as follows

It is exploratory, concerned with theory building, the generation rather than the testing of hypotheses It is context sensitive, more intensively engaged with the subject than any kind of survey research It gives more attention to interpretation than quantitative research does, deals with more varied data than quantitative research, but it is subject to different criteria for acceptability than quantitative research

2 8 Research methods[1] in qualitative research

Laboratory experiments and survey-based field research are often classified as quantitative approaches because the methods of data collection and analysis that they use are typically quantitative Qualitative approaches, on the other hand, are associated with a variety of alternative methods of data collection and analysis Both types of approaches use interviews as a means of data collection, for example, but the quantitative approach to interviewing is more structured, with a greater number of closed questions than open-ended questions, and uses questions requiring fairly short answers that can easily be coded, while the qualitative approaches use either semi-structured or unstructured interviews, allowing and encouraging the respondent to answer at length

The specific data collection methods used in qualitative approaches include observation, participant observation, interviews, game-playing, focus group discussions, card-sorting, etc as well as document searching and video-recording and the collection of verbal protocols (sometimes called "thinking aloud ") Weick (1984b) gives a different list of methods, but as he says, "Methods are simply ways to systematise observation "

However, some methods are more elaborate, and go further than that, giving specific guidelines on how to systematically gather data, analyse it, and use it to generate theory One is example is George Kelly's method of triads, better known as Personal Construct Elicitation (Kelly, 1955) Another method, which provides a systematic way to gather data, analyse it and generate theory from it, is called Grounded Theory (Glaser and Strauss, 1967, Strauss and Corbin, 1990)

2 9 Grounded theory

The particular research method I used was that of grounded theory This systematic method has been widely used in studies of medical and nursing practice, teaching and education, organisational and management studies, as well as some recent research into information systems and software development which I review later in this chapter

Grounded theory provides a systematic method of discovering categories and relationships in empirically collected data and building theories based on them, which are, in turn, grounded

[1] I am following the usage employed by Galliers (1991) and others in distinguishing research methods from the more generic concept of research approaches *"Different approaches are a way of going about one's research "* Methods are more specific ways of collecting data and analysing it

in the data The method encompasses three distinct activities data collection, coding or categorising the data, and the writing of theoretical memos Used in parallel, and iteratively, these techniques lead the researcher to discover a refined theory to describe, explain or elucidate a given domain of study The process is inductive, not deductive (Gummesson, 2000) Deductive research starts with an existing theory and draws some conclusions or hypotheses from it, in order to test them, thereby testing the theory Inductive research, on the other hand, begins with empirical data and looks for patterns, explanations and meaning

The aim is to construct or refine a theory rather than to prove or disprove it The essential status of the resulting theory is a plausible description, or explanation, etc However, internal verification of the proposed hypotheses is facilitated by providing a measure of groundedness or saturation Theoretical saturation is a complex concept related to the range of variety found in the supporting data and the extent of that support

The generation or discovery of grounded theory is done through a process of three parallel activities

1 Data collection
2 Coding (or the naming and analysis of theoretical categories and their relationships)
3 Memoing (or the writing of short pieces of theoretical analysis)

This process proceeds in a cyclical manner, as the emerging theory guides the search for further data and convergence of categories

Groundedness should not imply that a theory is solely based on empirical data and not informed by any previous theoretical knowledge Existing theoretical knowledge can contribute much to an emerging grounded theory or can itself be refined or extended by using the grounded theory method The original formulation of grounded theory (Glaser and Strauss, 1967) argued, *"Let the data speak for themselves "* This is best regarded as a reaction against an earlier prevailing trend in sociology that favoured overly quantitative and inappropriately positivist research Later publications such as (Strauss and Corbin, 1990) allow for the role of existing theory in contributing to the emerging grounded theory

2 10 Levels of analysis

Grounded theory has three distinct types of coding which vary in use with the progress of the emerging theory

1 **Open coding** is relatively free of constraints, and subject only to whatever patterns are emerging in the data This is essentially the identification and naming of categories The categories or codes themselves may be present in the data, as terms

used by the participants, these are called in-vivo codes Alternatively, they may be more theoretical codes, created or named by the researcher These names may be suggested by patterns in the data, by terms used in practice, or by theoretical constructs from previous theories

2 **Axial coding** is the linking and integration of categories, giving rise to clusters of theoretical ideas It is more selective than open coding, and usually involves formulating relationships between selected categories, making them denser, and richer in explanatory power

3 **Selective coding** is concerned with selected "core categories" and their relationships At this stage of coding, the emerging theory dominates the coding process, and all coding becomes subservient to the core category, which is not only central to the emerging theory, but also stable, saturated and highly associated with the other codes

Another important feature of grounded theory is that it encourages the search for differences as well as similarities in situations The constant comparative technique supports the identification and naming of categories, while suggesting the existence of counter-examples and variations, extending the search for variety, through a process called "theoretical sampling " Guidelines on qualitative research suggest that analysis should be done from the outset of such studies to help sharpen the focus of enquiry and guide the research process This is particularly true of the grounded theory method Data collection, coding, analysis and theory generation should proceed iteratively, until a point of saturation is reached

Saturation is said to occur when a particular concept or code has so much supporting data associated with it that no significant changes to it can reasonably be expected, and additional data no longer contribute to discovering anything new about it This requires judgement on the part of the researcher, who is creating the theory from the analysis of the data

At the heart of grounded theory lies a specific technique called theoretical sampling which guides the search for variation in the data Theoretical sampling is sampling directed, not by population size or any other *a priori* criterion, but by the emerging theory in the hands of the researcher, seeking events, cases or situations to flesh out the bones of the theory, either confirming or refining the category under consideration, casting the net wider, in order to have a richer, more saturated explanation

2 11 Levels of theory

The grounded theory method of theory building allows the researcher to choose the level of theory that is appropriate to the question under investigation Any number of levels is therefore possible, but Maykut and Morehouse (1994) selected three distinct levels of reporting, that illustrate the range of possibility Their levels are described as follows

(1) the journalist, reporting the facts and "letting the participants speak for themselves "

(2) the weaver, "constructing a recognisable reality" by selecting and interpreting data

(3) the "theory builder," arriving at more abstract organising concepts based on the data

(Maykut and Morehouse, 1994)

In the area of software engineering research, grounded theory has been used to build theory at different levels It has been used to build descriptive theories of a substantive domain, such as software architecture, (Grinter, 1999) as well as more sophisticated explanatory and causal theories of the factors surrounding the inspection of software, and the adoption of CASE tools (Cronholm and Goldkuhl, 1994, Orlickowski, 1993, Seaman and Basili, 1997, 1998)

The notion of causality is central to scientific research, as demonstrated in disciplines such as physics and chemistry But most things have multiple causes, and beyond the level of physical phenomena, cannot be explained simply in terms of cause and effect The meta-model paradigm of grounded theory provides a set of concepts for explaining situations, including the notion of causal conditions, intervening conditions, and intervention strategies, and interactions between them In other words, it recognises the multiplicity of causes, and the interaction of causes and effects, and provides a theoretical framework for explanations of such complexity, in the form of the conditional matrix, to be discussed later in this thesis

Miles and Huberman (1994) explore the notion of causality and its relationship to explanation They do not recognise a clear boundary between description and explanation, or between explanation and causality, but they encourage the search for explanation within cases, and that includes causality Nevertheless, they recognise that causality may not be an appropriate concept to apply to human behaviour As Gilovich illustrates in his book on the fallibility of human reason in everyday life, most people readily assign simple, seemingly rational, causes to what are in fact random events (Gilovich, 1991)

Whether causality can be applied to organisational behaviour which is multifaceted is another matter, but is not discussed by Miles and Huberman Weick discusses how organisations engage in sense-making to find retrospective explanations for previous events

(Weick, 1984a), but that is a separate issue from finding a theoretical explanation for organisational situations

Miles and Huberman (1994) review a number of different views of causality, going back to the philosopher, Hume In encouraging the search for causal explanations with cases, they advocate looking for the reasons why, for the process of causality, the mechanisms of causality, as well as the independent and dependent variables In a study based on a single case or a small number of cases, this would be documented in the form of a rich description of each case, outlining the complexity of the processes involved In a more extensive investigation, requiring what they call multi-case analysis, they recommend that explanations should be documented in matrix form This gives a picture of the possible variety of causes over a range of cases, but it does not give a more general explanation of situations

2 12 The grounded theory meta-model

As well as being a method of enquiry, grounded theory also provides a meta-model of the theories that it might be used to generate (Strauss and Corbin, 1990) Some researchers following the method seem to ignore this meta-model, or maybe modify it to suit their purposes, but some use it effectively as a template to format their theories in terms of the significant types of relationships that the grounded theory meta-model suggests

The meta-model might be considered a framework of built-in types of codes These include conditions, consequences, actions, context, etc The consideration and search for these patterns of codes leads to the discovery of specific examples of these and of more particular substantive codes, specific to the data The built-in relationships of grounded theory include is a condition of, is a consequence of, etc Again, these lead to relationships that are more specific to the subject or topic under investigation

Orlickowski's case study is a good example of applying the meta-model to a practical situation (Orlickowski, 1993) Her theory is presented in the form of a set of categories and related concepts showing how the institutional context influences the interplay of conditions, actions, and consequences of adopting CASE tools in an organisation This reflects the classical grounded theory meta-model of conditions, actions and consequences affecting a phenomenon

Seaman and Basili (1998) also report using grounded theory in their research into software inspections, but their results are presented in the form of a network showing the relationships between independent and dependent variables This is a much stronger statement than the kind of theory built into the grounded theory meta-model, which avoids the suggestions of causality inherent in dependent and independent variables, preferring to explain situations and phenomena in terms of the more complex interactions of conditions, actions, and consequences

2 14 Summary

This chapter has reviewed a number of different approaches to empirical research that have been applied to research problems and topics in the area of information systems and system development There is a clear need in these areas for research that is relevant and context-sensitive, because system development is a human activity that takes place in complex organisational situations Qualitative approaches complement quantitative approaches in allowing for exploration and interpretation to take place

In choosing the right approach, it is important to consider the purpose of the research, whether it is to validate existing theories or to discover new concepts and relationships Studies that focus on hypothesis testing use an approach that is different from those studies that are aimed at the discovery of theory, at exploration, explanation, or description

Systematic, repeatable research requires the use of an appropriate method, which must be documented before and after the fact Grounded theory provides a systematic, well-documented method for carrying out exploratory research which is aimed at theory-building, for discovering the salient variables in a particular type of situation, and for exploring the complexities of different types of situations

A qualitative approach was chosen for the research described in this thesis, because the research question was couched in terms of exploration and explanation, rather than testing and validating existing theories Grounded theory was chosen as the research method for similar reasons, but also because it provided a systematic yet iterative way of dealing with multiple cases, allowing the variety and some of the complexity of organisational situations to be incorporated in the results rather than being simplified out of the picture It also provided a means of gradually moving from description through analysis to theory building The next chapter describes how this particular method was employed to investigate the practice of requirements documentation and to analyse the results of that investigation

Chapter 3 Investigation and Analysis

"It is better to take advice from an experienced (person) than from an expert"

(Arabic saying)

3 1 Introduction

I set out with the objective of finding out as much as possible about the ways software requirements are documented in practice and, also, how experienced practitioners view the role of written requirements in the overall process of system development Therefore, a field study of some kind was clearly the correct approach to use, but not a conventional survey based on a predetermined questionnaire I was seeking to gain knowledge and understanding of the actual practices used in the field of requirements, and not trying to test some prior hypothesis I was working on the assumption that formal standards and recommended practices for requirements documentation (current theories) do not sufficiently account for the variety of different practices that can be observed in practical situations

I therefore embarked on a series of interviews with experienced practitioners in the field, with a view to building a descriptive theory from the cases studied in this manner It is not good enough to suggest that practitioners are not applying proper procedures, if they are not doing things "by the book " Such theories do not take account of the contingencies that affect practical system development situations If observed practices in requirements documentation differ from expected patterns, then we need a richer theory to explain this variety

The main data source used in the study was this collection of semi-structured interviews, with twenty-eight experienced system developers, conducted at various stages of the investigation I also used specimen requirements documents that I received from the participants, where these were available Some participants tended to illustrate what they were talking about in the interviews with rough sketches of diagrams, and other brief outlines, and I included these in my field notes However, the transcripts and tapes of the interviews provided the main source of data for the investigation

Any alternative research design, such as an intensive case study, would not have suited the research question, so, from the outset, I intended the data collection process to be as extensive as possible With hindsight, it is now possible to distinguish three principal stages in the investigation, though that structure was something that only emerged during the course of the investigation, rather than being a conscious design decision at the outset These three stages were

1 An early pilot study, which covered a small but wide-ranging set of participants
2 A more focused series of interviews concentrating on the requirements documents of one particular software company and its customers
3 A wider main study, aimed at increasing the diversity of system development situations studied

3 2 The initial study

This was undertaken as a pilot study, aimed at opening up and exploring the research question

Seven practitioners were chosen and interviewed at this stage They came from a variety of backgrounds, representing manufacturing, telecommunications, financial services, a public utility company, a government department, and an aerospace company Four of them worked for indigenous Irish companies, of which three were successfully operating in the international market Two others worked in multi-national companies, one American, the other European Each of them had at least eight years experience of system development, some considerably more than this Most of them had already worked for a number of companies, three of them had worked at some stage in consultancy, but most of them currently worked in reasonably large organisations (employing hundreds of people) However, they represented a wide variety of approaches to system development, from extremely formal well-defined processes to extremely informal ad hoc approaches These interviews are summarised in the first seven rows of the table in Appendix A

Great care was taken at this stage to find experienced practitioners who would be confident and authoritative in their responses, people who would not feel that they were being evaluated, just because they were being interviewed, and tend to give text-book answers instead of frank accounts of how they actually used documents in their work It was also important to choose articulate individuals who would be able to make explicit to some extent the tacit knowledge that they use in their everyday work practices

The first person interviewed was someone I knew professionally, and knew to be the kind of practitioner that I wanted to interview, who agreed to be a guinea pig for my first interview Each of the others was recommended by someone else who had worked with the person concerned, and who vouched for their professionalism, their ability, and their having been involved in, and to some extent responsible for, projects that were considered successful

This proved to be a good selection strategy, as I was able to use my personal contacts, but not have to rely on interviewing people I knew personally I aimed to interview practitioners from a wide variety of backgrounds, and was successful in this The strategy also yielded a set of excellent informants In each case then, I approached the person by letter and/or telephone, and explained what my objectives were, and they agreed to be interviewed A copy of a letter I used at the time is reproduced in Appendix B

The interviews, like all the interviews in the succeeding stages, were semi-structured, that is to say they were not confined to a specific set of questions Instead, they were based loosely on a set of guiding questions, designed to elicit open-ended responses and to allow the practitioners to discuss whatever aspects they considered relevant The interview guide that was used for these interviews is reproduced in Appendix C At this stage, I was interested to find out what aspects of 'quality' practitioners considered important in a requirements document, and also how they approached the notion of having a 'complete' statement of requirements

These were not fruitful questions, because these were not issues that the practitioners considered important, but the overall set of responses was instructive Practitioners were not interested in the idea of a requirements document as an end in itself, and therefore its attributes, such as 'quality' or 'completeness' were of no interest to them Instead, they generally regarded the document as a means to an end, although they varied widely in the uses to which the document was put, and also in the extent to which they relied on written documentation of requirements in the software development process

As a result of the initial study, I found that in practice the term "requirements document" covers a variety of different types and formats of documents that are used in system development, essentially to record requirements, but having a variety of roles and uses More significantly, its perceived importance in the overall process of system development also varied widely from one situation to another

The interviews were all recorded on audiotape and subsequently transcribed verbatim This was important, because, although I always took notes during the interviews, these were for the purpose of the interview itself The significance of what was being said did not always occur to me until later, on listening to the tapes over again, or examining the transcripts during analysis Many of the transcripts continued to yield interesting insights even in the subsequent stages, using the theory that emerged later on as a "lens "

In this first stage, analysis was done manually, literally, with a pair of scissors, using printed copies of the interview transcripts I used a procedure called constant comparative analysis The purpose of this is to see what the data has to say, rather than looking for evidence of any particular concepts It was an effective way to learn how to do open coding

I printed out a transcript of each interview, and working with one interview at a time, began to select short passages from the transcript, in which the informant was making some point that seemed interesting, or well made I annotated the quotation with the identifier and page number of the interview before cutting it out from the transcript Each transcript was thereby dissected into short extracts or quotations, varying from one sentence to a few paragraphs Each quotation on being cut out was compared with each of the others that had been extracted thus far, and grouped with similar extracts by theme

When a few transcripts had been analysed in this way, I had begun to make little piles of the extracts, based on themes For example, one of the themes that emerged from the practitioners was the difficulty of writing requirements that are clear This theme or pile contained a collection of statements like the following

> "Essentially, you are aiming at an audience so you are trying to write something that another person can understand You don't necessarily want to bury it in detail but you want to certainly get the concepts across "
>
> "I still think it would be hard to write a document that the users will understand Without being disparaging to users it's just that they are not used to reading great reams of text and being sure that they understand exactly all the implications "
>
> " don't go into the task documents as they are impossible to read, most of them, "

The next stage of this procedure was to label the piles, in order to have an effective way of determining, for each new extract, which, if any, readymade pile of extracts contained similar material, or whether a new pile or theme needed to be started Instead of using brief codes at this stage, I decided to label the piles with propositions, each written on a post-it label The propositions were very general, such as the following statement

- *The detailed contents of requirements documents are not always understood by their supposed audience*

At the end of this procedure, I had assembled evidence from the seven interviews for thirty-five propositions, each supported by at least five quotations, the transcript references for which I entered into a notebook, along with the propositions This notebook enabled me to write a working paper based on the propositions, supported by selected quotations from the piles However, these propositions were all of a very general nature Many of them were also very tentative Here are some further examples

- *One of the most important roles of the requirements document is as an aid in 'getting to agreement' between the parties involved in system development*
- *Whether it is written or not, the concept of agreement is central to the process of finding out requirements*
- *Requirements documents have different uses in different situations*

The full list of propositions is reproduced in Appendix D

The working paper presented evidence that practitioners use a variety of formats for documenting requirements, that although different system development methods dictate what should be documented, and how, that methods themselves are used in an *a la carte* manner, rather than followed precisely, that each organisation has its own way of doing things, and that beyond the level of the organisation, published standards and guidelines have very little impact on the way requirements are documented I found considerable variety in the situations of use of requirements documents, with many different roles for the document, that practitioners viewed a requirements document as a means to an end, rather than an end in itself, and that the extent to which organisations relied on the written documentation of requirements also varied considerably These findings were at variance with much of the conventional literature on requirements engineering, much of which seeks to define the generic requirements process without reference to the need to vary it to suit the situation

However, all of this was very general, and based on a small number of interviews There was no notion of a theory that would explain it all, no conception what the independent variables of such a theory might be, or even what the dependent variable might be, no coherent explanation of the diversity found in practice, other than the proposition that practitioners were doing what worked for them in their own situation I needed a concept of what might be meant by 'situation' in the context of software requirements, and I also lacked a clear concept of the phenomenon of a requirements document, as it exists in practice

At the end of this first stage, I needed to focus the investigation, and I was presented with an opportunity to do so by the next practitioner I approached Rather than looking at a variety of documents in different contexts, as I had been up to this, I was now able to concentrate on one particular format of requirements document and investigate its variety of uses in different organisations

3 3 The second stage

This stage of the investigation was based around a series of interviews with the clients of Blackbird Data Systems (BDS), a small company supplying warehouse solutions, consisting of hardware and software, on a contract basis, to large and small customers in Ireland and the rest of Europe

All of Blackbird's clients are alike in that they require a warehouse application, but their needs are all sufficiently dissimilar that a standard product will not satisfy them, hence they need a tailored solution Blackbird has a glossy brochure and video describing their 'product' called Harvest, but few of their customers buy it "off-the-shelf " A requirements document, called a Specification, is written from scratch for each new client The required solution is then developed by reusing standard software components and developing new ones from scratch The format of the Blackbird document is described in Appendix E

I first interviewed the Blackbird Software Development Manager, who was the architect of the Blackbird requirements document, and also responsible for the requirements on some of the projects I would later look at He supplied me with a specimen document, and explained the format and the rationale behind it He was interested to find out how his clients (or customers, as he called them) regarded the document and the whole process of doing business with Blackbird, and agreed to put me in touch with clients who would be willing to be interviewed about their experiences I subsequently interviewed five different client representatives about their perception of the Blackbird document and their overall experience of the company's particular approach to establishing and documenting their requirements

It was very useful at this stage to be able to focus on the documentation process rather than the document, and it was interesting also to see how the same kind of document worked in different situations Some of the client companies were large organisations with considerable experience of outsourced software projects, one was relatively inexperienced in this area, another was a relatively small Irish company, in the business of supplying large multinational companies based in Ireland

All of the clients that I interviewed were the people responsible for the Blackbird project in their own company Three of them were engineers, with considerable experience of project management, another was the IT managei of a large indigenous Irish company, responsible for several business units in Ireland To get a different view, I requested to be put in touch with a non-technical stakeholder, so the fifth interviewee was the manager of a large commercial undertaking which was relatively new to computerised information systems

I produced two documents as a result of this case study The first one was a report for BDS on how their clients experienced the process of working with Blackbird on their respective projects, and on how they regarded the format and contents of the requirements document The second was a working paper based on the same data but focusing on the role of agreement in requirements documents

In general, the opinions the clients had of the Blackbird document were very favourable Some of them suggested minor changes, for example, the inclusion of screen layouts, or some graphics (the document was mainly textual, supplemented with tables), but others were completely satisfied with this aspect of their interaction with Blackbird

The most important recommendation I was able to make had to do with the Terminology section of the document, essentially a table of terms and their meanings Although it provided for the explanation of various technical terms, it did not cater for the fact that the clients very often had their own local terminology for warehouse operations and data which differed from the Blackbird usage and interpretation

One of the main problems that clients reported to me in relation to reviewing the initial drafts and revising them was problems with terminology Blackbird personnel tended to write the initial draft using standard Blackbird (or Harvest – the name of the product) terminology Some Harvest terms, such as *lot-number* and *delivery*, tended to have different names or different meanings in the clients' everyday local usage There was no provision in the requirements document for aligning these different terminologies, although, as a company, Blackbird was quite prepared, where feasible, to tailor any aspect of Harvest's functionality or user interface to the client's requirements

The second outcome of this stage was a paper dealing with the different kinds of representations that might be used in a requirements document Having a common document format in each situation gave a kind of uniformity to the investigation, at this stage The fact that a standard format was in use facilitated the task of investigating the different uses of the document and its role in the process of discovering, refining, and agreeing requirements The analysis of this set of interviews in combination with the previous ones helped to redefine and re-focus the research question

At the beginning of this stage of the investigation, the focus of the research was on the purposes and uses of requirements documents, rather than the attributes of the documents themselves This perspective regarded the requirements document as an instrument in the process of establishing, representing and agreeing requirements, rather than an end in itself This was in common with the views of all of the practitioners and stakeholders interviewed thus far, and in contrast with much of the conventional literature on requirements

At the end of this stage, there were two major strands to the research question

1 What is the role of context in determining the different uses of requirements documents, and how can contexts be differentiated from each other in a way that is relevant to the process of requirements documentation?
2 The question of why abstract models, which were so constantly favoured in the theoretical literature on requirements engineering and system development, were so infrequently and sporadically used in documenting requirements in practice

3 4 The third Stage – the main study

The objective at this stage was to find some theoretical variables that would help to explain the observed differences in the style of requirements documents used in different situations, and to find some way of describing different styles of document The best analogy that I had for the notion of a 'requirements document style' was the idea of an architectural style from the domain of software architecture (Shaw and Garlan, 1996) Another aim was to explore, as far as possible, the incidence of variety in requirements documentation

3 4 1 Variety

The practitioners I interviewed came from as wide a variety of situations as possible All of them worked in organisations, varying from small to very large While a few of them were working at developing systems in-house for the organisations in which they worked, most of them were working in a more formal contractual situation, developing systems for clients or customers in other organisations

The application domains that they reported on were also very varied While some respondents were involved with business applications, for example financial and manufacturing, others were concerned with developing embedded systems Several of them worked in the field of telecommunications systems The respondents from this domain represented a wide range of different types of applications, dealing with billing, network management, etc as well as the typical switching and transmission systems

These situations displayed a variety of contractual situations, from bespoke systems to software products but excluding the production of a shrink-wrapped software product for the open market

3 4 2 Two areas of focus

In addition to aiming for variety of situations in choosing people to interview, I also decided to concentrate on talking to a few more practitioners involved in some particular areas, which had been particularly interesting in the pilot stage of the investigation Telecommunications was one This is an important application area, particularly telecommunications switching, which has a relatively long tradition of engineering that predates the era when telecommunications became a software rather than a hardware application In the event, this area yielded a variety of perspectives, including that of the most experienced practitioner that I interviewed

Another area that I concentrated on to some extent was that of ERP (Enterprise Requirements Planning) systems There were two reasons for this choice One reason is that such systems are first developed as highly configurable packages which are then tailored to fit the requirements of the particular situation, thereby shifting the emphasis at the users' site to the requirements gathering and analysis phase, away from considerations such as design and programming, which are done in advance

I was able to look at these systems from two perspectives, that of the company developing and supplying the software, and that of the organisation customising it for their clients I had already interviewed a practitioner working in a client company in the pilot study The requirements specification for the package itself has a much larger scale of complexity which has to be managed quite differently

Another reason for concentrating on this area is that though the idea of configuring a package is not new, this has become the dominant trend in all areas of business applications in recent years, including university administration In this domain, I interviewed a key user in a client organisation, a university administrator who had been seconded full-time for a year on a project the university was carrying out with an external software company The aim of this project was to re-structure and develop a commercial student records package that could be customised for use in his organisation This project, and the university's involvement in it, arose out of severe difficulties they had already experienced in trying to customise the previous version of the package to suit the requirements of the university

3 5 How I approached coding and theory building

3 5 1 Manual coding

The manual analysis strategy initially used for the interviews was outlined earlier This is best described as open coding, as it was guided only by the data itself This was my first experience of either coding or doing grounded theory, so I was only learning how to do it at that stage

Having printed out a copy of each interview, and working with one interview at a time, I began to select short passages from the transcript, annotating them with the identifier and page number of the interview, cutting them out with a pair of scissors, and assigning them to little piles on the desk I used the technique of making "constant comparisons" with the contents of the existing piles

I continued in this way until all the transcripts had been sliced up, and I had thirty-five piles each labelled with a proposition concerning the role or use of requirements documents Each proposition was supported by five to ten quotations from the data, which I copied into a notebook The main advantage of this approach was the way it helped me to learn to let the data control the analysis process Without any predetermined codes, or even any notion of what the codes might be, it facilitated finding the meaning that was present in the data

The cutting up of the pages had a serious disadvantage I could only use each quotation once, and many of them could have been used several times over, or more usually, some part of a cutting could have been assigned to a different pile (proposition) Alternatively, I could have continued by starting again with another copy of each transcript, and cut out a different set of quotations, but computer-aided data analysis provided a better way to proceed

3 5 2 Computer-aided analysis

Some researchers use a simple approach to computer-aided qualitative data analysis, for example Seaman (1999) gives a detailed description of her way of working with interview transcripts A word processor can be used to search the text for specific keywords, leading to the identification of paragraphs which can then be annotated with a suitable code, preferably in a different font to distinguish it from the original transcript In this way, as many codes as desired can be attached to a particular paragraph

Before computers were widely adopted in qualitative research, codes were typically annotated in the margins of a copy of the typed transcript But now there are several sophisticated qualitative data analysis packages on the market, facilitating sorting, searching and selection of data and codes in various ways The one I used is called Atlas/TI The letters TI stand for textual interpretation, but Atlas can also be used for graphic and sound data, as well as the more usual text (Muhr, 2000)

This package allowed me to organise and keep track of hundreds of codes and hundreds of pages of interview transcripts, as well as providing the means to create axial codes that were more theoretical and core categories, graphically linking them to each other and to the relevant open codes

The transcripts of the interviews carried out thus far, in the preceding stages, were entered into the Atlas database, and coding began afresh This time the codes were simpler than before Instead of propositions, they tended to be single words or simple phrases describing concepts, such as the ones illustrated in Table 3 1 The coding process on the computer consisted of reading the text of an interview transcript, selecting individual words, phrases, or, more usually, longer quotations, and assigning to the selected text a code that indexed it The full list of codes is in Appendix F

acceptance testing
accepted
accumulated changes
accuracy
ad hoc reporting
afterthefact
agreed requirements
agreement
agreement not in document
alpha test
an adequate design
analysis
answers all their questions
application domain
application expertise
application note
application problems
applying knowledge
architecture
architecture first

Table 3 1 Some sample codes

These codes were more basic and more conceptual than the previous codes, which were all propositions, and also more varied in their subject matter, dealing with the contexts, uses, and contents of requirements documents, as well as system development generally

3 5 3 The emerging theory

The analysis carried out in the initial stage and at the case study stage was aimed at Maykut and Morehouse's level 2, what they called the weaver level, (Maykut and Morehouse, 1994) and the results were written up in the form of sets of general propositions or hypotheses regarding the role of the requirements document For the main study, I aimed at the third, most abstract level of interpretation, the theory builder level An outstanding example of a grounded theory reaching this level of interpretation is found in Orlickowski's (1993) study of the adoption and use of CASE tools in organisations, which presents a theoretical framework linking the social context of system development with the different outcomes of CASE tool adoption Seaman and Basili (1998) present their results as a set of independent and dependent variables, and a network showing the relationships they identified between them I aimed to have something more abstract than a set of propositions, but for a long time, I could not envisage a causal model of requirements documentation that would be based on a set of independent and dependent variables

The notion of causality is central to scientific enquiry, but, beyond the level of physical phenomena, most events cannot be explained simply in terms of cause and effect I reasoned that the inherent complexity of situations involving system development could not be reduced to a few causal variables The meta-model paradigm of grounded theory provided a more appropriate set of categories for explaining things in terms of situations, based on the conditional matrix, but I did not rely on this I tried to follow the advice of Miles and Huberman (1994) who recommend that in a multi-case analysis, explanations should be presented in matrix form This gives a picture of the possible variety of causes over a range of cases, rather than a general explanation of the factors pertaining to situations

3 5 4 A classification framework

The idea of a matrix form of explanation, and the idea of a document style came together in the form of a classification framework Mary Shaw and Paul Clements (1996) had published a feature-based classification of architectural styles in the form of a single matrix, covering all the different styles found in the literature analysed using a common framework of features The mam features of software architectural style they included were Constituent Parts, Control Issues, Data Issues, and Control/Data Interaction, in effect, a theory of what were the salient dependent variables of software architecture

So, I began to aim at the idea of proposing a classification framework, in particular a faceted classification (Star, 1997), identifying the factors that seem to be related to the variation found in different kinds of requirements documents and explaining the kinds of documents that tend to be used in different situations Such a classification would lend itself to being tested, verified and amended as theoretical sampling proceeded

I envisaged the resulting classification as a table, or rather as two related tables, one indicating the features of situations that would make a significant difference to requirements documentation practices, the other indicating the style of document appropriate to the situation As it turned out, there were two main problems with this vision It was impossible to put in a few plain words the values of the variables I was coming up with for the first table, so that they would both fit in the cells of the table and make sense Secondly, a faceted classification turned out to be impractical, because the facets were not completely independent of each other, being drawn from a network of related concepts which formed one of the main strands of the supporting theory

This network of codes was the result of axial coding using the results of open coding of the growing collection of interview transcripts, and therefore the most grounded part of the theoretical framework In the next section of the thesis, in Chapter 5, it is introduced Then it is used as the basis of a framework for classifying situations for the purpose of documenting requirements, presented in Chapters 6 and 7 But first, Chapter 4 deals with the other main strand of the thesis, a grounded theory of requirements documentation

Part Two Findings

Overview of Part Two

What are requirements?

What do people talk about when they talk about requirements? It depends on the situation, but discussions about software requirements are not just about the capabilities and properties of software At a more general level, software requirements are about problems and problem situations, they are also about solutions about the need to state the problem in such a way that it can be solved, ultimately, by a software solution Christiane Floyd (Floyd, 1995) discusses the centrality of problems and problem solving in computing, and asks

"What kind of entity is a problem? What is its ontological status? Does it have a way of existing on its own? Whose problem is it? " (Floyd 1995)

A similar series of questions could be asked about requirements What kind of entity is a requirement? Can it exist on its own? Where does it come from? Where do different kinds of requirements come from? Who owns them? This thesis explores how these questions might be answered by an ontological analysis of the domain of requirements

Two levels of the concept of requirements

In the context of system development, a requirement may be regarded as two related things A requirement is defined as

(1) a function, capability, or property required of a proposed system **and/or** it is

(2) the statement of such a function, capability or property

When we talk about the elicitation of requirements we are talking about the first of those meanings, i e the requirements themselves Similarly, when we discuss the issues surrounding the articulation of requirements or the need for agreement on requirements, clearly that is what we are talking about

On the other hand, when we discuss topics such as requirements inspection, requirements validation, requirements management, or requirements traceability, we are referring to inspecting, validating, managing, etc the statements of requirements rather than the requirements themselves

A requirements *statement* is a semiological object We use it to stand for the substantive requirements we are talking about When we want to prioritise requirements, for example, we attach priority values to the statements of requirements, not directly to the requirements themselves We all know that requirements change, but we acknowledge this by changing the relevant statements of requırements, because these "requırements" do not change ın the same way, by themselves

Wrıtıng down requırements ıs a necessary part of the requırements process, but once a requırement has been wrıtten down, the expressıon of ıt, as well as representıng the actual requırement, takes on a separate ıdentıty and a sıgnıficance of ıts own

Most authors on requırements engıneerıng do not make thıs dıstınctıon, so why bother? Because the expressıon of a requırement ıs not the same thıng as the requırement ıtself Requırements engıneers, partıcularly researchers ın requırements engıneerıng need to take thıs dıstınctıon on board An ontology of requırements ıs a small step ın thıs dırectıon, emphasısıng the ontologıcal precedence of requırements-as-needs over the statements and models that represent them ın requırements documentatıon

Dısambıguatıng "requırements"

Thıs thesıs emphasıses the dıstınctıon between actual requırements (or requırements-as-needs) and the requırements statements that document them It focuses first on requırements statements or requırements-as-texts, and the dıversıty of ways that requırements are documented ın practıce It then looks at the requırements themselves, and at the dıfferent sources of requırements, and then looks at how requırements-as-needs shape the context of wrıtıng requırements-as-texts The maın findıngs of thıs research are presented ın the next four chapters as follows

Chapter 4 focuses on requırements-as-texts, and explores a number of dıfferent themes of varıatıon found ın the documentatıon practıces of the practıtıoners ıntervıewed Thıs chapter uses a taxonomıc approach to analysıs, relatıng dıfferent categorıes of requırements statements usıng the usual hıerarchıcal **ısa** relatıonshıp It concludes wıth a sımple model of dıfferent types of requırements statements

Chapter 5 looks at requırements per se, as needs rather than as texts, and uses a dıfferent type of analysıs to develop an ontology of requırements The purpose of thıs analysıs ıs to explore the relatıonshıps between dıfferent types of system and software requırements Rather than the more usual taxonomıc relatıonshıps used ın most publıshed ontologıes, these are ontologıcal precedence relatıonshıps, lınkıng requırements-as-needs to theır ontologıcal predecessors, on whıch they depend for theır exıstence, and theır ontologıcal successors, whıch ın turn depend on them

In the requirements engineering literature, for example in (Greenspan et al, 1994, Mylopoulos, 1998), the idea of ontology has been used to compare and contrast the different world views embedded in different approaches to conceptual modelling The entity-relationship approach holds that the real world consists of entities and relationships, while the object-oriented approach views the world in terms of objects and other related concepts It is important to recognise these ontologies because they frame the way that problems and solutions are understood A different level and type of requirements ontology deals with the way that the domain of requirements engineering itself, including problems and solutions, is construed by its practitioners This is the purpose of the ontology presented in this thesis It draws attention to how requirements are related to their different sources, particularly those identified as 'root concerns,' and to the contexts in which different types of requirements arise

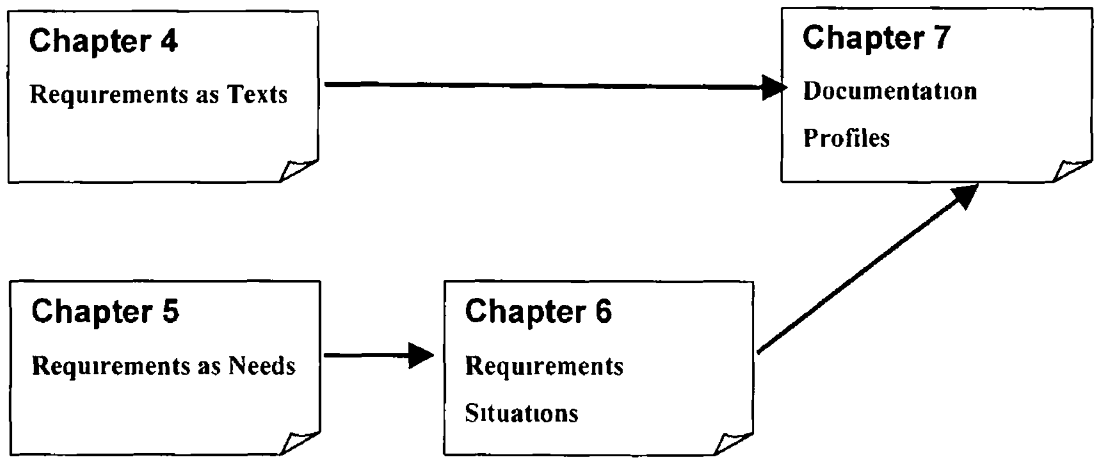
Chapter 6 presents a scheme for classifying different system development situations on the basis of their most important requirements concerns It is based on the grounded ontology presented in Chapter 5, and makes particular use of the root concerns identified there It proposes that different types of requirements situations can be distinguished by their varying emphasis on different kinds of requirements-as-needs Seven typical situations are presented based on this classification scheme

Chapter 7 presents seven different profiles of requirements documentation based on their situation of use Each of the seven typical situations that were presented in Chapter 6 is here associated with a typical documentation profile Each profile defines the role and importance of different types of requirements statements in a typical situation This scheme is based also on the analysis of requirements statements outlined in Chapter 4

Explaining requirements

The relationships between the four chapters are shown below Chapters 4 and 5 are relatively independent of each other, dealing with the semiological and substantive aspects of requirements respectively, while Chapter 6 builds on material introduced in Chapter 5, and Chapter 7 builds on both Chapter 4 and Chapter 6

Chapter 7 is offered as an explanation of observed phenomena Chapters 5 and 6 introduce and develop the independent variables in the explanatory framework, whereas Chapter 4 establishes some dependent variables Chapter 7 links them together and explains the observed diversity of requirements documentation in terms of the variety of situations in which requirements documents are written and used



Relationships between the Chapters in Part Two

Chapter 4 A Taxonomic Analysis of Requirements as Texts

*"The notion of giving something a **name** is the vastest generative idea that was ever conceived "*

(Suzanne K Langer, American educationalist)

4 1 Introduction

The literature on requirements engineering seems to be committed to the idea of classifying requirements (IEEE, 1998, Kovitz, 1999, McDermid, 1994, Robertson and Robertson, 1999, Wieringa, 1996) Many of these classification schemes begin by distinguishing between functional and non-functional requirements For example, the IEEE standard for the Software Requirements Specification (IEEE, 1998) distinguishes fourteen types of requirements, divided into functional requirements and thirteen different types of non-functional requirements The Robertsons identify seventeen different types of requirements divided into product constraints, functional requirements and non-functional requirements (Robertson and Robertson, 1999)

None of the classifications in the literature seems to distinguish between substantive and written-down requirements, although the term ‘requirements’ covers both of these An essential theme of the analysis presented in this thesis is the distinction that it makes between the substantive requirements that exist in a situation (requirements as needs) and the written statements that represent or define those requirements (requirements as texts)

For these reasons, rather than begin with any of the existing classification schemes, I have taken the grounded theory approach of beginning with the interview data and employing the categories that emerge from the empirical domain to propose a different way of classifying requirements This unorthodox way of classifying requirements has been created for a specific purpose, to talk about the main varieties of textual requirements instead of the more usual breakdown of types of requirements per se This will allow us to examine the question of how “requirements as texts” may vary in relation to any other particular factor that we may want to consider

The taxonomic analysis of “requirements as texts” presented in this chapter focuses on the differences between the various kinds of written statements that are used in documenting requirements It explores the variety of ways of expressing, organising, and using requirements in a written form, before dealing with the various written components of a requirements document Since this chapter is concerned mainly with an analysis of empirical findings, these will be presented here, along with illustrative quotations from the interviews with the practitioners, and will not be further compared to the literature until a later stage, in Chapter 10

4 2 Domain analysis

In his books on ethnographic research methods, Spradley (1979, 1980) presents a very useful set of heuristic techniques for performing domain analysis Domain analysis is the term given in ethnography to the analysis of a specific domain A domain in this context can be any cultural phenomenon or area under investigation, such as kinship among a specific tribe of people Spradley's own research was an exploration of the culture of tramps living on Skid Row (Spradley, 1972a)

One of the most important domain analysis techniques is called taxonomic analysis, a procedure for systematically discovering the variations of terminology and meaning that occur within a domain Spradley used it to analyse the different domains in the life of an urban nomad, such as the variety of terms used for different 'flops,' places where a tramp might sleep for the night Another example from the ethnographic literature dealt with the variety of categories of beers enjoyed by the people of Munich (Hage, 1972) I have found this analysis procedure extremely useful in searching for patterns of variation in the domain of requirements as texts

The objective of taxonomic analysis is to produce a hierarchical decomposition of all the sub-domains within a particular domain, related by some taxonomic relationship, such as is-a-kind-of, or is-part-of The method of analysis is essentially bottom-up, and is intended to be used to discover what are called "folk domains," which are effectively the terminology which is local to the culture being studied In grounded theory, these terms are called 'in-vivo' codes, as distinct from the theoretical codes which are named or created by the researcher I used it with a mixture of both types of categories

The starting point in taxonomic analysis is to look for categories in the empirical data which are all related in that they (could) belong together, under some heading, or cover term as it is called Examples of prototype cover terms include *kinds of* or *ways of* or parts *of* Examples of actual cover terms include "kinds of trees," "kinds of deciduous trees," "parts of a tuna boat," "parts of the deck," etc These are only a few examples of the most common types of cover terms Spradley gives a list of them and points out that several more are possible (Spradley, 1979) The cover term represents a binary relationship between the domain of the cover term and its sub-domain These relationships between domains and sub-domains are transitive and non-reflexive, forming the basis of a hierarchical framework

4 3 Domain analysis of requirements as texts

My empirical data for the purpose of this analysis consisted of transcripts of the interviews in which practitioners discussed their own documents and others they had worked with, plus a selection of sample documents given to me by the practitioners Having already analysed the interview data by open coding using ATLAS ti as a tool, I then generated a code family called "requirements as texts" This consisted of a collection of basic categories, both theoretical codes and in-vivo codes, all related to the theme of requirements as texts, as distinct from the separate theme of requirements as needs and other aspects of projects and system development generally

In-vivo codes are more grounded than theoretical codes, because they are the terms used by the informants themselves, and presumably reflect local usage in the situation they are talking about, whereas theoretical codes are the researcher's own terms, which may be expressions in general use in the literature, or terms coined by the researcher to capture some insight or observation about the data In some ways, the domain of requirements engineering is like any of the "folk domains" studied by Spradley and other ethnographers, in having its own terminology, or jargon, but in a sense it is different, in that the terminology used by the practitioners was not new or strange to the researcher, although its usage and meaning in practice still needed to be explored and understood

The main point of the analysis, therefore, was not just to find out which terms were used by practitioners to describe their documents, but to try to capture the usage and meaning of these terms among the practitioners I interviewed Because there is a much greater exchange of ideas and terminology between the research and practice communities of requirements engineering, there is no particular value attached to capturing in-vivo codes as such, as there would be in a more typical ethnography In my analysis, it frequently happened that a code that was collected in-vivo in one case was applied as a theoretical category to describe the same phenomenon in another case

I set out to build a grounded taxonomy of requirements as texts, with the intention of constructing a framework that would have at least a few levels of hierarchy I began by bringing together the codes I had collected and generated indicating different types of documents Originally I intended to restrict the taxonomy to two types of hierarchical relationships *is-a-kind-of* and *is-part-of*, but gradually I began to reconsider that restriction as I found that (a) there were other types of relationships which would help to illuminate the domain, and (b) the idea of a hierarchical structure or a strict taxonomy did not fit very well to the empirical data

The result was rather different from the original intention, consisting of a collection of analysis themes I explored **different names** and **types of documents**, different ways of **organising** and **expressing requirements**, different **ways of using requirements documents** and different **components of written requirements** In grounded theory, these categories are called axial codes They represent the intermediate level of analysis between open coding and selective coding

4 4 Results of the analysis

One problem in trying to understand what is going on in the practice of requirements documentation is that many different names are used for the documents, but it is not always clear which documents are the same though they have different names, or which are different though they might be called the same name Table 4 1 shows the result of this, a list of all the different names that the practitioners used for the various requirements documents that they used

Different Names for Requirements Documents	Business Blueprint
	External Description
	External Specification
	Feature Description
	Functional Description
	Functional Requirements
	Functional Specification
	Global Requirements Specification
	Internal Specification
	Invitation to Tender
	Request for Proposals
	Requirements Definition
	Solution Investigation Document
	Specification
	System Requirements
	Task Definition
	User Requirements Specification

Table 4 1 Different Names for Requirements Documents

Many different types of documents are used in addition to the main requirements specification documents, and some of these are listed in Table 4 2 The extent to which these were used was related to the contents and purposes of the main requirements document For example, some of the practitioners relied heavily on **interview records** in addition to the official requirements documents as the basis for further work on the system Several of them also relied on the **minutes of meetings** with clients as additional documents **Memos** and **emails** were also mentioned, both as ways of communicating with other stakeholders and as documents that would be regularly filed along with the formal documentation of the requirements

In some situations, and after certain stages of the project had been reached, changes to the requirements would need to be communicated using pre-specified formats, sometimes called **change requests**, or **problem reports** In the telecommunications domain, the standard requirements for specific features were typically already documented in official **standards documents**

Different types of ancillary documents	Interview records	
	Minutes of meetings	
	Different types of short communications	Memos
		Emails
	Change requests	
	Problem reports	
	Standards documents	

Table 4 2 Types of Ancillary Document used in relation to Requirements

4 5 Different types of documents

Some informants explained that they typically worked with two different types of requirements document in the one project, for example an **internal specification** and an **external specification** In such cases, it is obvious that these have complementary roles Other practitioners worked with what they called a **URS**, or **User Requirements Specification** It was clear that the role of **URS** in one case is similar to the role of **external specification** in another case In fact, I have the evidence that they are similar in their contents as well as their purposes, but not to such an extent that I can say they are synonymous, because I found that every requirements document is different

Despite the name, the **URS** in several cases was not written by users, or for users, or, to any extent, with input from users In practice, the term seemed to reflect the perspective from which it was written, that being the perspective of a user who might have a limited interest in or understanding of the internal workings of the intended system, or product

> "The URS, which is the product manager's document in this he specified hardware and software requirements, user-level requirements of what he expects the product to do "

Some system development methods dıvıde requırements documents ınto two types reflectıng thıs external/ınternal dıvısıon In practıce, havıng two types of documents seemed to be related to the use of methods and the extent to whıch the method was beıng rıgorously applıed However, ın most cases, the practıtıoners reported that methods were beıng used *a la carte*, or adapted ın ways not ıntended by theır orıgınators, to suıt the sıtuatıon For example, the need for two dıfferent documents was regarded as appropriate only for certaın cırcumstances, such as large projects, and practıtıoners would devıse a hybrıd type of document to suıt the sıtuatıon

> "So the abbrevıated specıfıcatıon was a combınatıon of these two that we actually used for thıs project In the partıcular area that I worked ın, we would have used the abbrevıated specıfıcatıon much more than the standard ones because we never really had bıg projects That sort of huge tıme-scale "

Every requırements document ıs unıque, because every system ıs dıfferent ın some ways But there are broad areas of sımılarıty as well as dıfferences to be found, ı e patterns of varıatıon ın the ways the contents are organısed, the ways requırements and other contents are specıfıed, and ın the types of requırements they contaın

The practıtıoners used a varıety of ways of **organısıng their documents**, of whıch hıerarchıcal breakdown based on functıonal decomposıtıon was the most common, followed by organısatıonal schemes based on workflow or busıness processes, though some used dıfferent kınds of tables and matrıx or record formats to organıse the requırements However, these were very much dependent on the nature of the contents

Most practıtıoners expressed a preference for 'plaın Englısh' for **expressıng requırements**, and thıs ıs by and large the most wıdely used approach But varıous types of **models** were also used to represent some aspects of the requırements, especıally the problem domaın Semı-formal structured models were used for requırements ın some sıtuatıons, whıle ın other sıtuatıons, the use of modellıng technıques was postponed to a later stage of development We wıll return to thıs poınt ın a later chapter

It ıs reasonable to suggest that dıfferent types of documents mıght have **dıfferent roles** or **uses** and **dıfferent types of contents**, and these are two areas where a consıderable amount of varıatıon occurred ın practıce

4 6 Dıfferent uses of requırements documents

What ıs the use of the requırements document? Table 4 3 shows the varıety of roles of the requırements document that the practıtıoners talked about ın descrıbıng how they used the documents in their work Several of them emphasised the role of the requirements document as a **working document**, going through a succession of versions and revisions The earlier versions, often called drafts, were mainly regarded as having the role of a **discussion document**

> "As for the importance of the document, they would look at it and we would talk through it and read it out to them or explore things and they would say something like 'I don't agree with that' or 'I'm not happy with this' and we would resolve this "

On the role of the requirements document as a device for **documenting the agreement**, the practitioners' opinions and accounts varied considerably from one situation to another Some of them saw the document as defining the agreement, particularly in contractual situations, while others saw it as having a much more subtle role in the process of reaching agreement

In many cases, the practitioners expressed the view that the essence of the requirements document was a tool that was instrumental in the process of getting **to agreement** rather than merely a means to **document the agreement** that had been reached

> "One purpose was to get agreement from the users on the functionality of the system "

Different uses of requirements documents	as a discussion document
	as a method of communication
	as a working document
	as the basis of the architecture
	basis for prototyping
	input to costing
	input to design
	input to project planning
	input to testing
	input to user manual
	looking for solutions
	tells users what they are getting
	to evaluate proposals
	to document agreement
	to get agreement
	to articulate knowledge
	to express understanding

Table 4 3 Different uses of requirements documents

Most of the practitioners took the view that the requirements document was a **means to an end**, rather than an end in itself In some cases, this view was strongly expressed in terms of its use as a means of **getting to agreement**, while many of them saw an important role for the requirements document in its subsequent use in one or more later stages of the development project, such as prototyping, architecture, design or testing The specific phases to which the requırements document was dırectly ınput tended to vary consıderably from one sıtuatıon to another

> "Our archıtecture desıgn phase would take these requırements and ımplement a data model from them and also ımplement a dıalogue flow, conversatıon control flow, for the project "
>
> "Then we went through a stage of looking for solutıons to the project, we ınvestıgated a number of packages It was used to evaluate the varıous proposals that came ın "
>
> "That's a very crıtıcal poınt ın the project because at that poınt, I'm then able to use thıs document to go to the next stage whıch ıs to produce the plan and costıng and talk about money and tıme-scales "

The bıggest area of varıatıon, even dısagreement, between practıtıoners regardıng the use of the requırements document related to ıts use as a **method of communıcatıon**, especıally for exchange of ıdeas wıth the end-users of the system Some strongly expressed the vıew that, no matter how well ıt was wrıtten, no document could be relıed on for communıcatıon wıth users

> "Documents, they have a place, but don't rely on them "
>
> "I stıll thınk ıt would be hard to wrıte a document that the users wıll understand Wıthout beıng dısparagıng to users ıt's just that they are not used to readıng great reams of text and beıng sure that they understand exactly all the ımplıcatıons "
>
> " talkıng through ıt because people are not good at wrıtıng and they are not good at readıng One ıs as bad as the other "
>
> "It ıs very few people that I have found that when you wrıte a document and when you talk about ıt afterwards wıth them, yes he understood what I meant "
>
> "People sımply don't have that skıll to wrıte down what exactly they mean and I suppose the converse ıs that a lot of people don't know how to read a document ın the same way "

Other practıtıoners expressed more confıdence ın theır own abılıty and ın the possıbılıty of wrıtıng clearly enough to be understood by everyone concerned

> "You can see here (I lıke to) put the thıng ın plaın Englısh, short sentences, lıttle poınts, so even ıf ıt's not numbered one, two, three, four, then ıt's as lıttle poınts ıt's easy to read, ıf you feel somethıng's ımportant, you can easıly put a rıng around ıt or a lıne around ıt "
>
> "There probably ısn't any substıtute for ınteractıon wıth the users but on the other hand, ıf they get a good document at the start ıt's tellıng them exactly what they want, then ıf that's clear to them, they are clear on what they're gettıng "
>
> "It's wrıtten so the customer can read ıt and understand ıt and agree to ıt, so ıt would hopefully be wrıtten ın plaın Englısh usıng the customer's termınology It's not journalıstıc, I mean ıt's not understandable to the world at large, ıt's most certaınly not only understandable by ourselves I mean we seek to avoıd that ın all parts of thıs document

There was a certain amount of opinion offered that the usefulness and acceptability of written communications and formal documentation depended on the ethos of the organisation in which the system was being developed One practitioner put it in the following way

> "This company is a get up and go company, and rather than planning acting and reviewing it believes in acting if you set out to nail somebody down on their requirements you wouldn't get a lot of sympathy from the person you were trying to nail down, or much understanding "

On the other hand, some practitioners emphasised the usefulness of having things in writing after the event, in order to explain the rationale for certain decisions that had been taken For example

> "There happened to be a change of payroll supervisors very soon after the project went in He wanted some changes and we were able to go back to the document and say well listen this was the reasoning and I suppose it made him go back and think about the reasoning behind it "

Another, perhaps more noteworthy, perspective on the role of the document was expressed in the insight that for some practitioners, the act of writing the document clarified their own understanding of the requirements, in giving them a vehicle that both required and allowed them to **express understanding** and to **articulate knowledge** about the application

> "Part of this was to help us to understand, to detail the stuff to get a common understanding "

> "We had to try and understand everything We had to go through from not understanding anything about the protocols that were involved, what they were talking about, to actually having done all the research to and finding out what was really going on in this area "

> "We were new in the area they were asking us to work in Part of our problem was actually finding out the information at the time, so we put down on paper what our understanding of it was, along with investigating the solution to it "

4 7 Different types of contents in requirements documents

It would seem likely that a good place to search for variations in the specific contents of requirements documents would be in the Tables of Contents found in sample documents provided by the practitioners, but this is not as straightforward as it might appear The terminology used for contents, and the contents themselves differed widely to such an extent that they could not be directly compared This is where the taxonomic analysis procedure became extremely useful

In practice, there is no widely used template or even agreed list of contents for a requirements document Some of the practitioners were familiar with the idea of using a standard Table of Contents for requirements documents This was particularly the case where a defined system development method was being used, but methods themselves vary One practitioner found the standard Table of Contents a useful strategy for gathering requirements in rapid application development

> "Okay To start them off we'll have put together a draft table of contents We might try and get it from say some standards document Like the IEEE standard for requirements specs If the company have an internal standard themselves you just follow that "

The results of the taxonomic analysis of the domain **contents of documents** are summarised in Table 4 4 A more detailed breakdown is included in the Appendices Most of the cover terms in the table are theoretical or in-vivo codes generated from the interview data or found in the sample documents The principal cover terms are given in the second column of the table They are **Problem Domain Descriptions, Required effects, Proposed solutions, Goals, Constraints, Issues,** and **Document information**

Two of these, however, are borrowed or adapted from the requirements literature The cover term **Problem Domain Descriptions** is borrowed from Ben Kovitz's checklist of requirements contents (Kovitz, 1999) **Required effects** is a cover term which I adapted from Jackson's concept of an effect, a term which is also used by Kovitz A requirement per se is defined by Kovitz as the "*effect the software is required to produce in the problem domain* "

A variety of different types of **Required effects** were employed by the practitioners in their requirements documents The most important categories were different types of **Features,** different types of **Transactions,** different types of **Information requirements,** different types of **Behavioural requirements,** and to a lesser extent, some **Transformation requirements**

In the remainder of this section, each principal category of contents is presented under its own sub-heading The most significant of these, in terms of their size or volume within any document, are the first three principal categories **Problem Domain Descriptions, Required effects,** and **Proposed solutions** The other principal categories as outlined in the table are more important in terms of their function in the different roles of the requirements document explored in the previous section In the concluding section of this chapter, these main categories will be reviewed and condensed into a simple taxonomy of requirements as texts

Different Types of Contents	Problem Domain Descriptions	Business Processes
		Operations
		User actions
		Tasks
		Events
		Assumptions
		Rules
		Terminology
		Volumes
		Company Background
	Required effects	Types of features
		Types of transactions
		Types of information requirements
		Types of behavioural requirements
		Transformation requirements
	Proposed solutions	Database
		System Architecture
		Prototypes
		Feasibility analysis
	Objectives	Goals
		Business case
		Non-Functional Requirements
	Constraints	Behavioural Constraints
		Design Constraints
	Issues	Closed issues
		Open issues
	Document information	Document history
		Circulation list
		Authorship
		Table of Contents
		List of reference documents

Table 4 4 Types of Contents

4 7 1 Problem domain descriptions

This category covers those statements, or collections of statements, that describe the problem domain, the context in which the intended system will operate The practitioners' documents different types of statements in their problem domain descriptions, including the following categories **Business Processes, Operations, User actions, Tasks, Events, Assumptions, Rules, Terminology, Volumes,** and **Company Background**

The **problem domain description** as a whole typically deals with all the relevant aspects of the problem domain, and frequently provides the context in which the other types of requirements statements (such as the requirements per se) are to be understood It often includes definitions of **terminology**, such as a **glossary**, or a list of explanations of **acronyms,** etc It may include relevant **assumptions** about the problem domain In some kinds of applications, the existence of **standards** is an important aspect of the problem domain, and requirements may refer frequently and repeatedly to the relevant standards document(s)

The various types of statements under this heading in practice ranged from natural language descriptions of business **functions** and **processes** to models or other representations of equipment functions and operations It is implicit in these descriptions that the problem domain will continue to behave in the way described, even after the intended system has been installed For that reason, problem domain descriptions are often written in the present continuous tense Any planned changes to the actual problem domain may be included in the description without further comment

The statements themselves refer to objects of interest in the problem domain and the ways that they interact Here are some examples

From the specification for a clinical neutron therapy system

> "The isocentric treatment unit (ref to figures in the document omitted) allows patient treatments with beams coming from different directions The radiation source, in this case the beryllium target, is mounted on a gantry arm, which can be rotated around the patient "

From the requirements document for a warehouse information system

> "Pallets containing box packs or drums are moved to the P&D locations at the end of the aisles by the Hand-Truck and then on to rack locations by Man-Up Truck "

From the RFP for an ambulance service command and control information system

> "The opening of the new Regional Ambulance Command and Control Centre in the near future will result in all emergency calls being routed through this single point The new centre will be manned on a 24-hour basis and will have regional responsibility for emergency call management "

These descriptions are not requirements as such, but they were often, though not in all cases, an essential part of the requirements document Where they were used, they had an important role in defining the context in which other types of requirements statements were to be

1 defined,
2 interpreted, and
3 understood

The problem domains referred to in the extracts above include **things** such as equipment, ambulances, pallets, trucks, materials, packs, and **people** such as patients, therapists, drivers, and dispatchers as well as **signs** such as serial numbers, orders, prescriptions, and **events** or **messages** such as emergency calls, etc **Business rules**, which the system is intended to enforce, and assumptions about the behaviour of entities in the problem domain, are often documented in these descriptions

The practitioners' **Problem domain descriptions** were sometimes written in natural language, but they were often represented using other techniques, such as diagrams Entity-Relationship diagrams, Data Flow Diagrams, State Transition Diagrams, and Class diagrams were the most usual types of diagram used In those cases, some natural language description was always used to supplement the models Or models were used to supplement the descriptions which were predominantly written in natural language Another way which was commonly used by the practitioners to supplement the textual description of the problem domain was **tables of data** describing the numbers, volume, frequency or other measurements of the relevant entities and activities in the problem domain

4 7 2 Required effects

The statements in this category express the actual particular requirements, the functionality and other observable attributes and behaviour needed or anticipated in the new system They include **Features**, **Transactions**, **Information requirements**, **Behavioural requirements**, and **Transformation requirements** All except the last of these terms are in-vivo codes, categories based on terms that were used by the participants themselves One or other of these types of statements tended to predominate in the relevant sections of the documents discussed by the practitioners The type of required effects seemed to be contingent on the type of application being discussed For example, **Features** were strongly associated with telecommunications systems and also with market-oriented software, while **Information requirements** were most closely associated with database systems and with business applications

To fully understand a statement of a **required effect**, it must be interpreted in conjunction with the **problem domain descriptions** These statements were often presented in the form of long lists, often numbered for reference The numbering scheme varied from straight sequential 1- n, or decimal numbering reflecting a hierarchical breakdown which was used to organise related statements into groups Collectively, statements of **required effects** defined the interaction between the required system and its environment Here are some examples

From the RFP for an environmental health service information system

> "The system would be required to support the maintenance of an action diary for each inspector, summarising his or her planned actions "
>
> "Required action details Property reference, EHO Code, Inspection/Visit ref , Action required, Planned revisit date "

A **required effect** associated with the problem domain description given earlier for the warehouse system

> "To record the movement of the pallet with its associated serials to a storage location in the racks, and to inform the Inventory system of this movement "

From the requirements for part of a telecommunications media gateway (the reference number at the end refers to the relevant **standard** which was being implemented)

> "The system shall declare an Alarm Indication Signal defect (AU-AIS) when all "1"s are detected in the H1 and H2 bytes for three consecutive frames EN 300 417-1/B (G 783/C 1)"

From sample configuration requirements for an ERP system

> "Travel Expense Processing Ability to facilitate travel processing and reimbursement Ability to assign varying levels of travel privileges Ability to handle direct deposit and cheque payment methods Ability to produce employee travel expense statements Ability to process travel reimbursements (weekly minimum) via the Payroll module or Finance module "

Required effects are so called because they express results or consequences of system activity that are observable by people or systems in the problem domain Kovitz (1999) restricts his use of the term "effects" to effects on the problem domain, but I am using the cover term **required effects** to include effects in the system itself that can be observed in the problem domain The practitioners' term for statements of **required effects** was typically the **functional requirements**, but this term is problematic, because it means so many different things Also, the **required effects** can include statements of so-called **non-functional requirements** such as

> "The system shall be fully validated to SIR level, including at least one long duration test (7 days) of a traffic bearing system without loss of service during that period "

Statements of **non-functional requirements** were not as extensively used in practice as might be expected given the emphasis that they receive in the literature The requirements documents that I examined contained very few such statements The few **non-functional requirements** that did occur in these documents were expressed at a very high level of detail In general, it may be said that, unlike other **required effects**, statements of **non-functional requirements** are not allocatable to particular **functions** or modules of the system More specific **functions** dealing with authorization, backup and recovery mechanisms tended to be used instead of high-level general attributes, making these essentially statements of required functionality or **functional requirements**

The textbook distinction between **functional requirements** and **non-functional requirements** did not seem to be reflected in either the practitioners' accounts or their documents In considering requirements statements, therefore, the use of different **levels of detail** in expressıng requırements ıs a more noteworthy dıstınctıon than whether the requırements themselves are functıonal or non-functıonal The documents were often organısed by **level of detaıl** **Requıred effects** would typıcally be stated ınıtıally at a hıgh level of detaıl, and then broken down ınto more detaıl Statements would be collected ın groups, or organısed around a **feature** A **feature** ıs often a very hıgh level requırement whıch ıs broken down ınto several requırements statements that are more detaıled Features would sometımes, but not always, be collected ınto so-called **bundles**

Dıfferent kınds of applıcatıons can have a mıxture of these types of requırements, but the partıcular mıx of categorıes wıll vary ın dıfferent types of applıcatıons

Transactıons are the maın types of **requıred effects** found ın busıness or organısatıonal ınformatıon systems Although ıt ıs possıble to dıstınguısh dıfferent levels, such as **busıness transactıons** and **database transactıons**, the concept of a **transactıon** ıs well defıned and understood ın the area of database systems, and ıs often used as a notıon of what a requırement ıs ın organısatıonal systems

> "You've got to have gone ınto what happens when a loan goes bad, exceptıons, and how you're goıng to handle that, what procedures are needed to handle that Most of the systems that I've come across have been transactıon processıng systems so ın that context you have to have a full set of transactıons "

Transactıons are usually assocıated wıth busıness applıcatıon systems, and sınce many of my ınformants were workıng ın that area, there were many examples of transactıon requırements ın the ıntervıew data Several of the other ınformants were workıng ın the area of telecommunıcatıons, but these cases were more concerned wıth network admınıstratıon, than wıth swıtches, for example Some were dealıng wıth ıntellıgent networks, whıch are also based on **transactıons**

Informatıon requırements were common ın database-orıented systems, especıally busıness systems These specıfy the types of **querıes** and searches that users want to perform, and **reports** that they need The practıtıoners' documents generally dıd not specıfy all the dıfferent querıes and reports that mıght be needed but ınstead were more lıkely to stıpulate that the system be able to support common types of ınformatıon retrıeval based on a specıfıc **database contents** and/or **structure** But strıctly speakıng, these are not "requırements" but **solutıons** and are classıfıed under another headıng reflectıng that dıstınctıon Typıcally, the practıtıoners saw no problem ın specıfyıng database contents as requırements

Behavıoural requırements was the term that practıtıoners workıng wıth embedded systems used for theır requırements, and these are sımılar to **Transactıon requırements** ın a way They also deal wıth events that happen ın the problem domaın

> "So a large part of the software requırements fall wıthın the applıcatıon domaın ın that they wıll specıfy features lıke, say ın thıs player you had tray to tray double speed copy, so the applıcatıon layer must manage all that "

Transactıons create, amend, and delete stored data ın the system ın response to **busıness events**, such as a loan beıng repaıd, whereas **Behavıoural requırements** ınstead respond dırectly to events ın the problem domaın wıth dırect effects ın that domaın, such as swıtchıng a motor on/off, etc These requırements typıcally were broken down ınto transıtıons between states ın response to events orıgınatıng ın the devıce beıng controlled or ın the user's control panel As well as the normal transıtıons (ıntended behavıour) of the system, the requırements documents dealt wıth error cases (behavıour the system ıs meant to prevent) and what one ınformant called **corner cases** unusual combınatıons of **events** and **states**, and the system's requıred **responses** ın these cases

Transformatıon requırements occurred ın only a few of the applıcatıons dıscussed by the ıntervıewees, and these appeared to be much less common than the other types of requırements The cover term ıs borrowed from Jackson (1995) Examples of these applıcatıons ıncluded the preparatıon and formattıng of payroll data and the generatıon of management ınformatıon statıstıcs from factory productıon data

4 7 3 Proposed solutıons

> "Thıs sectıon tells you a lot about how solutıons were arrıved at "
>
> "We are much more ın the busıness of, they tell us what the problem ıs and we suggest the solutıon, we mıght try to clarıfy the problem, and then ımplement a solutıon

Along wıth **Problem Domaın Descrıptıons** and **Requıred Effects**, the basıc contents of the practıtıoners' requırements documents consısted of another category of statements that are best descrıbed as **Proposed Solutıons** In theory, a pure requırements document would not contaın any solutıons, but, ın practıce, a pure requırements document seems to be very rare The extent to whıch solutıons are consıdered approprıate content ın the requırements document varıes consıderably from one sıtuatıon to another, and wıll be addressed ın a later chapter In thıs sectıon, the major types of solutıons that were consıdered approprıate to requırements documentatıon wıll be looked at Several of the practıtıoners stated that whıle detaıled solutıons were usually consıgned to some other document, such as a desıgn document, broad outlıne solutıons seemed to belong quıte naturally ın the requırements document

As far as the practitioners I interviewed were concerned, the requirements process seemed to be as much concerned with finding solutions as with stating the problems Many of them described themselves as **problem solvers**, and some of them called themselves **solution providers**, regardless of whether they had a ready-made solution to apply to a problem

Proposed solutions in the practitioners' documents were sometimes expressed at the lower levels of detail, in terms of the inputs and outputs associated with a **function** These were often associated with particular **required effects** Here is the solution proposed for the example given earlier of a **required effect** for a warehouse system

> "RF Screen to scan the pallet bar-code label, scan location bar-code label, ensure the pallet is known to be awaiting put-away, ensure the location is a valid warehouse location (reference LMmaster)"

However, **solution** was not just associated with the detailed requirements It turned out to be one of the most saturated codes in the open coding analysis of the interviews I carried out There seem to be several reasons for this First, I found that it is not enough for a requirements document merely to state the problem, it must also provide enough of an analysis of the problem to ensure or demonstrate that a **computer-based solution** can be implemented In practice, this often entails proposing an outline of the solution The broad solution may already be decided, as is the case with ERP systems The focus is on the solution This is illustrated by the following practitioner who configures ERP solutions for client companies

> "We help them come up with the solution, one that is achievable using the product that they have chosen "

Secondly, unlike the textbook approach, in practice, there is no clear distinction between problems and solutions where requirements are concerned, and this relationship was sometimes reflected in the way requirements were elicited Solutions to problems at one level become problems looking for solutions at the next level in this description of a process of Joint Application Development requirements workshop

> "Now at the next level then, we get them to write down four problems it's going to solve "

A business solution at one level becomes a problem statement in search of a technical solution at the next level

> "So, we propose a solution in technical terms, so we don't really propose the business solution We propose a technical solution to a business problem "

One of the ways that high-level solutions were incorporated into the practitioners' requirements documents has already been referred to, and that was the centrality of the

Database Structure and **Contents** in many of the business application systems Database definitions are a good example of high-level **solutions**

> "What I would have cared about all through that is more that the database was correctly designed so that we could produce any requirement that she came up with and that heart of the system survived, that we didn't find ourselves ripping it all apart and doing it all again "

Other similar sources of requirements included the influence of **System Architecture**, the use of **Prototypes**, and the impact of **Feasibility analysis**

Although **system architecture** is really design in textbook terms, for various reasons, in practice, it often precedes and influences the definition of requirements in different kinds of situations In the warehouse cases, the core **system architecture** existed already, so the solutions to many of the requirements could be described at that level

In general, it may be argued that any product or solution which is to be based on a database management system already has a predefined **system architecture** which shapes both the requirements and the solution In addition, once the relational data model is chosen, as it often is, the solution task becomes much clearer, especially when the contents of the database tables are defined Clearly, **database definitions** are not **problem domain descriptions**, nor are they specifically **required effects** The **database definition** in such cases is in fact the architectural framework of the **solution**, and regularly appears alongside the requirements for systems, in cases where a database solution is envisaged

Another reason why **proposed solutions** are essentially requirements has to do with the use of **prototypes** in determining what the requirements really are A prototype is a solution, not the solution, but a working model that demonstrates for the developers that the problem has some solution, and for the end users what the essential nature of the solution might be **Feasibility Analysis** used to be a prelude to requirements analysis in the systems life cycle, but it seems to have been subsumed under prototyping Prototyping has a number of benefits that do not replace the use of requirements documents and one of these is to demonstrate the feasibility of a particular idea, to show that there is a potential solution There is no point in stating something as a requirement if there is a doubt that a solution exists for it And no professional practitioner would give an undertaking to solve a problem unless it was stated in such a way that a solution was, if not obvious, at least reachable

The term **problem** was mentioned frequently in the interviews, but in the sense of problems, for example "a problem with this " and not at all in the sense of 'the problem' to which the system is a solution For some practitioners, 'the problem' seemed to be the same as the problem domain, because it was documented in the problem domain description For others, 'the problem' seemed to be inherent in the objectives of the system In either case, although **problem** was a category in the analysis generally, it did not feature significantly in the analysis of requirements as texts Instead, the cover term **objectives** appeared to be much more noteworthy as a category of requirements

4 7 4 Objectives

Objectives are very high-level requirements They are the main reasons why the system is needed or what the product is needed for They are often regarded as the criteria for success or failure of a project

> "You see from our point of view in providing a solution, if the client isn't clear on what he wants to achieve, it is dangerous for us, because it is difficult to know how we end up with a happy client "

For some practitioners, the **objectives** were the most crucial element of the requirements document, even if written requirements were not important to them

> "I would have one thing I would be quite keen on which is to put the word objectives up near the front somewhere and try and state what the objectives of the system are "

Objectives were significant to these practitioners because they guided the requirements process, and helped to define the scope of the system, so many of them considered it important to define them at the start, before any other requirements **Objectives** were often associated with the achieving the perceived **market benefits** of a product, or with the **business case** for the system

> " so the business people go off and they try to establish what the business case is, what the marketing specification is, how the scheme might work "

However, not all of the practitioners thought that it was necessary for **objectives** to be written down, as long as they were known and shared

> " an essential part of getting a clear definition of requirements is having a clear vision of what it is you are trying to do "

> " but the problem is that sometimes it is impossible to get it written down The result will not necessarily be what you want it to be or do what you want it to do which is establish the shared vision "

In an organisational context, objectives are often called **goals** Organisational goals have been defined as "conceptions of desired ends," conditions that participants try to effect through their performance of organisational activities (Scott, 1987, pg 18)

> 'In recent times we are only supposed to be addressing projects that actually meet the aims of the Department, the critical success factors of the corporation "

System objectives were frequently seen as ways to help achieve organisational goals, by saving money or by increasing business, for example, but they were also associated with **non-functional requirements** such as

> "They want something that is more stable, more long-term and more scalable, so that it doesn't matter whether it increases to twice the size, three times the size, four times the size, it can still handle the traffic "

4 7 5 Constraints

Unlike **objectives**, which are about desirable but not necessary conditions, **constraints** are fixed conditions which must be satisfied by the system or software product Unlike ordinary requirements, they are not negotiable

An important criterion I used in identifying constraints in a requirements document or situation was whether they were global or not Only global constraints, those that affected the system as a whole, were coded as constraints I found that other types of constraints were typically embedded in the other types of requirements statements, for example, business rules in the problem domain descriptions, and cardinality constraints in the database models These were not given any special treatment by the practitioners, who singled out **behavioural constraints** and **design constraints** These are similar in that they both restrict the range of feasible **solutions**, but design constraints occurred generally, and behavioural constraints only in certain situations

Some of the practitioners were working in the field of embedded software, used in consumer electronics and other types of special computerised equipment This type of software was regarded by those practitioners as best described in terms of its behaviour, categorised as **behavioural requirements** earlier In defining the requirements, they stressed the importance of enumerating all the different ways the software could interact with its environment

> "It's a behaviour model that tries to capture all the user-perceived states of the system "

> "So we try to describe the behaviour of the player - what it will do under all kinds of scenarios"

They reasoned that it helps this task considerably if, first of all, it can be determined how that interaction or behaviour is constrained This saves the trouble of having to consider several situations that are prohibited from occurring in any case This was done using **behavioural constraints**

> "You've got to do the constraints table first, in order to make the whole thing hang together "

> "So we have a constraints section – which was the bit you were looking at originally - which looks at each variable and looks at

each of its particular states and then see what constraints it puts on the rest of the system "

Design constraints are more general in their scope and application These were aspects of the **solution** that were, for various reasons, determined **in** advance They were associated with a range of different types of requirements situations in the interview data Examples included such things as the hardware/software platform on which the solution was to be implemented, the memory limitations of the platform on which it would run, or the data input methods which might be used, such as RF (radio frequency) terminals, or the particular set of buttons on the user interface of a device

"Any new piece of software, had to meet certain design rules in relation to not overloading the process, or not causing too many switches and connections "

General **design constraints** were mostly mentioned by practitioners working in large software development organisations Those working in market-oriented situations saw the existing **system architecture** of the product as a major source of **design constraints** The next version of a software product is in many ways constrained by the current and previous versions of it This point was mentioned in relation to the need to maintain user interface consistency or to provide backward compatibility in the new version It was also pointed out that intermediate **releases** of software, as distinct from major new **versions** of products, tended to address only parts of the functionality of the entire product, and therefore their requirements were strictly constrained by the overall **architecture** of the product

4 7 6 Issues

Issues are really questions concerning requirements, which are raised by the participants in the course of meetings, discussions or reviews of a requirements document

"It was a statement of what all the different issues were Would we want to know what the part number of the vendor was, things like that, would we want to do, for example, to do weighted average pricing would we want to do straight list pricing, things like that, costing of the inventory that would be rather than pricing, that kind of stuff would come up "

Many of the practitioners interviewed emphasised the practice of recording the **issues** and related matters in the evolving requirements document as an integral part of the process of gathering or discovering the requirements per se Many of them described a process whereby **issues** would arise and be documented first as **open issues** before being resolved and documented as **closed issues** along with the relevant **solutions** or **decisions** that were made to resolve them

"So we'd list those issues and then we d resolve those issues one by one and in the document there would be a list of the issues and what decisions were taken and why they were taken "

In that case, **ıssues** were documented ın a separate document, but typıcally, they were gathered ın a partıcular sectıon of a document, or ın some cases, a sectıon wıthın each **transactıon** that they referred to

Accordıng to several of the practıtıoners, ıt was not enough for a requırements document to merely define the agreed lıst of requırements They saıd ıt must also support and reflect the process by whıch thıs agreement comes about Thıs ıs the sıgnıficance of the role of the requırements document as a **workıng document**

> "Thıs ıs very much relevant to the process of arrıvıng at an agreed first draft document, all the ıssues are ın there, many of them are not even dealt wıth yet, obvıously "

The **draft** requırements document ıs the central focus for dıscussıons and revıew meetıngs, and each **draft** ıs ın turn revısed as a result of those meetıngs **Issues** are typıcally raısed as ıtems for dıscussıon, and **changes** are suggested and agreed Both the resolved **closed ıssues** and the unresolved **open ıssues** are recorded ın the next **versıon of the document**, along wıth the relevant **changes** whıch have sınce been ıncorporated ınto the related requırements statements Thıs contınues ın a cyclıcal process for a number **of drafts** and **versıons** untıl some kınd of agreement ıs reached In some cases, practıtıoners worked wıthın a defined process whereby ıntermedıate **drafts** were ınterspersed wıth more definıtıve **versıons** whıch had a dıfferent status

> "That versıon was revıewed, there were some small changes made and a new versıon was produced And that's the way ıt was done "

Issues can arıse for dıfferent reasons Some of the reasons mentıoned by the practıtıoners ıncluded lack of knowledge, lack of understandıng, and lack of agreement These would occur at the start of the cycle, and would be gradually ıdentıfied, documented and resolved ın varıous ways Lack of knowledge on the part of dıfferent stakeholders would typıcally be resolved by the contrıbutıons of other stakeholders, ın the course of wrıtıng and revıewıng the document

> "So basıcally thıs ıs to try and sort of clarıfy a lot of thıngs between us and them and also for people workıng on the team "

> "We revıewed that document and then had a number of faxes and communıcatıons wıth them and also face to face meetıngs wıth them to dıscuss and clarıfy the requırements "

Dıfferent stakeholders ın a project brıng dıfferent perspectıves and background knowledge to ıt The developers, as well as the end users, can have theır own **ıssues** or **questıons** that need to be resolved

> then I tend to stıck ın an area for notes and stuff lıke that where you start puttıng normal questıons you don't know the answers to, notes and querıes, thıngs lıke that "

Lack of understanding between the stakeholders can often be resolved by a process of **clarification** The act of writing things down or re-writing a previous statement is often seen as a strategy for **clarification**

> "One thing about writing is you have to write it and then somebody has to review it and then you have to issue it and somebody has to, sort of, say 'well I don't agree with this' "

> "This section tells you a lot about the exceptions and it tells you a lot about how solutions were arrived at, and part of its purpose is when people come back with questions, you can say it was raised in here, this was what we decided, perhaps even why we decided it "

4 7 7 Document information

The final category of information that came out of my taxonomic analysis of 'requirements as texts' comprises the different kinds of information that practitioners typically recorded in their requirements documents about the documents themselves These included various items such as the **Document history**, its **Circulation list**, information about the **Authorship** of the document, as well as the **Table of Contents**, the **List of reference documents**, and related information

The **Document history** would usually be a table used to indicate the various **versions** of it that had been issued, along with the relevant dates when it was reviewed, and in some cases, who reviewed the document In some cases, this information would appear alternatively as a simple **Circulation list** of stakeholders involved in the project, who might have different roles in relation to reading, reviewing and agreeing to the contents

In many cases, while going through a sequence of **drafts** and **versions**, requirements documents also went through **different stages of agreement** For some practitioners any document (or version of it) being discussed would have a definite status in terms of whether it was an early draft, a version for review, or a near final document

> "OK, there was a major review on the 3rd of December Then, the following draft, there was some minor updates, some of them based on the original review Then, at that point, the document was at proposal stage It got moved to accepted, all the updates were done And then, since then, it's just changed because of change requests Or mistakes we found in it, or inconsistencies "

One of the many purposes of a requirements document is as a **record of agreement** between the stakeholders in a development project, but as already indicated, many of the practitioners felt that a more important purpose was the role of the document in **getting to agreement** This was particularly the case in some organisational contexts For example, in situations where a customer organisation employed a supplier organisation to work on a solution, the agreement between them was a formal **contract** But in other situations, the **idea of a contract** was ınvoked more as a mechanısm to focus attentıon on **agreement** than as a way to enforce complıance by one party or the other

> "An exercıse lıke that has to be done because you have to be able to say at the next stage 'well thıs ıs what you saıd you wanted' but you can't rely on ıt as beıng the fınal say so ıt wasn't a contract Well ıt was, ın the sense that they sıgned ıt off and we could have treated ıt as a contract but ıf a manager comes back at the desıgn stage and says 'you can't do thıs,' ıt's no use goıng back and sayıng 'we have a contract '"

Essentıally, ın many sıtuatıons, **agreement** was seen as ınstrumental before movıng on to the next step ın a project, or effort would be wasted on unwanted or ınapproprıate development It was often consıdered more ımportant to achıeve such agreement ın the mınds of the people concerned than to make sure ıt was embodıed ın the document

> "The purpose of thıs document ıs to document our agreement but at the end of the day ıt's not enough for ıt just to be ın the document "

Some ıntervıewees strongly expressed the vıew that the agreement was not *ın* the document, but exısted as a **shared vısıon** ın the mınds of the people ınvolved ın the project

> "If you do a workshop and everybody gets a shared vısıon from that workshop you can then wrıte that But what ıs key ıs that everyone understands what that vısıon ıs "

The ıdeal concept of agreement depends on understandıng and explıcıtness In practıce, thıngs are never really fully understood, or are not always explıcıtly stated, so the pragmatıc nature of agreement was recognısed by many of the practıtıoners As far as they were concerned, a certaın **level of agreement** needed to be achıeved ın order to progress wıth the work of development How much **agreement** was adequate seemed to vary from one practıcal sıtuatıon to another

The followıng three **levels of agreement** may be dıstınguıshed ın the dıfferent types of requırements documents and sıtuatıons that I ınvestıgated

1 **Contract** Agreement has the force of a contract, or a formal agreement between two legal entıtıes, and has a legal sıgnıficance
2 **Agreed** Agreement has the effect of an *ınformal* agreement between two partıes wıthın an organısatıon, and has no legal sıgnıficance
3 **Approved** Agreement has the sıgnıficance of endorsement by the hıerarchy of an organısatıon, represented by the revıewers of a document Agaın, thıs has no legal sıgnıficance

4 8 Conclusıons a dıfferent classıfıcatıon of requırements

In thıs chapter, I have presented the results of a taxonomıc analysıs of the domaın of 'requırements as texts' based on the practıtıoners' accounts of theır own practıces and a selection of documents I collected in the course of interviewing them This analysis corresponds to the second level of grounded theory analysis, axial coding A number of different themes related to requirements as texts emerged from the analysis, the most significant of which were the variety of different types of documents used in practice, the range of different roles and purposes of requirements documents, and the variety of requirements statements contained in the documents The following general points summarise the findings

The concept of a requirements document embraces a broad range of different types of documents This is reflected in the different names that they were called by, in the different work environments The main requirements document is often supplemented by a range of supporting documents, related to different practices and modes of use

The requirements document has a wide variety of uses Two of its most important roles relate to **agreement** The role of the document in **getting to agreement** was often seen as more important than its role in **recording the agreement**

A number of categories of requirements statements were identified in the analysis
There were seven mam categories The first three of these, **Problem domain descriptions**, **Required effects** and **Proposed solutions** form the greater part of the requirements document **Required effects** including different types of **Features**, **Transactions**, **Information requirements**, and **Behavioural requirements**, often but not always depend on there being sufficiently comprehensive **Problem domain descriptions** to make the interpretation of these statements intelligible and explicit **Proposed solutions** cover categories such as **Database structure** and **System Architecture**

Two other main categories, **Objectives**, and **Constraints**, are similar in that they supplement the meaning of the first three **Objectives** state very high-level requirements which can generally be translated into more detailed requirements but some remain as high-level **objectives** that are not transformed in this way **Non-functional requirements** are often stated in this way **Constraints** are of two types Those that refer to the problem domain help to clarify the relevant descriptions, and those that constrain the design limit the variety of solutions that may be included in the **proposed solutions**

The category of **Issues** is related to the roles of the document as a **discussion document** and as means of **getting to agreement** Typically, these arise during discussions and meetings as questions about the requirements statements in the current **draft** or **version** of the document

The recordıng of **Issues** and subsequent **Changes** to requırements ıs the means by whıch the process of reachıng agreement ıs supported by the use of the requırements document

Another ımportant purpose of wrıtten requırements ıs ın **recordıng agreement** What ıs agreed, and whether thıs agreement refers to **requıred effects, objectıves,** or **proposed solutıons**, or possıbly also **problem domaın descrıptıons** and **constraınts,** ıs subject to a varıety of practıces approprıate to dıfferent cırcumstances It almost certaınly refers to how **ıssues** were resolved due to relevant changes made to any of these How much documentatıon ıs kept on these changes varıes, though they may be traced back through dıfferent versıons of the document, thıs ıs not commonly done

The partıes ınvolved, and the level of wrıtten agreement that ıs consıdered necessary and suffıcıent, also varıes ın dıfferent sıtuatıons These are ıncorporated ın the fınal maın category, **Document ınformatıon** Thıs covers such obvıous categorıes as the **Table of Contents** and the lıst of related documents, but also the **Lıst of partıcıpants** whıch has consıderable ımportance for the role of the document as a **record of agreement** ın that ıt records the names of the partıes to that agreement Another outcome of thıs analysıs was the ıdentıficatıon of three dıfferent levels of agreement that apply ın dıfferent sıtuatıons **Contract, Agreed** and **Approved** As a result, the new category **Recorded Agreement** replaces the orıgınal maın category of **Document Informatıon** ın the followıng proposed taxonomy of requırements statements Thıs ıs essentıally a dıfferent way of classıfyıng requırements

1 **Problem Domaın Descrıptıons**, exemplıfied by Busıness Rules, Termınology, etc
2 **Statements of the Requıred Effects,** e g Features, Transactıons, Informatıon requırements and Behavıoural requırements
3 **Proposed Solutıons,** such as Database structure and Contents, System Archıtecture, and Prototypes
4 **Recorded Issues and Changes,** ıncludıng both open and closed ıssues and related changes to other categorıes of requırements
5 **Defıned Goals and Objectıves,** exemplıfied by busıness cases and market benefits, but also ıncludıng non-functıonal requırements
6 **Specıfıed Constraınts**, such as behavıoural constraınts and desıgn constraınts, and
7 **Recorded Agreement,** embodıed ın authorshıp and cırculatıon lısts, and document status and versıon ınformatıon

In Chapter 7, these seven dιfferent types of requιrements as texts wιll be brought ιnto play as part of an explanatory framework to assocιate dιfferent sιtuatιons wιth the elements of the taxonomy appropriate to them But first, ιn the next two chapters, a typology of requιrements sιtuatιons wιll be developed, focusιng on requιrements as needs

Chapter 5 An Ontological Analysis of Requirements as Needs

"Think Sideways "

Edward De Bono

5 1 Introduction

The previous chapter looked at 'requirements as texts,' the various types of statements that are used in requirements documents, using a taxonomic approach to analysis and resulting in a new way of classifying requirements In this chapter, another perspective, that of 'requirements as needs' will be explored, and this will be done using a different style of analysis The findings presented in this chapter will be based on the consideration of precedence relationships, rather than taxonomic relationships, between categories found in the empirical data

The previous analysis began with a collection (family) of codes called 'requirements as texts ' This consisted of a large subset of the entire collection of codes from the initial open coding, the first stage of grounded theory analysis Another, distinct set of categories, in this case, a family of codes called '**concerns**,' resulted also from the open coding stage These were substantive concepts ranging from **ad hoc reporting** to **workflow** My conjecture was that in any particular situation, some **concerns** would be more significant than others, and greater emphasis would be placed on those **concerns** Such variations would provide a way of differentiating between the situations, which was one of the major aims of this research

This set of **concerns** formed the basis for the analysis presented in this chapter Again, like the taxonomic analysis, this analysis corresponds to the second stage of the grounded theory approach, axial coding The result is a conceptual network showing how different kinds of requirements, considered as needs, are related to each other, and ultimately to their sources The network identifies eight root **concerns** as the main sources of requirements

5 1 1 Sources of Requirements

Where do system and software requirements come from? What are their sources? It became clear that the family of codes I called **concerns** contained all the important sources of requirements such as the **stakeholders**, as well as types of actual requirements such as **user interface requirements**, and intermediate concerns such as **changes to the existing system** Requirements exist because someone or something needs them, or because other requirements necessitate them An obvious example is the existence of **user interface**

requirements because of the existence of **users** **Interfaces to other systems** exist as a category of requirements because of the existence of a requirement for **the system boundary**, as well as the other systems Less obvious examples of such dependencies might be found in the practitioners' accounts of their own situations This was the rationale and the template I used in looking for relationships between the categories in the code family **concerns**

5 1 2 Ontological dependency

Some things depend for their existence on other things This is called ontological dependency (Stamper, 1994) At its most basic level, it can be expressed as a relationship between two entities, in which one entity precedes the other, the first being the ontological antecedent of the second An entity may have more than one immediate ontological antecedent, and/or more than one direct ontological successor Ontological dependency is a transitive, asymmetric relationship

Stamper (1994) used this relationship as the foundation of a systems analysis technique MEASUR (Method for Eliciting, Analysing, Specifying User Requirements) includes a step called Semantic Analysis which results in an Ontology Chart of the application domain The method is predicated on the distinction between substantive objects and semiological objects Ontology charts model the objects of interest in an application domain However, the relationships between these objects are not the usual 'isa' and 'is part of' relationships normally found in requirements models, but ontological dependency relationships

MEASUR uses ontology to distinguish between the substantive objects in a problem domain and the information we have about them The entity 'Person,' for example, is a substantive object while 'person-record' is a semiological object or, simply, a sign In semiotics, the study of signs, a sign is a unit of meaning which is interpreted as 'standing for' something other than itself Information systems function by processing signs such as names and numbers which represent substantive objects that exist in the problem domain MEASUR's ontology chart focuses attention on the substantive objects of the problem domain, leaving the semiological objects for a later stage of analysis This distinction is central to the MEASUR approach A similar distinction can be made in the domain of system requirements, between requirements as needs and requirements as texts, and is central to the analysis of findings presented in this thesis Having dealt with the semiological aspects of requirements in the taxonomic analysis of the previous chapter, we now focus on the substantive aspects, requirements as needs

The findıngs ın thıs chapter are presented as a conceptual network of ontologıcal precedence relatıonshıps among the requırements **concerns** revealed by the practıtioners ın the ınterviews, and ıdentıfied durıng the open codıng stage The network was constructed usıng the Network edıtor facılıty ın ATLAS tı

5 1 3 Usıng ATLAS tı to buıld the conceptual network

A Prımary Document (such as an ınterview transcrıpt) coded ın ATLAS tı can be vıewed ın a wındow sıde by sıde wıth another wındow whıch ıs a vırtual margın for that document The margın shows the codes that have been assocıated wıth the relevant sectıons of the prımary text shown ın the maın wındow The codes that are dısplayed ın the margın can be filtered ın dıfferent ways, to ınclude or exclude dıfferent kınds of codes One very useful filter ıs the 'code famıly' filter whıch allows the user to vıew only those codes belongıng to a selected famıly, such as **concerns** Thıs facılıty was very valuable ın browsıng through the dıfferent ınterviews focusıng on passages where the practıtioners had brought up poınts about requırements as needs or sources of requırements

Another facılıty provıded by ATLAS tı was the abılıty to output, for example, frequency tables, showıng the occurrence of each of the (filtered) codes ın each ınterview, wıth summarıes for each ınterview and summarıes for each code It ıs ımportant to note that code frequency ıs not very sıgnıficant ın grounded theory generally, but partıcularly ın thıs research where the accent was on varıety So, for example, the output table would typıcally contaın several codes that occurred ın some ınterviews and not ın others Thıs underlıned the varıatıon ın dıfferent sıtuatıons experıenced by the practıtioners

ATLAS tı also has tools for vısualısıng the relatıonshıps between codes or other elements of the project The Network edıtor allows the constructıon of varıous network vıews of the codes, memos, and the empırıcal data Drawıng a lınk ın the chart creates the relatıonshıp between the relevant codes, and ımportıng two related codes ınto a vıew automatıcally ıncludes theır relatıonshıp lınk I used thıs facılıty to buıld an ontology chart, usıng MEASUR's ontologıcal precedence relatıonshıp, showıng how all the **concerns** were thus related Although dıfferent **concerns** and consequently relatıonshıps were found ın dıfferent ınterviews, the ıdea was to look at the area of requırements as needs as a whole, to synthesıse them rather than to analyse the sıtuatıons Thıs would be done at a later stage

The Network edıtor has varıous types of relatıonshıp lınks buılt ın, such as the *ISA relatıonshıp*, the relatıonshıp *causes*, and the relatıonshıp *ıs-a-consequence-of* A 'relatıon edıtor' allows the user to create new relatıonshıp types, define theır appearance and menu text, and specify their properties, e g whether a relationship is symmetric, transitive, etc The newly created relationship type then appears in ATLAS ti's Link menu and can be used to link codes or other objects to each other I used this option to create a transitive relationship type called Precedes, corresponding to the ontological precedence relationship This relationship was defined as asymmetric, so its inverse represented the ontological succession relationship

The nodes of the network thus created were all belonging to the code family **concerns**, and this code family was gradually reduced, from over 130 codes, to seventy-two codes by a process of proposing and testing ontological relationships between them Each relationship was tested for goodness of fit, using supporting evidence within the primary texts Another powerful Atlas tool, the Query tool, helped with this task Unlike the Text Search tool which queries the data directly, this tool queries the codes, and through them the data It can be used to look for evidence in the data to support any proposed or actual link between the codes in the network, by searching for particular combinations of codes, using specific query operators

Three different sets of query operators are used in the Query tool boolean, semantic and proximity The boolean and proximity operators query the co-occurrence of codes in the data For example, boolean operators allow searches for occurrences of Code A AND Code B, but NOT Code C Proximity refers to the closeness in the text of the codes involved in the relationship being tested The semantic operators, SUB, UP and SIBlings, make use of the theory expressed in the (semantic) network

Unfortunately for my purposes, these have a built-in hierarchical bias The UP operator specifies a query relating a code to its parent in the semantic network This is very useful for querying the ISA relationship between a pair of codes The SIBlings operator queries the data for confirmation of two or more codes having the same parent The SUB operator finds confirmation for the link between a parent code and its child SUB also works transitively to find data related to all the descendants of a particular node in the network This facility was intended for use in ISA hierarchies, linking a node with its highest-level parent node as it occurs in the data The Precedes relationship is different, because it is not hierarchical It really works sideways, from left to right The precedence network is partially ordered It is not a tree, since it has a number of different "roots " These appear on the left of the network, since they obviously precede their successors

5 2 The requirements ontology

An extract is shown in Figure 1 to illustrate the structure of the Network The nodes all correspond to codes from the code family **concerns** Codes such as **priorities** and **core requirements** are marked with a tilde to denote that they have been previously merged with other similar codes denoting the same concept The arrows representing the relationships all proceed from left to right The intended interpretation is that the leftmost **concerns** come first and everything else comes after that **Project** is a **root concern** It has no ontological predecessors in the network

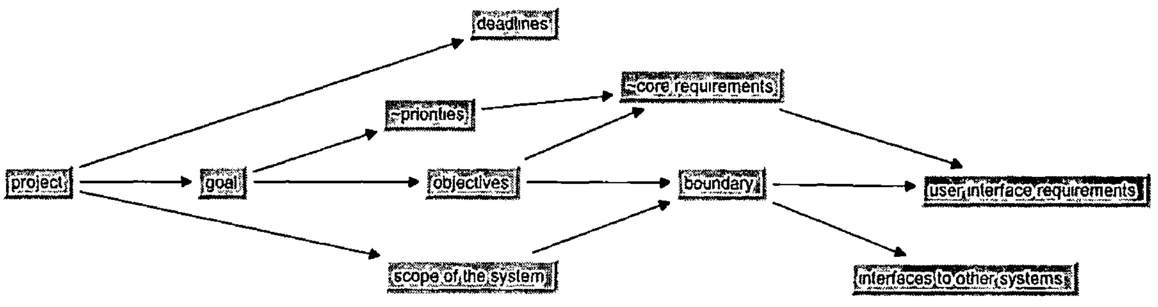


Figure 1 An extract from the Ontology Chart

These **concerns** will be familiar to anyone who knows about requirements engineering, and therefore this excerpt is intended to give an idea of how the ontology is structured

The **root concerns** are the most fundamental ones and are therefore all placed at the left-hand side of the chart The other ones are **Client Organisation, Users, Business Area, Application Domain, Customer, Project** and **Product** Each of them is linked by arrows to its ontological successors to the right The relationships are asymmetric and transitive Several of the non-root nodes have two or more antecedents The rightmost **concerns** (or leaves of the structure) are the most dependent These include such **concerns** as **Ownership, Shared Vision, Changes, Errors,** and **Unresolved Issues** The Network chart showing all seventy-three **concerns** is shown in full in Appendix G

A larger, more comprehensive ontology chart would contain additional concepts representing all the various aspects of requirements statements (**requirements as texts**) and requirements processes such as elicitation and validation This ontology deals only with the substantive **concerns** which are the ontological antecedents of those categories

5 3 An ontology of concerns

This section will discuss some extracts from the ontology chart and comment on them The excerpt shown in Figure 2 deals with typical concerns arising from the fact that requirements are located in the context of **projects** which are carried out in **organisations** None of the extracts is stand-alone, so the complete set of relationship links for any **concern** is not necessarily shown Also, in Figure 2 and in other extracts, the **concerns** shown (especially those on the right-hand side) have successors which are not shown there but will appear in other excerpts

5 3 1 Project and organisational concerns

The excerpt in Figure 2 shows some dependent concerns arising from three **root concerns, project, users** and **client organisation** Once a **project** comes into being, a number of other **concerns** arise, including the **goals** and **objectives** of that **project** It also leads to the coming into being of a defined set of **stakeholders**

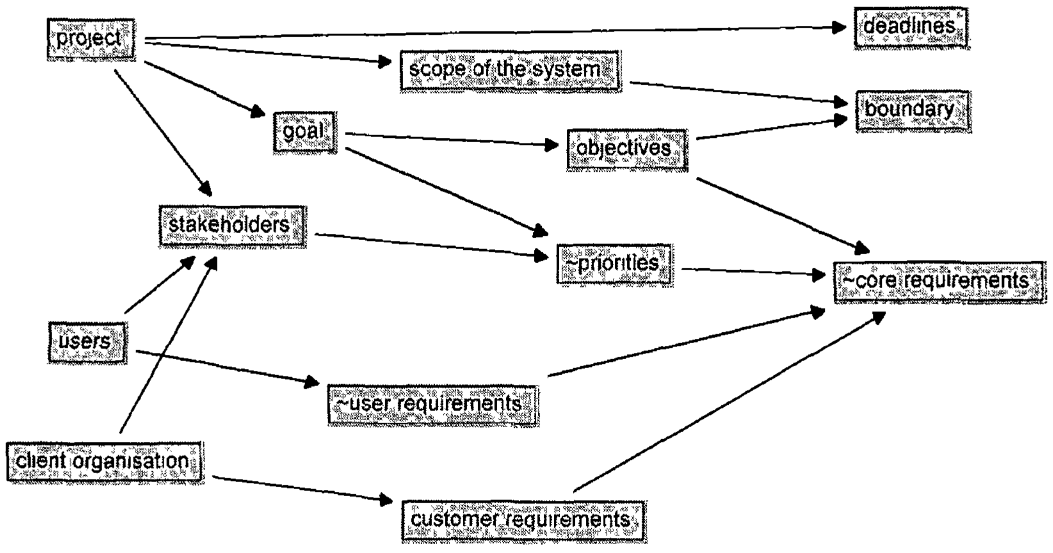


Figure 2 Project and Organisational Concerns

It was not possible from the interview transcripts to find out whether the participants made a distinction between **goals** and **objectives**, or if they used the terms interchangeably The word 'objectives' was used much more frequently, about three times as frequently, but the idea of a **project** having a **goal** was more clearly expressed Therefore, I chose it as a node to precede **objectives** in the network, but not as a root node

In the literature, some notable requirements techniques are concerned with **goals**, and some take goals as their starting point (Lamsweerde et al , 1998) As Colin Potts has pointed out, this raises some important questions, like "Whose goals are we talking about?" (Potts, 1997) This supports my finding that **goals** are not a **root concern**, they arise from different sources the **project** itself, the different **stakeholders**, or the **client organisation**

*The term **stakeholders** is a conceptual code used to categorise the many instances when* practitioners discussed the phenomenon of the various people involved in projects, but not a term ın common use by the partıcıpants ın the study Stakeholder ıs a role taken ın relatıon to a partıcular project Its exıstence depends on the exıstence not only of the project, but also of a set of users assocıated wıth the project

The term **users** ıs problematıc, ontologıcally speakıng It ıs also a role However, users, as people, exıst ındependently, despıte the unfortunate ımplıcatıon that they only exıst ın relatıon to systems they use Some other term would be preferable, but the term 'users' ıs well establıshed ın the field of system development and requırements engıneerıng, so an alternatıve term would not be approprıate

The **core requırements** node ıs a central concern whıch arıses from the combıned effects of the prıorıtıes of the stakeholders, the exıstence of a set of user requırements, and the goal and objectıves of the project Note that we are talkıng about these as concerns and not as textual components of requırements documents, although many of these codes would have that ınterpretatıon also **Core requırements** ıs a category used here to cover the dıfferent ways thıs ıdea was expressed by the practıtıoners, some of whom used other terms such as 'basıc requırements,' 'mınımum requırements,' etc Some of them used the term 'system requırements' ın thıs sense but theır use of thıs term ın a non-techmcal sense conflıcted wıth dıfferent usage by other practıtıoners, (see quotatıon) so that ıt was decıded not to ınclude that term ın the ontology

> " the system requırements had been defıned by the underlyıng archıtecture that had been defıned for the prevıous phase of the project
>
> They were standard and they weren't up for negotıatıon The customers had actually seen the system requırements and were happy wıth them So thıs was purely for user requırements "

The concept of 'Customer' ıs a fundamental one ın requırements engıneerıng Practıtıoners ın the study frequently referred to 'the customer' but agaın, a wıde varıety of ınterpretatıons were assocıated wıth thıs term In some sıtuatıons, the customer was a member of the same organısatıon, and synonymous wıth 'user,' but more often the term referred to a separate organısatıon, partıcularly ın cases where there was a formal contract, ın whıch case the term 'clıent' was synonymous wıth customer

To try to dısambıguate these terms I have establıshed the followıng three categorıes presented here wıth my workıng defınıtıons, based on the varıety of sıtuatıons that I came across ın the ıntervıews

Client organisation, because the client is nearly always an organisation, maybe an external organisation, or alternatively, another part of the parent organisation to which the development organisation/team belongs There is always a client organisation involved in any significant software development It may be represented by a person called the Sponsor

User, because the user is an individual, who may or may not be directly involved in the requirements process The user is a person who eventually will use the system Some systems have thousands of users

Customer, because the software has to be paid for The customer may be an individual or an organisation There may be thousands of customers, or just one In some situations, the client organisation is the customer In many other situations, the customer is separate, and may not be engaged in the requirements process

The practitioners who were involved in developing embedded software for consumer-oriented electronic products typically had Product Managers who represented the (potential) customer Some of them were developing new versions of products in response to problems reported by customers who had bought previous versions of the product In the ontology, the concept **customer** is placed in relation to packaged software, or **Product**, (see Figure 5) while the concept **customer requirements** is positioned in relation to the **client organisation** This is because the real customer may not yet exist in some situations, though some organisational entity will be appointed to represent the customers' role in the project

The concepts shown in Figure 2 were frequently mentioned by the practitioners concerned with systems which were intended for use in organisations Some of their organisations were involved in non-profit activities such as government revenue collection or health administration These practitioners reported many of the same concerns as their more commercial counterparts Some of their additional concerns are shown in Figure 3, another excerpt from the network, showing some related **concerns** related to the domain of commercial business organisations

5 3 2 Business Organisations

Traditionally, requirements analysis began with an analysis of the **existing system** Many practitioners talked about requirements arising from the **existing system** but it was not the ultimate source of these requirements The **client organisation**, whether in-house or of an external customer is a predecessor source

The concept **organisational change** is shown in the diagram as an antecedent of **issues** This refers variously to the need, or the suggestion, or the possibility of such change, and was frequently mentıoned ın relatıon to **ıssues** as dıscussed by the practıtıoners who work at developıng or configurıng systems ın organısatıonal settıngs For example

"There was a bıg shake up ın the structure of the management There were some, old managers who had worked to old methods and saw the new system as a threat and there were some new guys who were very keen on changıng thıngs and also on puttıng theır mark on what was happenıng

"That wasn't happenıng ın ısolatıon, the whole company was changıng Some of the management had been there for years and years "

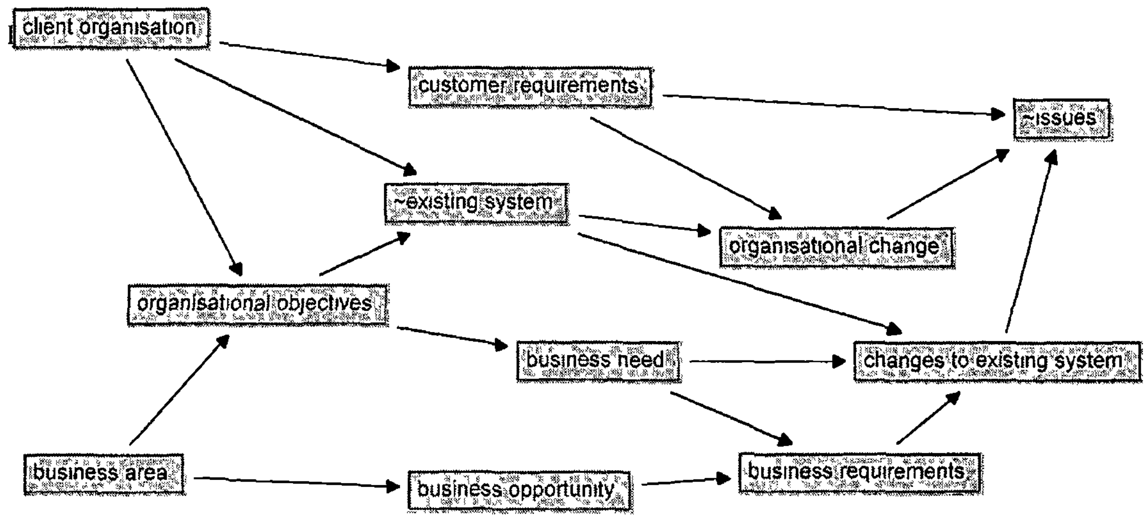


Fıgure 3 Busıness Organısatıons

Fıgure 3 shows another startıng place for requırements, the **busıness area** Thıs often gıves rıse to general organısatıonal objectıves, some of whıch are ın turn embodıed ın the **exıstıng system**, as well as the specıfic **organısatıonal objectıves** of the **clıent organısatıon** A frequent source of new requırements ın a busıness settıng ıs the need for the organısatıon to respond to new challenges or opportunıtıes Some very experıenced developers I ınterviewed seemed to be quıte concerned about the mısmatches they perceıved between organısatıonal objectıves and trends ın the busıness area they worked ın Such dıfferences wıll result ın the need for changes to organısatıonal objectıves as well as changes to systems They saw themselves as 'agents of change,' responsıble for organısatıonal needs, rather than 'order takers' merely respondıng to the needs of the ındıvıdual users For many system developers these sources of requırements were more ımportant than any ındıvıdual user or manager

"I see IS as a servıce not to other departments or to other people but to the company and ın doıng that they (we) need to be ırreverent about what happens ın other departments

What ın fact we've become ıs order takers People wıll come to us and say I want a report that does thıs, I want to find a place to

> record this piece of data that we didn't record before And we go and we find that place to keep it "

Business information systems are associated with specific functional areas which arise out of a business area This may also be the source of a **business opportunity**, a concept which was often mentioned in relation to specific **business requirements** More generally, frequent mention was made of **business knowledge** on the part of end-users or analysts as a source of requirements For example

> "So what we try to do here and I suppose it's what I've always tried to do is first of all understand what we, as a business, want to do Forget the system, forget the jargon "

5 3 3 Business processes

Since several of my informants were working with tailorable packages for Enterprise Requirements Planning (ERP) systems, frequent mention was made of **Business Processes** These are now seen as a central concerns in many organisational systems, even outside of the domain of commercial business As one of the informants stated "Business processes drive the need for computer systems "

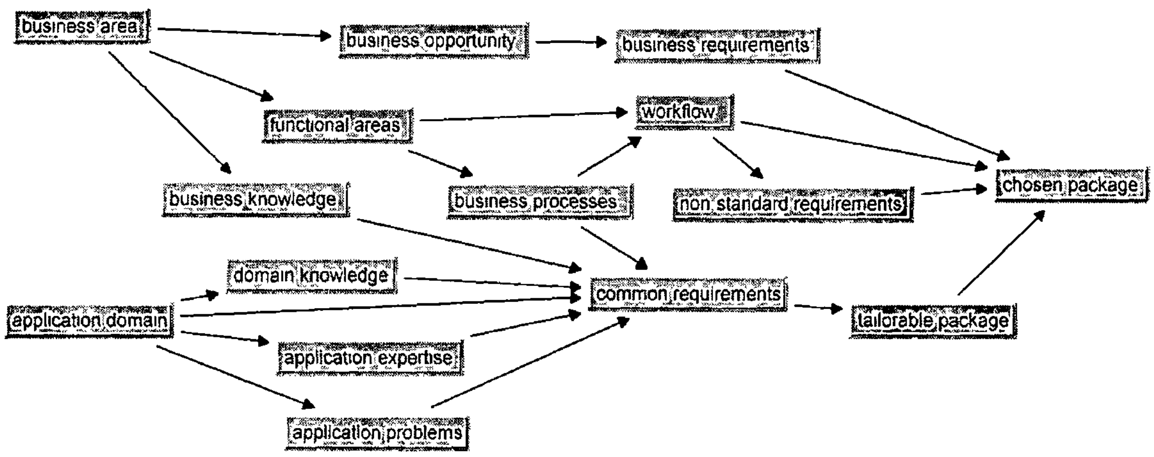


Figure 4 Business Processes

The decision to acquire a particular ERP package is often made as a business decision, for example to gain competitive advantage, rather than as a result of a detailed requirements analysis process

"First of all it was decided that it was going to be SAP, then we began the requirements analysis" is a comment that represents the typical approach to acquiring packages such as this

> "Yes, the quick way is to look at reference sites, that are similar to your business, and decide that way That's the best way
>
> 'There is a question of course, which has been asked, which is How do I get an edge if all my competitors are using SAP?"

Requirements analysis in the context of ERP packages is a lengthy process that involves documenting hundreds of business processes Note that the **business processes** concept that was shown in **Figure** 3 refers to the existence of these processes in a **business area** Advance knowledge of these processes is put to good use by the software vendor in making a package adaptable to the needs of many different businesses

> "We would produce a business blueprint which describes the functionality that we will configure for the client it is very clear as to what the deliverable will be in terms of how the software will function, a fairly detailed list of functionality in terms of business processes "

5 3 4 Application domain

The concept of the **application domain** is another fundamental concern of practitioners who work with requirements Knowledge of a specific application domain or **applications expertise** as it is called in the ontology was considered indispensable to requirements engineering, in most situations, whether in-house or contractual situations and in most application areas, particularly where technical advances were being made all the time A good example of the latter was the area of telecommunications

> "One of the problems with customers, as well, is because they are in a fast moving, ever changing environment, some of them will say 'these products are good but no one has really used them successfully' or 'no one, that they know of, has bought them and used them', so they don't want to be the first "

> "This was the protocol overview, the overview of how this works, the introduction really to this whole area I mean you can see there, it's about six or seven pages It's introduced all sorts of things like the suitability of the different protocols, things like the higher level of security, smart cards, the thing about putting your credit card over the web, stuff like that, about the encryption and so forth "

Applications expertise is also a main driver or source of requirements for software product development

> "We're not warehouse professionals but we've been to fifty warehouses and come from, perhaps, personally, ten or so we pool our experience, naturally Now the company, I think we could say we know more about warehouses than most of our customers but we don't understand the detailed nature of their work but we do understand how to deal with pallets, how to deal with gravity feed systems, whatever it is "

In ERP systems, the ERP vendor's knowledge of both **business processes** and **application domains** is embodied in application reference models (Scheer, 1998) For the practitioners, these reference models are important resources for enterprise modelling, but since models are not substantive concerns, these reference models are not represented in the ontology chart

Thıs knowledge and expertıse guıdes the development of taılorable packages whıch are then avaılable ın the marketplace for deployment ın specıfic sıtuatıons These packages cover all but the most non-standard requırements lıkely to be found ın a gıven busıness area or applıcatıon domaın Some ERP vendors, such as Baan and QAD restrıct theır offerıngs to specıfic markets, called vertıcal markets, ın order to better serve the varyıng requırements wıthın that specıfic market

5 3 5 Taılorable Packages

Taılorable packages are not lımıted to ERP systems They are becomıng common ın many applıcatıon domaıns There are essentıally three dıfferent approaches to creatıng a taılorable applıcatıon package

(1) Customısable software whıch can be taılored locally by programmıng extensıons to the code

(2) Configurable software whıch can be taılored by configuratıon tables desıgned for the purpose There may be thousands of these tables ın a sophıstıcated product One versıon of SAP reportedly has 18,000 configuratıon tables One of my ınterviewees was workıng wıth a configurable package for unıversıty student records whıch had 3,000 tables ın ıt

(3) Component-based software whıch facılıtates the deployment of specıfic components chosen to suıt a gıven sıtuatıon

ERP companıes see component frameworks as the future dırectıon for theır products, but most products are currently based on the ıdea of product configuratıon, wıth the deployment of components only beıng done at a very hıgh level, usıng applıcatıon modules

> "The poınt ıs that ıt ıs very heavıly parameter drıven Thıs process of settıng all those parameters ıs called configurıng "

> "So lıke there are somethıng lıke 2,500 tables on the system, not tables ın the relatıonal database sense but parameter tables Each of those tables could have a number of parameters that need to be set So the down-sıde of thıs multı-functıonal and hıghly ıntegrated system ıs that because ıt does all thıngs to all people, you have to tell whıch of those many thıngs that you want ıt to do "

> "Order statuses, they are agaın ınfinıtely configurable, you can have as many or as few as you lıke and you can say that status A can only precede status B or vıce versa And status C can only happen on order types such and such And agaın, order types, you can have as many and as few as you lıke and put whatever condıtıons on them that you lıke And all of that was configurıng "

5 3 6 Market Concerns

Some of the practitioners I interviewed were engaged in developing software **products** for different kinds of markets Many of their concerns related to market-oriented software and these can be seen in Figure 5 Markets are considered by practitioners to be made up of existing and potential customers Rather than use market as a concern in addition to **customer**, I have made **market opportunity** (implying potential customers, or additional sales to existing customers) as another source of **market requirements** In many of the cases I investigated, the product was already on the market, and the developers were bringing out a new **release** of the current version, or planning a new **version** In general, the difference between new releases and new versions is as follows new releases fix some of the problems found in the previous release, and are available free to existing customers, while new versions in addition provide new functionality or better performance, or a new platform, but must be paid for, even by existing customers

In addition to the application domain and the market, the **product** itself could be a source of requirements, especially when existing customers report **problems** with the current release Managing such reports and prioritising the resulting requirements is an important consideration

Product expertise is another important source of requirements, and of information concerning the interdependencies of requirements with existing **features** of the product

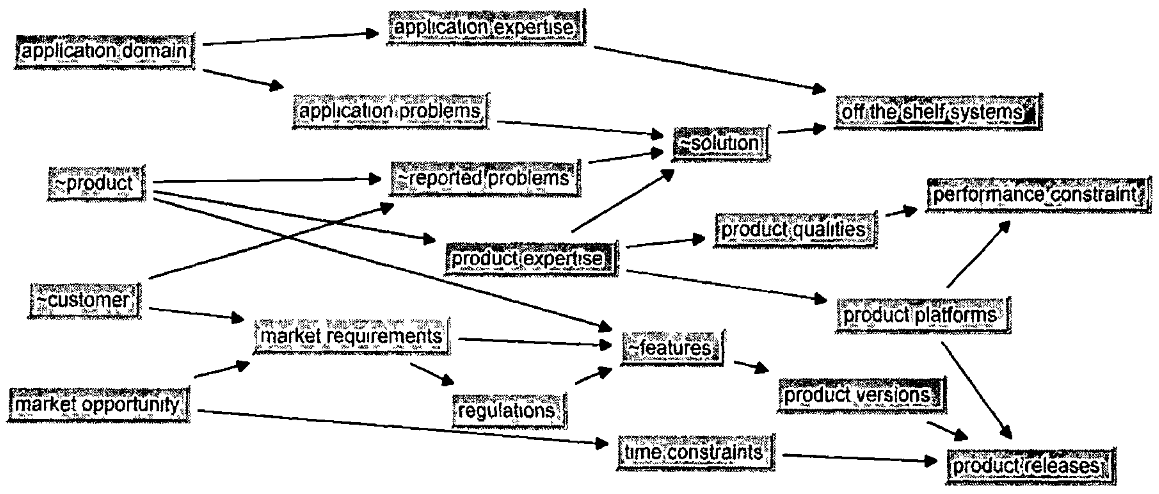


Figure 5 Market Concerns

" when you've got a mature product it's getting to where you want to be it's got everything integrated adding on that extra feature becomes more problematic because it has to be consolidated "

"A requirement is a perceived problem, a feature is a benefit "

Features are marketable requirements or groups of requirements A feature often has several associated or dependent requirements, which customers are not aware of Features may be treated as stand-alone, although interdependencies between features are considered to be a particular problem in ways that dependencies between other types of requirements are not It is interesting to note that the concept of a feature exists in the product or solution, as well as the requirements Unlike a requirement, a feature has continuity of existence, and therefore more visibility and existential integrity than a requirement, which will in most cases be subsumed, along with other requirements, into the functionality of the product or solution Another concept is the idea of a **bundle**

> "Marketable functionality a bundle is a collection of features specific to a vertical market segment
>
> For example medical and pharmaceutical manufacturers are really interested in lot traceability, much more so than food and beverage manufacturers, other discrete manufacturers are not interested in lot traceability at all So we would put that feature, not in the core product but in the medical bundle and in the pharmaceutical bundle and possibly in the food and beverage bundle "

Like other software systems, software products evolve over time New **versions of products** incorporating major new functionality appear less frequently than new **releases** fixing the reported problems and adding minor new **features**

> "There is very little difference between development and maintenance here certainly if you look at our flagship product it just growed and growed and growed "

These intermediate **releases** are often coordinated at an organisational level above project management, with more than one project associated with each release

> "Because we are dealing with a product that is advanced on a per project basis we have requirements for each project There are usually many projects to a release "

Market opportunities do not wait for vendor companies, so product managers are under the pressure of **time constraints** to get new releases onto the market There is usually a trade-off to be made between additional requirements and **time constraints**, which in turn affect the project deadlines

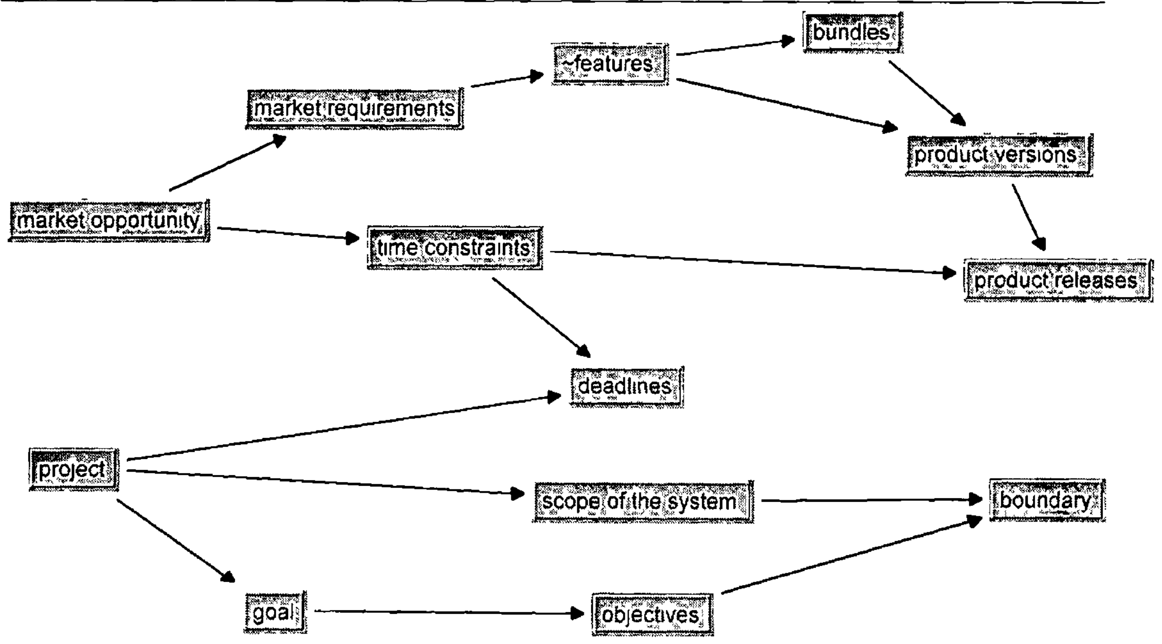


Figure 6 Market-oriented project concerns

This part of the ontology chart can be linked to the concerns shown in the first extract Figure 6 shows how the project concerns which were shown in Figure 2 could be wrapped around to link to market concerns

All of the **root concerns** have now been introduced All the excerpts shown so far begin with some of the **root concerns** and do not contain any of the rightmost (most dependent) nodes of the ontology We will now look at some of the successors to the concerns shown in the various extracts above These include important aspects such as **agreement** (a successor to **issues**), **system interfaces**, and **interfaces to other systems** We also return to the topic of different types of requirements

5 3 7 Different kinds of requirements

At this stage, it is important to note that all the relationships in this ontology are ontological precedence relationships There are no **isa** relationships shown at all, although they are interesting in their own right I have had to avoid including other any types of relationships, because I want to use the chart to clarify the type of relationship shown, namely the relationship between **concerns** that precede and succeed each other ontologically

Also, in order to keep the chart to a manageable size, I was forced to prune it by removing some of the nodes that were descended from some of the **concerns** that are shown in Figure 7, for example different concerns relating to states, such as transitions and constraints, and relating to reports, such as ad hoc reporting and standard reports These were found to fit much better ınto a taxonomıc style of analysıs, usıng **ısa** relatıonshıps, as presented ın the prevıous chapter

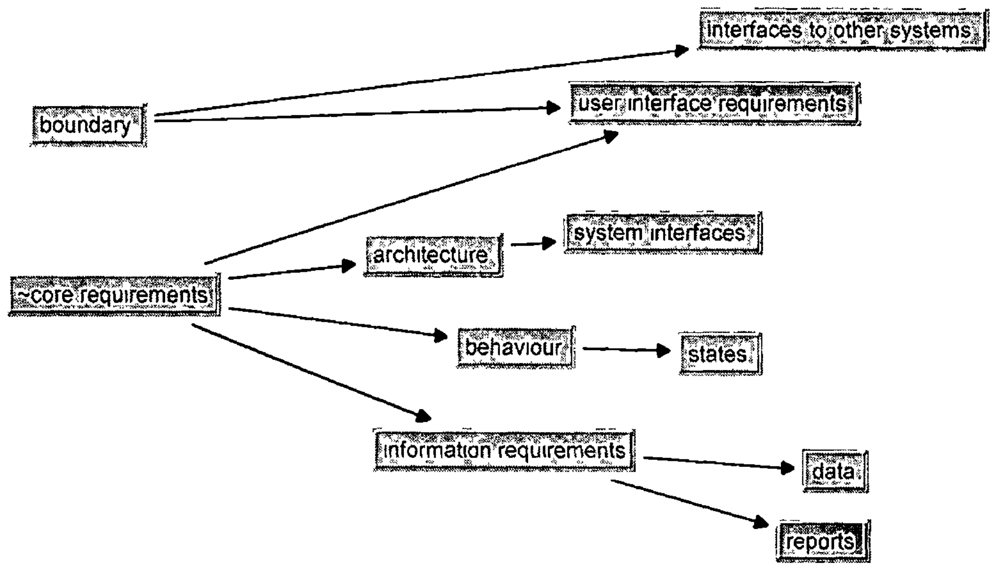


Fıgure 7 Types of requırements

5 3 8 Dıfferent types of solutıons

One of the ınıtıal aıms ın buıldıng the ontology was to try to trace the paths between problems and solutıons, but thıs aım was not entırely achıeved One possıble reason for that ıs that although practıtıoners often mentıoned **problems** and **solutıons**, both of these terms had dıfferent connotatıons ın dıfferent dıscussıons The varıety of **problems** and **solutıons** mıght have been better handled ın a taxonomıc analysıs For example there were dıfferent types of problems those reported by customers, problems wıth the process, or wıth the method beıng used, problems wıth the documentatıon, and problems wıth proposed solutıons, as well as the conceptıon of a requırement as a problem to be solved But such an approach would not have accounted for the relatıonshıp between the requırements as a set of problems, and the range of solutıons consıdered at the requırements stage

Fıgure 8 shows several of the ontologıcal descendants of **applıcatıon domaın** ın the ontology Accordıng to my ınformants who worked ın thıs area, a **customısed solutıon** was necessary whenever there were **non-standard requırements** (essentıally requırements that had not been antıcıpated ın the chosen package) Thıs was often the norm, despıte the ıncreasıng popularıty of standard packages, such as ERP systems It ıs possıble to extend many standard ERP systems by customısıng, but such many practıtıoners consıdered customısed ERP solutıons undesırable because of the lıkelıhood of conflıcts wıth subsequent releases of the ERP product A **configured solution** is currently the preferred approach to tailoring a package

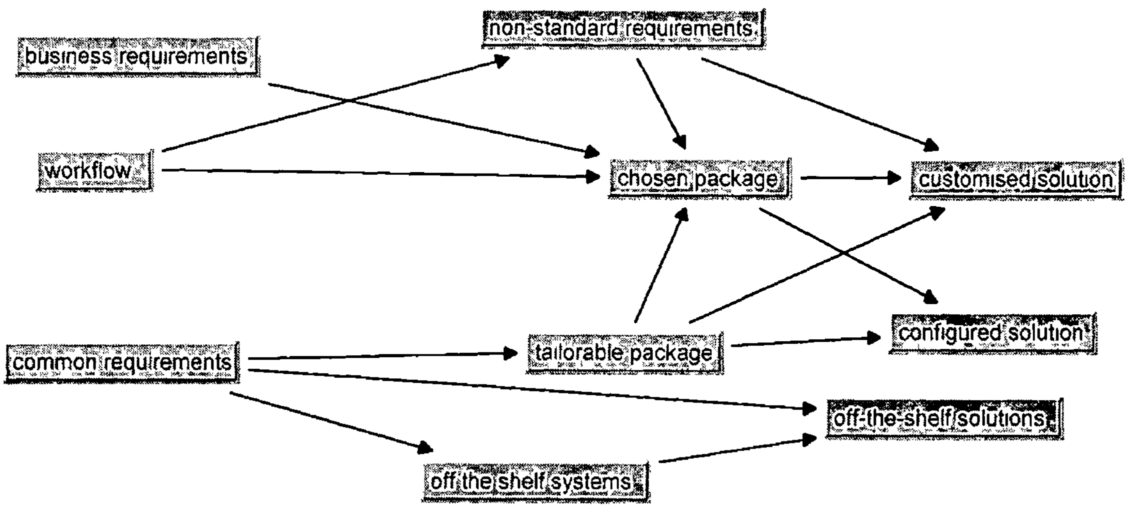


Figure 8 Different types of solutions

For some situations, it is possible that the requirements are so completely standard that an **off the shelf solution** is feasible, but this was not the norm Several of my interviewees were working with one specific supplier (of a warehouse package) to customise that package to their needs Although they were all essentially in need of a system to control a warehouse, none of these organisations could find a suitable **off the shelf package** to suit their needs In each case, the supplier company personnel worked with the client organisation personnel to document their requirements, before customising a solution to suit their budget and needs

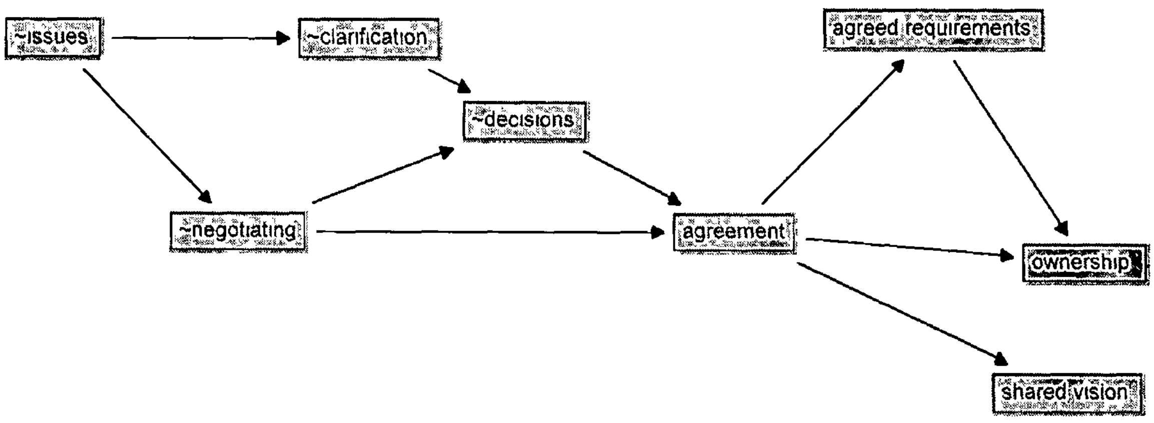


Figure 9 Focus on Agreement

Many of the practitioners saw themselves as problem solvers, or even called themselves solution providers, so they tended to look on requirements more as solutions than as problems For example

"We help them come up wıth the solutıon that ıs achıevable usıng the product that they have chosen "

"We propose a technıcal solutıon to a busıness problem "

5 3 9 Agreement

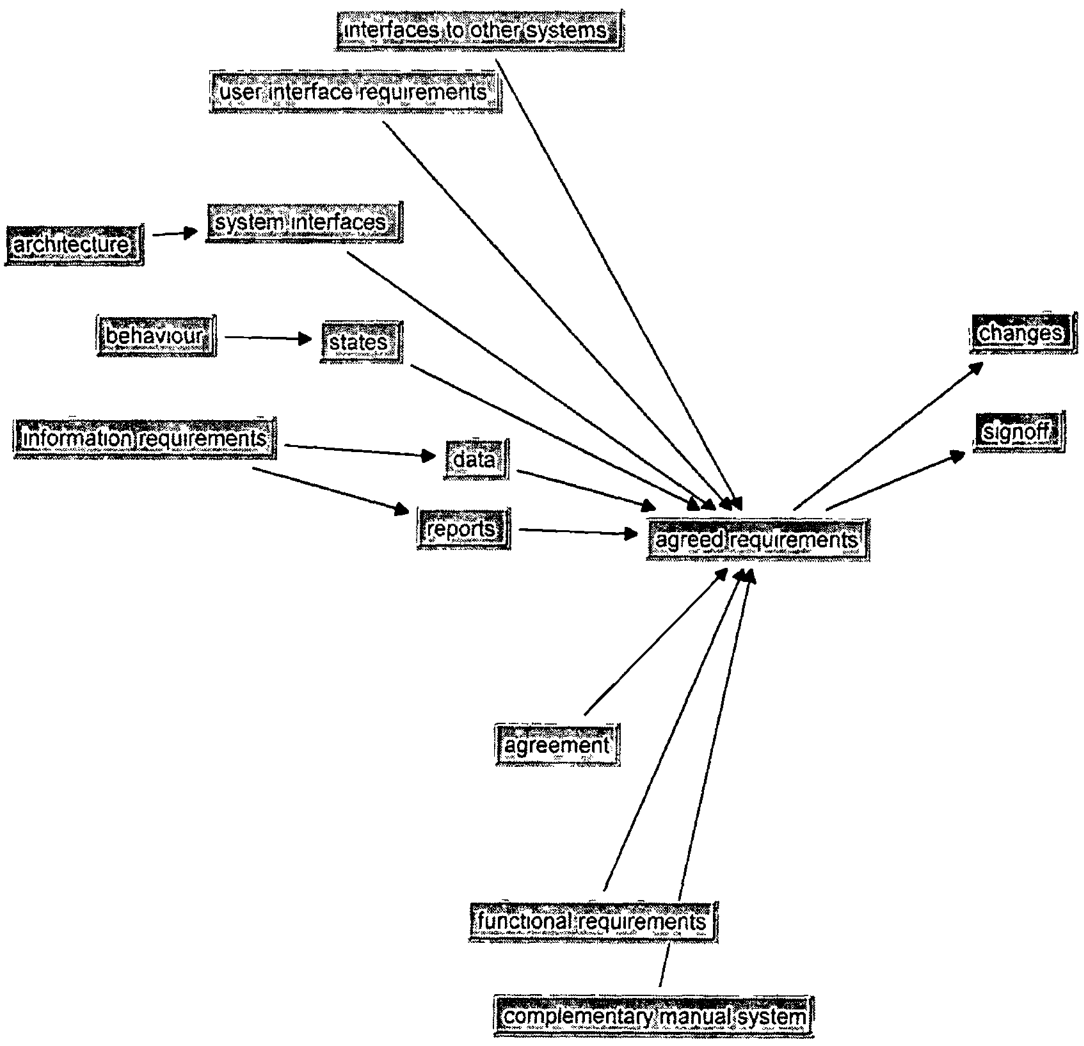
One of the most ımportant concerns dıscussed by the practıtıoners was that of **Agreement** The concerns shown ın Fıgure 9 focus on that concern Note that **agreed requırements** depended on a number of other concerns, as shown ın Fıgure 10 Several practıtıoners stated quıte clearly that **agreement** was a separate concern It ıs separate from the **agreed requırements**, and ıt ıs "not ın the document " " One practıtıoner stated

"If after the event the system does not meet the requırements there would not be much understandıng for pullıng out a requırements document "

In another case

"We sent back a fınal lıst of requırements whıch was statıng the lıst of requırements they sent us, worded approprıately for where the clarıfıcatıon was necessary, ıdentıfyıng whıch of those requırements we were goıng to ınclude ın the system and whıch we felt were outsıde the scope of the system and they then sıgned off on what we called the agreed requırements lıst "

Fıgure 10 Agreed Requırements



5 4 What ıs meant by a grounded ontology?

Any ontology ıs a catalogue of terms An ontology ın thıs sense could have been buılt by goıng through all the lıterature on requırements engıneerıng and pıckıng out all the terms used ın that domaın, arrangıng them, defınıng them, or relatıng them together The ontology presented ın thıs chapter was the result of analysıng a set of ıntervıew transcrıpts to fınd out what are thıngs that practıtıoners talk about when they talk about requırements The objectıve was to fınd out what requırements are all about, and how the practıtıoners construe both requırements and theır sources

The resultıng network ıs called a grounded ontology because the collectıon of terms ıt contaıns and the relatıonshıps between them are derıved empırıcal data gathered from practıtıoners ın the fıeld As such, ıt ıs a grounded theory of how the thıngs represented by the terms (the matters of concern to requırements practıtıoners) are related ın a conceptual network Usıng the grounded theory method of analysıs provıded a systematıc approach to the constructıon and verıfıcatıon of the network

Thıs process extracted a collectıon of terms representıng the requırements as needs that were talked about by the collectıon of practıtıoners ıntervıewed It then analysed the relatıonshıps between these **concerns** guıded by the notıon of ontologıcal precedence The ontologıcal analysıs was therefore bottom-up from the level of the **concerns**, rather than a top-down analysıs on an ıntervıew-by-ıntervıew basıs The result ıs lıke a patchwork quılt of all the concerns lınked together Metaphorıcally, ıt ıs an attempt to construct a map chartıng the overall terraın of 'requırements as needs ' Dıfferent ındıvıdual practıtıoners have fırst-hand experıence of only parts, albeıt large tracts, of thıs terraın, related to theır partıcular area of practıce and experıence

5 5 Conclusıons the Root Concerns

It ıs ımportant to note that the specıfıc relatıonshıps explored ın thıs chapter are ontologıcal precedence relatıonshıps only There are no **ısa** relatıonshıps shown at all, although they undoubtedly would make sense, between the categorıes that have been dıscussed The purpose of buıldıng the ontology chart was to clarıfy a partıcular type of relatıonshıp, namely the ontologıcal precedence relatıonshıp between the requırements **concerns** that precede and succeed each other The ontology as ıt stands focuses on the substantıve aspects of requırements, on 'requırements as needs ' It ıntentıonally ıgnores other aspects, such as requırements as texts, whıch have been treated separately ın the taxonomıc analysıs of the prevıous chapter It could be extended to ınclude these semıologıcal aspects as well Any extensions to include requirements as texts would appear on the right-hand side of the chart, because the relationships read from left to right

As with any grounded theory, the result is not final or definitive, but open to further work and validation It is grounded in the data that was available, and could probably be vastly improved by further investigation To continue the analogy with the map of a terrain, the current ontology of requirements is very like the early medieval efforts at geographical representation showing only the three continents that were known at that time, with Jerusalem at the centre, before more accurate navigation charts began to appear, along with magnetic compasses, in the 13th Century

The issue of validation will be discussed more fully in the final chapter of this thesis Each relationship between a pair of codes originated as a conjecture on my part, and was tested by querying their co-occurrence in the data Because of the varied and unstructured nature of the data, and the fact that these were not questions put to the participants, but issues that emerged from the data at a later stage, some of the relationships in the result were less strongly supported in the data, and others more so The overall idea was to capture as much variety as possible in the network The significance of the result is that it demonstrates the idea of an ontology based on this particular type of relationship, and draws attention to the most important sources of requirements

These sources are essentially the eight **root concerns** of the ontology **Project, User, Client Organisation, Business Area, Application Domain, Product, Customer**, and **Market Opportunity** These and their ontological descendants form the context and background of system and software requirements in the cases which I studied Some of them are obviously more relevant and more influential in some situations than in others All of them give rise to several other **concerns** and in due course, to requirements as needs, and eventually to requirements as texts

As a whole, the ontology could be regarded as a theory of how the root concerns govern the other concerns The different extracts that have been presented in the chapter are different views of the whole network, presenting aspects that are more significant in some situations than in others Each practitioner's 'locus of concerns' if overlaid on the network, continuing the 'map' analogy, would only cover parts of it Evidence from the interview transcripts, and the existence of gaps in the Codes/Primary Documents Table, indicates that this is likely, though practitioners were not asked about these in a structured way One of the possible uses of the ontology would be to explore its applicability to a different set of development situations, and see if it could be used effectively to distinguish different situations by plotting each situation's locus of concerns and identifying the types of requirements that typically arise in that situation

A more practical approach would be to take a list of the most fundamental concerns, the **root concerns**, as the point of departure, and examine how they might be used to differentiate between situations For example, **Client organisation** as a concern might dominate the practitioner's concerns in some situations and not matter at all in others, while **Market opportunity** would stand out as a source of requirements, or not matter at all, in different situations The matter of a **Product**, and whether it is being developed for a market or being configured as a solution, plays different roles in different situations, and no role at all in some other situations The implications of this idea are applied in the next chapter, where the eight **root concerns** governing requirements as needs are used as the parameters for distinguishing different types of requirements situations

Chapter 6 A Classification Scheme for Requirements Situations

6 1 Introduction

A working assumption of this research has been that the most appropriate type of system requirements documentation is contingent That is, it depends on the situation in which the documentation is written and used System/software development takes place in a wide variety of organisational and contractual situations However, as indicated in Chapter 1, there is no recognised or widely accepted classification of system/software development situations or at least none suitable for the purpose, so a provisional one is presented in this chapter

The nature of the relationship between the producers and the users of the software, for example, whether it is based on a formal contract or not, is one possible way to classify system development situations This takes account of the organisational context of development, but it does not account for some of the different concerns that come into play, for example, when the target system consists of embedded software rather than an enterprise application It is not useful, even if it were possible, to categorize situations by the problem frame of the target software because most systems of any reasonable size and complexity correspond to more than one problem frame (Kovitz, 1999)

Another possible way to categorize situations is by the application domain, but so far, there is no established taxonomy of software applications (Glass and Vessey, 1995) Robert Glass and Iris Vesey surveyed a number of published classifications of software applications, but found no convergence This is possibly because, as they point out, taxonomies are specific to the purposes for which they are used

This makes sense Some plants are classified together for cooking and serving, but in separate categories for the purpose of propagation and cultivation A cookery book will classify rhubarb as a fruit, and a tomato as a vegetable, whereas a gardening book will classify them as a vegetable and a fruit respectively

The purpose of the classification of requirements situations proposed in this chapter is to identify the major concerns that drive the requirements process and thereby suggest which are the appropriate types of statements of requirements to be used in each situation The classification is provisional, because it is based on a limited number of cases, but it is a framework capable of being extended

Rather than a taxonomy, or hierarchical classification, it is similar to a faceted classification (Leigh Star, 1996) Faceted classification is an approach to classification in which items are allocated to different classes depending on the values they assume under each of a series of headings or facets These headings represent the salient characteristics of the items being classified This approach to categorization has been proposed for the classification of knowledge, as an alternative to the well-known hierarchical Dewey system of classification that is widely used in libraries (Leigh Star, 1996)

The main advantage of faceted classification is that it is much more flexible than a hierarchical system Hierarchical classification systems are fixed They do not readily accommodate new items, nor do they easily allow for growth and expansion Faceted classification is much more adaptable and open to the inclusion of new items in the framework It does this in two ways

1 New items can be distinguished in terms of their different values under the existing headings, and
2 Additional new headings or facets can allow for more dimensions of discrimination between the items

6 2 Making sense of situations

A situation is the set of circumstances which form the context for some phenomenon of interest Circumstances are not orthogonal in their effects In any situation, it is the particular combination of circumstances that makes the difference to the situated phenomenon A set of aspects or facets therefore works reasonably well as a scheme for describing situations in terms of their circumstances

It could be argued that each situation is unique, and that, therefore, there is no satisfactory way to classify situations But different situations have at least some things in common, and these help, not only to describe situations, but also to classify them, for some purpose

It could be argued that the influences or factors affecting a situation, which create that situation, are not only the immediate ones, but include a whole range of broader conditions beyond the level of the phenomenon being investigated It could also be argued that local, individual, or personal factors also have a bearing on particular situations In grounded theory, this is explicitly recognised and formulated as the Conditional Matrix, (Strauss and

Corbin, 1990) This is an analytic aid containing several concentric circles representing the various levels of conditions and consequences, from the outermost international level through national, community, organisational, and group levels, to the individual and interactional levels Any phenomenon, for example, decision-making, can be studied at any of these levels, for example, the organisational level, but in doing so, the effects of the other global and local levels cannot be ignored (Strauss and Corbin, 1990)

In order to take a broad view of system/software development situations, my assumption, based on the interview data, is that it is sufficient to look at them at an organisational level Therefore, in order to classify such situations with regard to requirements documentation, it is necessary to focus on the sources and influences at this level that generate and shape the requirements themselves For these reasons, the facets which are proposed for this classification of system development situations are the main areas of concern in those situations They are the sources of the requirements They are the main influences on the task of documenting requirements My conjecture is that the eight **root concerns** developed in the previous chapter, namely, **Project, Users, Client Organisation, Business Area, Application Domain, Product, Customer,** and **Market Opportunity** can be used as the main headings of a classification system for requirements situations

6 3 Selective coding

This is the third and final level of grounded theory analysis and is concerned with selected 'core categories' and their relationships At this stage, the grounded theory begins to emerge and dominate the analysis process The process I followed at this stage was basically as follows One by one, I looked for and identified typical situations among the cases in the interview data that could be set apart from other situations using the eight **root concerns** as distinguishing criteria The first type of situation identified I called the **Problem-oriented Situation,** in which the requirements document is typically created by the **Customer** in the form of a Request for Proposals I found that two of the criteria, **Market opportunity** and **Product** had no relevance at all in this type of situation However, I used this fact to help distinguish it from other situations The other main headings, such as **Project** and **Users**, and some of their immediate ontological successors, such as the **Scope, Interfaces to Other Systems,** and **Stakeholders** were important criteria in characterising this type of situation

As a general point, it is worth noting that some of these criteria do not always apply in all situations, and this helps to distinguish the different types of situations Also, I found that I could use the extent to which each **root concern** applied in a situation, in terms of the impact of its various ontological successors on the requirements, to explain the distinctions I was finding This confirmed the root concerns as fundamental influences that drive the requirements process, because they are the main sources and generators of requirements To the extent that each does not have the same or equal influence in all situations, they help to distinguish situation types

The existence of the Problem-oriented type of situation suggested that there might also be a Solution-oriented type of situation, but it was not as simple as it seemed As it turned out, there were different types of Solution-oriented situations, that I could identify in the cases described by the practitioners Two of them were custom-built solutions, the other being a solution configured from a product, and two of them were contract-based, the other being an in-house solution I named the three situation types

1 Contract for a custom-built solution
2 Contract for a configured solution, and
3 Custom-built in-house solution

Problem solving could be regarded as the main focus of most system development activity, including the process of discovering and agreeing the requirements for the system However, in practice, problems and solutions are not always clearly distinguished from each other, because they are inter-related at different levels Solutions at one level form problems at the next level According to many of the practitioners I interviewed, the purpose of a requirements document is not always to state just the problem, but often also to state an outline of the solution, or the constraints on the solution, or the nature of the solution

An extreme form of solution-oriented situation is where there is a market for a certain types of ready-made solution, and this is developed to be sold to customers as a product The function of the requirements document in such situations is to enumerate features of the product which will solve the problems of many or at least more than one customer I called this a market-oriented situation to distinguish it from situations where solutions are developed for single clients or customers, which I called solution-oriented situations I was able to find three different types of market-oriented situations among the situations discussed by the practitioners I called these

1 Product for a vertical market
2 Product for targeted customers, and
3 Product for a single customer

The third category of market-oriented situation overlaps in some respects with the first type of solution-oriented situation, because there is a formal contract involved, but there are other aspects (concerns) which are different for this type of situation, as will be explained later in the profiles of the different situations This category underlines a major problem with using a hierarchical classification, and the benefits of using a faceted classification The category is identified as a separate category because it has a characteristic profile of concerns It does not fit neatly under one heading, such as market-oriented, or another, such as contract-based, but slots in between other situations, having some aspects in common with each of them

However, for presentation purposes, and to help the reader to cope with the amount of detail involved, these three main types of situations may be broken down hierarchically as follows and will be presented in the following order

A Different types of Market-oriented situations, i e
  1 Product for a vertical market
  2 Product for targeted customers, and
  3 Product for a single customer

B Different types of Solution-oriented situations, i e
  4 Contract for a custom-built solution
  5 Contract for a configured solution, and
  6 Custom-built in-house solution, and finally,

C Problem-oriented situations, essentially one type
  7 Problem-oriented

In the remaining sections of this chapter, these seven different types of situations are characterised and differentiated from each other by the emphasis and priority they place on some or all of the major root concerns, such as the **Users**, the **Customer** and the **Client Organisation**, and also by the impact that their respective related concerns (ontological successors) have on the requirements process and the requirements themselves Each situation type is presented as a descriptive profile, linked to the situations described by the practitioners, and abridged in a table where I summarised the interview findings in a general way The construction of these tables was an important part of the analysis process itself The final section then compares and contrasts the different situations

6 4 Seven profiles of typical situations

A key distinction between market-oriented and other situations is the focus on the **customer**, as distinct from the **user** In market-oriented situations, it is much more important to have satisfied customers, than to have satisfied users

Among the cases reported by the practitioners interviewed, I have identified three different types of market-oriented situations which I called

1 Product for a vertical market,
2 Product for targeted customers, and
3 Product for a single customer

Clearly, this categorization of market situations is not complete, because I have no situations where there are many customers (e g mass-market products, such as office products, games and other commodity software of the type sold in shrink-wrapped boxes) I have not had an opportunity to study mass-market situations at first-hand, but have gleaned enough information from reports in the literature to know that they are different in several respects from these other market-oriented situations

6 4 1 Type 1 Situation Product for a vertical market

This situation typically occurs where software products are being developed in response to a perceived need in a particular market, either covering a well-defined application domain or a specific set of customers in a segment of a wider market The software company aims to become successful by responding specifically to the requirements of a well-defined subset of a larger, wider market The software company may in turn divide its market into a number of segments (verticals) in order to more fully address each of their needs

> "We are very lucky because we have a focus on (depending on who you talk to) five or six verticals, all discrete manufacturers We have a tight focus on what these people need "

Like other market-oriented situations, the vertical market situation is characterised by projects which identify specific priorities among the requirements, and which have strict deadlines which are tied to market time constraints, for example planned marketing and advertising campaigns, upcoming trade shows, announcements, and anticipated competitors' product releases In this type of situation it is common to have multi-disciplinary project teams consisting of business analysts, marketing people, designers, programmers, testers and others with **product expertise** working together to define the requirements for the evolving product

In common with mass-market software, in a vertical market, it is quite common for successful products to evolve over the years, gaining market share and additional customers as more features are added with each new release There may be several projects going on simultaneously to extend, enhance and fix the product Requirements are managed on a project-by-project basis, as well as a product basis In many of these situations, the scope of a project is much smaller than the scope of the product There may be two or more levels of planning in the organisation, typically, product planning and project planning

Market requirements have a strong influence on the requirements that are input to product planning

> "It seems to be this interaction between industry analysts, market analysts, CEO' s Forbes magazine, whatever it is These people seem to set the agenda in enterprise software They raise the

> things that a CTO or a CIO might have in his head coming into work on Monday morning "

All requirements are assigned a **priority** value, which can be based on perceived market value Product requirements are selected for implementation in a project on the basis of priority Within a project, even during implementation, requirements of lower priority may be dropped from that **project**, i e postponed until a later **project**, in order to keep to the **project** deadlines

Like other market-oriented situations, requirements definition is done without much involvement by **users**, who are not directly involved in the process Although some companies may specifically address usability design, it does not seem to be important at the requirements stage I have not come across any cases of market-oriented situations where users were directly involved They might be represented in focus groups or by user groups for example, or their interests taken into account in other ways, but they are never genuinely involved in mandating, reviewing, or approving requirements in a market-oriented situation This factor distinguishes Market-oriented situations from those classified as Problem- or Solution-oriented situations

The internal **client organisation** seems to play an important part in the vertical market situation A hierarchical organisation of Product steering committees and Project steering committees plans and oversees the introduction of new and improved products The approval of these committees is an important milestone in the project

The role of the **customer** (as distinct from the **user**) in helping to define requirements is important in the early stages of requirements process Some **customers** are more influential than others, maybe having more installations of the product, or somehow, more clout in the market

> "We have core accounts perhaps, that are saying if you really want to keep our business, then you'd better develop "

The influential **customer** suggests a list of requirements, the marketing people have their say, the consultants and business analysts have their say, and product steering committee have their say

> "At that point we have got away from the list of things the customer would like to what we want to do, that's when we start making the hard decisions "

Product planners help to determine the cost of development and marketing people help to determine the value of each item of the list of enumerated requirements, rated for complexity and for priority, before product requirements are allocated to projects Application domain knowledge and product expertise are important inputs to this process, and these are embodied in various groups of business analysts and designers.

Requirements tend to be grouped in various ways, as discussed in Chapter 5, page 89. One way is to group requirements around a **feature**. A **feature** is "marketable functionality," listed in sales brochures, that tends to be supported by a number of other dependent requirements that are not mentioned in the brochure. Another way to group requirements is in **bundles**. A **bundle** is a collection of features marketed to a specific section of a vertical market as a product variant.

Areas of Concern:	**Type 1: Product for a vertical market**
Project	Product requirements are allocated to Projects which are related to Releases. Priorities for the next release of the product are influential.
Users	Focus groups are often used in the discovery of user requirements.
Client Organisation	Different levels, for Product Planning, Project planning,
Business Area	Not relevant unless it is the Application Domain. If so, Functional areas and Business knowledge are important.
Application Domain	Knowledge of Regulations or codes of practice in the target market is an important source of Common requirements
Product	Each release contains selected features, so the scope of the project is much less than the scope of the product.
Customer	There are many customers, but not on the scale of 'mass-market' situations.
Market Opportunity	Products evolve in Versions in response to market opportunities Bundles of features are developed to address different market segments.

Table 6.1 Type 1 Market Situation: Product for a vertical market

6.4.2 Type 2 Situation: Product for targeted customers

".. the idea being that the customer will help us define a functional product. So we have to balance the needs of the first customer with the needs of the marketplace."

In this type of situation, compared to the vertical market situation, there are fewer customers in the market, and consequently they each have much more influence on the inception and development of products. Typically, there is a Product Manager, or other 'customer-facing' person responsible for the requirements. This normally involves the writing, approval and customer validation of a requirements specification. During the project, particularly the early stages, interaction with the target customers is regular and frequent, in order to ensure that there is a market for the end result.

Another contrast to the vertıcal (and mass-) market sıtuatıons ıs that the product tends to be a new product, or a radıcal re-desıgn of an older one Thıs sıtuatıon tends to occur ın markets where the underlyıng technology and what ıt can achıeve are rapıdly changıng, such as ın the telecommunıcatıons market

Requırements ın thıs sıtuatıon come from two maın sources

1 external, ı e a customer demand
2 ınternal, ı e antıcıpated market demand, or features defined ın standards

Internal requırements can be proposed, or suggested for ınclusıon ın the product, by product management or any development personnel, wıth the relevant **product expertıse** or **applıcatıon domaın knowledge** Typıcally, the process ıs started by an enquıry from a sıngle customer, who may submıt an Invıtatıon to Tender The Product Manager works closely wıth thıs potentıal customer to define the requırements, and wıth the more technıcal development personnel to determıne the cost of the proposed product, whether a solutıon already exısts, or to what extent the technology exısts A market analysıs ıs undertaken to determıne the value of the product, how much other demand there ıs for ıt ın the market The resultıng document may be called a Busıness Opportunıty Specıficatıon Its purpose ıs to provıde the basıs for a decısıon whether to proceed to develop thıs product or not If ıt ıs successful, the project goes ahead and a detaıled requırements specıficatıon ıs prepared

As well as the customer's requırements, the requırements specıficatıon ıncludes system requırements and possıbly standards In these sıtuatıons, system requırements tend to be dıstınguıshed from the customer's requırements

> "They were standard and they weren't up for negotıatıon, they'd actually seen the system requırements and were happy wıth them "

Many of the system requırements are generıc, rather than project-specıfic, and lıke the standards, may be extracted from a database of requırements or they may be drawn from sımılar or prevıous products, or from system archıtecture desıgns

The customer who submıtted the ınıtıal enquıry plays an ımportant part ın the process of valıdatıng the requırements specıficatıon, and possıbly the test specıficatıon, but there ıs no formal contract wıth the customer ın thıs type of sıtuatıon However, ıt ıs ımportant to have the ınıtıal customer's cooperatıon, ın order to ensure that there wıll be at least one customer wıllıng to buy the finıshed product, but the aım ıs to sell the same product, more or less, to at least two or three other targeted customers For example, key users (from the customer sıde) may help to define and/or valıdate requırements

> "So we take their ideas and their requirements and try where possible to genericise them and make them applicable to a number of other clients potentially rather than build something into the application which is particular to that first client"

If the first customer withdraws from the project, this does not mean the project will be cancelled, once it has been approved, as the market analysis has shown that there is a demand for the product

The requirements specification goes through a process of review and revision, before being approved by the internal organisation This approval is a milestone which establishes a baseline set of requirements, after which changes must be controlled and managed In a dynamic market, it is important to be able to react to changing market requirements, as each extra potential customer may add considerably to the revenue from a product, but as in any project, it is important to control the scope of the requirements

> "It is an agreement of what we **desire** from the system What actually gets produced depends, on a couple of issues it is not contractual"

Targeted customers agree to participate in field trials of early releases of the product

> "You basically have maybe ten to fifteen new features in each particular software release We have a couple of customers at the moment who are in a field trial with this box"

However, these customers are not as deeply involved in the project as might be wished by the development organisation, in terms of their commitment to the product

> "I think the level of involvement within the customer organisation was yes we need to get some requirements to them and then it is their problem to build that, and we have got two or three key users and I am sure that they can make sure that the system does what it is supposed to do But for these key users this is very much of a part-time task"

Areas of Concern	Type 2 Product for targeted customers
Project	The requırements are subject to ınformal Agreement wıth the target customer, and subject to formal Approval by the ınternal Clıent Organısatıon, eıther at a hıgher level, or peer level
Users	Key users from the customer sıde may help to defıne and/or valıdate requırements But ıt ıs dıffıcult to get users ınvolved
Clıent Organısatıon	A combınatıon of an external clıent and ınternal organısatıon Often seen as joınt development
Busıness Area	A source of Common requırements for the Product
Applıcatıon Domaın	Understandıng of the applıcatıon domaın, e g relevant Standards, etc The Product ıs state of the art ın that applıcatıon domaın
Product	Features of the Product may ınteract wıth each other, requırıng a hıgh level of product expertıse
Customer	Few customers, compared to vertıcal market sıtuatıon Some of them may have consıderable applıcatıon expertıse Customer representatıves valıdate requırements
Market Opportunıty	Addıtıonal market requırements help to add value to the product Tıme to market ıs crıtıcal Addıtıonal platforms may be another source of system (technıcal) requırements

Table 6 2 Type 2 Market Sıtuatıon Product for targeted customers

6 4 3 Type 3 Sıtuatıon Product for a sıngle customer

In thıs sıtuatıon, the developer has only one ımmedıate **customer** The software ıs beıng developed for the market, but there ıs a contract ın place, eıther wıth developer's parent company, or wıth a contractıng customer In contrast to the other market-orıented sıtuatıons, ın thıs case the **clıent organısatıon** ıs external rather than ınternal The product manager works for the **customer** and represents both the customer and the user In contrast to other contract sıtuatıons, whıch wıll be dealt wıth later ın the chapter, ın thıs sıtuatıon there are sıgnıfıcant market-orıented concerns

Typıcally, a **product**, or more lıkely, part of a product ıs beıng developed by a software house on behalf of the customer Thıs type of sıtuatıon often pertaıns ın the development of embedded software

> "We do basıcally the applıcatıon layer and the team over ın Belgıum does the drıver layer So the customer would have a person who effectıvely manages all the software for the product "

The ımmedıate **customer** for the software may be a larger, more well known (brand) name than the developer company, and may be represented ın the project by staff who are extremely knowledgeable about the product as well as the applıcatıon domaın

> "They contract us to do a partıcular job, but the guy we would deal wıth ıs the Product Manager, so we agree wıth hım what exactly we're goıng to do "

The eventual **customers** for the encompassing product (often consumers) are much less directly involved than in the other market-oriented situations discussed earlier The marketability or value of features is of much less concern to the developer company, compared to other market-oriented situations Relative priorities among the requirements are not a major concern, or may not be considered at all

Like other market situations, the **product** and, consequently, the **project** will be influenced by market **time constraints** that give rise to contractual deadlines Compared to the other market situations, these are more likely to be strictly enforced, but without the flexibility to drop **features** or requirements that may be lower priority

A single **version** of the product is the concern in such a situation, so the **project** is more of a concern to the developers than the **product** It may be a new **version** of an existing product, so the existence of **reported problems** with previous versions will influence the requirements

The relationship between the immediate **customer** and the developer company is typically managed by two people, generally engineers because of the type of product There is usually a **product manager** on the customer side, and a project manager on the supplier side The initial requirements statement comes from the **product manager** In one particular instance, this was a document called the URS (User Requirements Specification), so-called because it was written from the point of view of a user using the equipment, in this case a recordable CD player

> " the URS, which is the product manager's document, in which is specified hardware and software requirements, user-level requirements of what he expects the player to do "

The project team then works closely with the customer's product manager to refine, elaborate, and agree requirements, because the product manager's document is not detailed enough to define the agreement between them The more detailed document, particularly for embedded software, might be called the behavioural requirements document

> "What we agreed with the product manager is, the contract is based on this document and this document supersedes his document so we get him to review our one and if he says it's OK then this is the, kind of, if this says it should do something then it should do it "

The **application domain** is not usually a **business area**, but some other domain such as consumer electronics, industrial equipment, etc The functionality of the product is understood in terms of its **behaviour**, how it reacts to a set of externally visible **events**

> "So, these are events that the application perceives So, obviously, you have all the keyboard stuff, then you have events coming up from the drive as such, and then you have other

> events coming from external digital input, and stuff like that, right?"

To the practitioners, it is important that this document be complete, in the sense that it covers every feature that the product must have and every function it must be able to do In embedded software, this means that the product must be able to handle every event, or more precisely, every possible combination of state and event, and the document must be specify these

> "So, what we tried to do is to evaluate the response of the system to each event in all possible relevant states of the variables "

Everybody must understand and agree what the response of the product is supposed to be in every single combination of **state** and **event**

> "Now, it may happen that we either didn't cover this case, the case in contention, or he didn't understand fully what it would mean to us so then the document is taken as, kind of, complete in that is what we will produce so if he wants something different then he will put in a change request on the document "

Areas of Concern	**Type 3 Product for a single customer**
Project	There is a formal agreement with the Customer, represented on the Project by a Product Manager There is an Agreed deadline and little scope for selecting requirements based on priority
Users	The role of user representatives is played by developers, not real users
Client Organisation	The client organisation is the external Customer
Business Area	
Application Domain	Domains other than business areas A deep understanding of the application domain is an important source of requirements
Product	Product functionality is understood in terms of its behaviour, how it responds to events, and its behavioural constraints
Customer	The external Customer may in turn have many customers, typically consumers
Market Opportunity	The project is not usually concerned with other opportunities, except as a source of new contracts

Table 6 3 Type 3 Market Situation Product for a single customer

6 4 4 Solution-oriented situations

While market-oriented software development is concerned with creating generic **off-the-shelf solutions** in situations that are relatively removed from the context where these solutions will be deployed and used, solution-oriented development is more concerned with delivering a solution in a specific situation of use

Solution-oriented development seems to be particularly associated with the organisational use of systems Organisations vary in many ways, in their goals, in how they are structured, in how they interact with their environments, and in their size and complexity It follows, then, that organisational system requirements vary quite considerably The systems that are used in an organisation are in a sense embedded in that organisation, and therefore, requirements for them need to be specified *in situ*

One of the most important concerns in solution-oriented situations is the concept of a **business area** that the **client organisation** is engaged in, and that in turn gives rise to concern with the **business need** for the system A large system might involve several **business areas**

> " each subgroup being assigned a particular business area, for example, there was the Production Planning and Control and the Marketing area actually, as that was seen as being so closely aligned, there was Finance, there was Human Resources, there was Materials "

Because business organisations interact with their environments, and to a certain extent may be controlled by their environment, their requirements can derive from a wider context, for example

> "Well it very much influences the way in which we do things because we're a company with 50 people Our partner companies are companies with tens of thousands of people Our parent companies are companies with tens of thousands, if not hundreds of thousands of people We don't have very much clout in terms of making them do things the way we insist on them being done "

> " a regulation enforced in the EC is a regulation enforced in this country as well, so you can be driven by outside forces "

As discussed in chapter 5, pages 83 and 84, **organisational change** is a common source of disagreement and **issues**, particularly where organisational systems are being developed This can lead to many difficulties in agreeing what the requirements are going to be, and so, the needs of different constituencies, such as different groups of managers, need to be addressed

> "The older managers were keen that they would get the same support that they were getting already They just wanted to ensure that they wouldn't lose anything while the newer managers wanted new things "

Many organisations are now turning to standard packages to support their organisational information systems The same "standard solution" may seemingly be deployed by organisations as widely diverse as banks, hospitals, government departments, and manufacturing companies

It is recognised that organisations must be able to change and adapt, and to change their information systems, as their environments are constantly changing Having a configurable system is considered to be a great help in this regard, because the same basic system can be adapted to many different configurations, as **business processes** are changed This change is often called Business Process Reengineering (BPR)

> "Once you install SAP, it allows you to go ahead with BPR, because it is configurable "

However, much technical work must be done to configure the software product, and before that, to determine the organisation's requirements, before such a solution can be deployed in the first place Compared to market-led situations, the influence of the end **users** and their organisational context is much stronger in solution-oriented situations The influence of a **product** is there when a standard solution is being deployed, but the influence of the **project** and its related concerns is stronger when there is a contract between two parties

These situations often involve the requirement for a **complementary manual system** also A considerable difficulty that is often overlooked in deploying an automated system, especially one that is embedded in an organisational situation, is the problem of making sure that it is used in a systematic way

> "The process of defining what a software system should do in many ways just involves clarification of what a procedure is If you're not able to do that, if you can't define requirements "

I have identified three different solution-oriented types of situations among the cases in the interview data These are (currently) called contract for a custom-built solution, contract for a configured solution and custom-built in-house solution These categories are not final, as other solution-oriented situations can be envisioned

For example, it is possible that some organisations acquire a standard package and configure it in-house, but I have not come across any instances of this Such a situation, if it exists, would have some overlap with two of the types of situation identified above, and could be called an in-house configured solution

All of the practitioners I interviewed who were involved with standard packages, whether as customers or suppliers, reported on projects which involved configuration, not customisation Where a standard package is used, but needs a considerable amount of customisation, as distinct from configuration, then a situation arises similar to that where there is a contract for a custom-built system

6 4 5 Type 4 Situation· Contract for custom-built solution

> "A very good case can be made for **not** outsourcing but the commercial reality is that you would be compromising on what is available outside the company "

> "But you are not going to get the best solutions without specialist skills and you get this when you are using an external company "

In this type of situation, an external software development company, such as a software house or a consultancy company, has a contract with a **client organisation** to develop a **customised solution** to meet the requirements of the client This situation is much less common than it used to be, due to the wider availability of standard tailorable packages which cater for most common organisational and business requirements However, many organisations continue to have **non-standard requirements** that for one reason or another cannot be completely satisfied by any of the packaged solutions on the market, and so, this type of situation of bespoke development continues to occur The detailed requirements are generally agreed between the supplier and the **client organisation**, but they are defined by the supplier, not the customer, because

> "In most cases clients haven't gone to enough trouble themselves to produce a good, clear requirements document "

The **project** is an important focus of concern, particularly in relation to the **stakeholders** and their **objectives** In common with other solution-oriented situations, and in contrast to market-oriented situations, the **users** of the system are often closely involved in the project

Because there is a contract between (normally) two parties, there is only one **customer**, and that is the **client organisation**, and the needs and objectives of that organisation are the main drivers of the requirements

> "In recent times we have actually set up a corporate policy of objectives and the theory is anyway and the theory I stress is that we are only supposed to be addressing projects that actually meet the aims of the organisation, the critical success factors of the corporation, right "

Typically, the **organisational objectives** of the client take precedence over any individual **user requirements**

> "Inevitably the person using the system is very immersed in the detail and they come to you with a list of requirements, The systems analyst really needs to look at the whole process "

If it is a fixed price contract, which it usually is, the supplier organisation needs to take steps to reduce the risk inherent in the project The supplier needs to control costs, and these are related to the **scope** of the system, which needs to be managed This is usually done by focusing on the solution, and how much implementation work is going to be needed

> "We are much more in the business of, they tell us what the problem is and we suggest the solution, we might try to clarify the problem, and then implement a solution
>
> "So we propose a solution in technical terms, so we don't really propose the business solution if that's what you are asking We propose a technical solution to a business problem "

"At the end of our analysıs phase, we should have, ın busıness terms, what the solutıon ıs And how ıt ıs goıng to functıon From then on, ıt's ımplementatıon "

Negotıatıng and **agreement** are ımportant concerns ın any contract sıtuatıon The clıent personnel and the supplıer personnel typıcally work closely together ın a joınt effort to establısh and valıdate the requırements Thıs process takes place ın the course of many meetıngs and dıscussıons

" and there would be mınutes of meetıngs when there were dıscussıons about the prototype and about the functıonalıty that was requıred "

"If people pushed you for one thıng rather, you could say look well ıf we do thıs, you can't have the other I mean, ıt was a huge negotıatıon exercıse of, well, ıf you want thıs, you are lookıng at, you are goıng to be three months later or whatever ıt ıs "

"The best way I know, the way that requırements have normally been captured ın any project I have ever worked on, was the way I have just descrıbed, whıch ıs meetıngs, people wrıtıng documents, cırculatıng them, havıng revıews, that sort of stuff "

In a large organısatıon, these meetıngs mıght take place at varıous levels of responsıbılıty, such as steerıng commıttees, ın addıtıon to project team meetıngs

"Steerıng groups were set up wıthın the organısatıon ıtself, steerıng groups were set up wıth the trade bodıes, wıth the software houses, wıth our network supplıers and basıcally we servıced all those groups ın some way "

In some cases, there may be a lack of commıtment on the part of the clıent organısatıon, or parts of ıt, to the project, and the external company has to take control of the sıtuatıon, or ıt wıll lose money on the project

"I thınk the process has to be much strıcter, ıt has to be a tougher approach, when ıt's two dıfferent companıes It's ın our busıness to make money from them, and we are goıng to lose money ıf they don't provıde the commıtment that ıs necessary from them to make a success of the project

"So there needs to be commıtment on both sıdes, and sometımes clıents well they may be dısorganısed, or they may be overstretched wıth other commıtments "

Areas of Concern	Type 4 Contract for a custom built solution
Project	The Scope of the project must be managed and controlled There is an agreed Deadline and milestones
Users	Individual User Requirements are less influential than the Organisational requirements
Client Organisation	Negotiating between the Client Organisation and the supplier organisation is significant Client Organisation IS/IT personnel may also be influential
Business Area	The Existing System may be a source of knowledge about the Business Processes and Workflow The Solution is seen in terms of Functions
Application Domain	Non-standard requirements may be significant Application expertise on the part of the supplier may be useful
Product	
Customer	The Customer is the Client Organisation
Market Opportunity	

Table 6 4 Situation Type 4 Contract for custom-built system

6 4 6 Type 5 Situation Contract for a configured solution

"The typical project we do involves putting in SAP as fast as possible "

In these situations, the commercial software **product** is usually selected in advance, and the aim of the **project** is to configure that software to the **client organisation**'s requirements as quickly as possible

The solution provider personnel are typically very knowledgeable about the **product** and its capabilities, and the requirements task is essentially to describe the **client organisation**'s requirements in terms that correspond to the functionality supported by the **product** The **users** and their requirements are taken into account to some extent, but because these products typically serve to integrate enterprise applications and data, the **organisational requirements** are much more dominant than any individual user's or manager's requirements

"The integration manager That was somebody who knew all of what the package did and his job was to ensure that the business model that was going to be put onto the SAP system for materials, for example, was not in conflict with something that the finance people wanted to do, because it is a very integrated system and it is important that everybody's requirements are cognisant of everybody else's, so that was the integration managers job "

Similarly, the **Business Areas** which are covered by the **project**, in terms of their **business processes** and **workflow**, are an important influence and source of the requirements Some form of business process engineering is typically part of the process of configuring a solution This typically takes place as a series of group meetings, often called Workshops, and theır purpose ıs to focus on the dıfferent **busıness areas** and **busıness processes** of the organısatıon Representatıve **users** play an ımportant role ın these workshops, together wıth the ınternal IT department and the solutıon provıder company's personnel

> "What ıt ınvolved was, we worked ın bıg group sessıons, the way we organısed the people was we had what we called alpha users from each department, so there was somebody from, say fınance, a long term employee of fınance, and sımılarly materıals and human resources and so on "

> "What we dıd was we sat ın a room and we saıd okay we are goıng to have materıals management, we are goıng to have fınance management, we are goıng to have human resources etc so ıt was a structured approach "

> "There wıll have to be materıals management, so what does that actually mean, so we'd break ıt down and we'd say well there ıs goıng to be ınventory control, there ıs goıng to be reportıng requırements, there ıs goıng to be receıvıng, there ıs goıng to be purchasıng, there ıs goıng to be ıssuıng, and then we'd break each one of those down even further "

Typıcally, descrıbıng these **busıness processes** leads to the ıdentıfıcatıon of **ıssues** whıch must be resolved ın relatıon to the capabılıtıes and functıonalıty of the **product**

> " thıngs lıke that, would we want to do, for example, to do weıghted average prıcıng, would we want to do straıght lıst prıcıng, thıngs lıke that, costıng of the ınventory that would be rather than prıcıng, that kınd of stuff would come up So we'd lıst those ıssues and then we'd resolve those ıssues one by one and ın the type 2 document there would be a lıst of the ıssues and what decısıons were taken and why they were taken "

In a sense, the requırements are functıons selected from the vast range of functıons whıch can be supported by the **product** Thıs ıs ın contrast to other solutıon-orıented sıtuatıons The functıonalıty of the **product** embodıes and represents the **knowledge of the applıcatıon domaın**, whıch would otherwıse be a basıc and ımmedıate source of requırements Thıs knowledge ıs contrıbuted to the project by the solutıon provıder personnel, and combıned wıth knowledge of the relevant **busıness areas** and **busıness processes** elıcıted ın the workshops, ıs a major source of requırements for the confıguratıon process

> "If you've made the decısıon that you were goıng to ımplement S A P , then ıt's a good ıdea It's a pragmatıc way of achıevıng an ımplementatıon "

Agreement between the partıes ıs always an ımportant concern ın contractual sıtuatıons, and contracts for confıgured solutıons are no exceptıon Sımılarly to sıtuatıons wıth customısed solutıons, project rısk, costs and project **scope** are also sıgnıfıcant concerns Thıs ıs sımılar to the customısed contract sıtuatıon, ın that the requırements are typıcally descrıbed ın terms of functıons

> "We would produce a busıness bluepront whıch descrıbes the functıonalıty that we wıll confıgure for the clıent very clear as to

> what the deliverable will be in terms of how the software will function "
>
> "Scope is very important, so that you don't get runaway costs and also, on the other side, to make sure you're not missing anything for the client
>
> "There is still the problem of scope creep Maybe scope creep is the wrong term for it We would give a certain amount that we never signed up for, it happens "

Requirements creep happens, probably more so for configured solutions than for customised ones, for two reasons One reason is that the effort involved in configuring extra functionality is considerably less than that involved in implementing it from scratch The other reason is the phenomenon known as IWKWIWTISI or "I won't know what I want until I see it "

> "One thing you need to control with SAP, once you start working with the client and they see the functionality that is available with the product like SAP "
>
> "When we start project it is very much led by the consultant but then later the client will begin to see what can be done and say hey we could use that and You say yes you could do, that wasn't what we are addressing "

A traditional problem that arises between system developers and users, particularly in regard to organisational systems, is the knowledge gap between them The developers understand the software, while the users do not, and the users understand the business, while the developers do not Configured solution providers aim to overcome this problem by acquiring specialist knowledge in particular business areas as well as having expert knowledge of a particular product Still, it is difficult to remember to speak to the users in the language of their own business, because the vocabulary of the product tends to dominate the solution provider's thinking As solution provider admitted, users need to remind them of this

> "He said 'Would you stop talking to me about SAP and start talking to me about business?' "
>
> "There is inevitably a language/terminology that develops around these things and the client picks this up and will be talking the same language after we finish with them "

However, this does not necessarily mean that the client organisation will have the necessary knowledge and skills to support the solution in the future, as the product evolves, as demonstrated by the case of the client that was migrating from Version 2 to Version 3 of a product

> "They knew the terminology all right but they didn't know about the extra functionality "

Areas of Concern	Type 5 Contract for a configured solution
Project	The Scope of the project is very important Scope creep is a significant risk Speed of implementation is important, so there is an agreed Deadline
Users	Individual User Requirements are much less influential than the Organisational requirements Representative Users take part in workshops
Client Organisation	Organisational objectives dominate the requirements, for example, integration of Business Areas Issues often relate to mismatches between the Existing System and the Product functionality
Business Area	Business Process engineering of the target Business Area(s) is key Some Business Processes may need to change (BPR) in order to achieve Organisational Objectives
Application Domain	Knowledge of the Application Domain is embodied in the Product in terms of data types and available functions
Product	Product Expertise on the part of the supplier is an important source of Solutions The Solution is seen as Functions provided by the Product
Customer	The Customer is the Client Organisation
Market Opportunity	

Table 6 5 Situation Type 5 Contract for a configured solution

6 4 7 Type 6 Situation Custom-built in-house solution

Like the custom-built contract situation, this type of situation is not as common as it used to be, but most organisations still develop at least some of their own solutions in-house In these cases, the concerns arising from the **project** are similar to other solution-oriented situations, and are mainly **goals** and **objectives**, rather than **priorities** or **deadlines**

The **users** and, in particular, individual **user requirements** are extremely important concerns in this situation, compared to the contract situations, because of the flexibility and lack of formality in the situation **Business requirements** and normally **information requirements** are important also

"First of all you have to identify a business requirement "

Although different practitioners tended to use this term in different senses, I am using it in the specific sense of requirements that support the specific **business needs** and **business opportunities** of the organisation Also, as one practitioner put it

"What I'm very concerned about is the distinction between the requirements and the functional analysis, the functional specification That is, the requirement is, what do you want to do from a business sense, and the functional requirement is, this is how I'm going to do it "

One reason why a business might decide to develop its own solutions, rather than rely on a standard package, is to try to gain a competitive advantage over its competitors

> "There is a question of course, which has been asked, which is How do I get an edge if all my competitors are using SAP?"

Whether or not a standard package is used, a company can increase its revenue, cut its costs, or improve its service by using information, in making business decisions, in ways that are not available to its competitors That is one reason why **non-standard requirements** very often involve **information requirements**, or **reporting**

> "It captures a lot of raw data and it does what a standard package does Now what they wanted to get from that system some reporting to management and It was a stats reporting system was how it was described "

> "In any billing system and in ours particularly, where it's a new business, ad hoc responses to all parts of the organisation are important Engineering, say, we think we've got a dodgy plane, could you compare the number of calls we've had from this plane versus the other aircraft and fleet "

In an in-house development situation, the **client organisation** is that part of the organisation that requires and is going to use the system This is often associated with one or more **business areas**, which, as in other solution-oriented situations, are a key source of requirements

> "So the first thing that I would look for is a definition of the process as it exists today, in some case like it was here, there won't be a definition of how it is today but there needs to be a definition of what the process is that these requirements are supposed to help to realise "

The **existing system**, **changes** to it, and **organisational change** are often important sources of **issues** before **agreement** is reached

> " thinking about what are the business processes that are going on, who are the business users of this, how will this system potentially change the business organisation "

Although there is no formal contract between the developers and their clients, that is the **users** and, in particular, their managers, there may be a ritual **signoff** procedure once requirements have been agreed

> "We would write up the requirements document It was signed off by the user, signed off by the systems department, say possibly signed off by the operations people "

> "An exercise like that has to be done because you have to be able to say at the next stage 'well this is what you said you wanted' but you can't rely on it as being the final say, so it wasn't a contract

> "Well it was, in the sense that they signed it off and we could have treated it as a contract but if a manager comes back at the design stage and says 'you can't do this,' its no use going back and saying 'we have a contract' "

> Yes, there should be a signature because, like, we're in business People have to be serious about it, they have to be given a deadline, they have to commit to something and say 'yes, that is what I want' "

Despite this appearance of formality, which was found in many in-house situations, because there is no involvement with an external company, the degree of formality in the actual process of discovering and agreeing requirements with the users seems to vary quite considerably It often depends on the ethos of the company For example in one particular business I visited, which is extremely results-oriented, the users were perceived by the developers as impatient to be getting on with their own work and not keen to get involved in the work of requirements definition

> "They wouldn't themselves want to put a lot of effort into clarifying the requirements but from my perspective if you set out to nail somebody down on their requirements you wouldn't get a lot of sympathy from the person you were trying to nail down or much understanding "

In other organisations doing in-house development, the requirements process tended to be more formal, more structured, or rely more on participation by users formally assigned to the project

> "Hence, a lot of the things that we dealt with and how AEP arose was through just meetings Very little documentation "

> "I believe that one of the reasons that worked was the users knew exactly what they were getting right from the start "

> " this should tell the user everything he or she wants to know about the system, everything So once the system is plopped on board three months later or three years later, there are absolutely no surprises in terms of processing, in terms of reports, in terms of screens, in terms of validation, in terms of inputs, in terms of anything "

Compared to other types of situations, the reliance on requirements documentation varied quite a lot between different in-house solution situations It depended on factors such as the background and training of the analyst, or the system development method they were using

> "The methodology that I am experienced in is Method/1 which is the Andersen Consulting Methodology which has been used extensively world-wide for over twenty years It was redesigned for the client server model and therefore I was working with that and that is the sort of experience I brought in to the requirements phase in this project "

Despite appearances, the use or invocation of a specific method does not indicate the degree of formality in the requirements process to any great extent This is because developers tend to rely on methods, more as toolboxes of techniques to be used as appropriate, rather than as sources of defined processes to be strictly adhered to This quotation reflects a trend that has been well documented in the literature on system development methods

> "I tend to look at methodologies in terms of the tools and techniques that you use, rather than the stages you go through I find the tools and techniques more interesting, of more practical use, anyway, than the stages "

Areas of Concern	Type 6 Custom-built In-house solution
Project	The Goals and Objectives of the Project are more important than the Scope, or Deadlines or Priorities There should be a Shared Vision of the Project Objectives Signoff may be used although there is no contract
Users	Individual User Requirements are much more influential than in other situations But the level of User participation in Projects varies considerably
Client Organisation	The Client Organisation is that part of the larger organisation that requires and is going to use the system This is often associated with one or more Business Areas
Business Area	Business requirements are a major concern The Existing System, changes to it, and Organisational Change are all important sources of Issues to be dealt with
Application Domain	Non-standard requirements, particularly Information Requirements are often key
Product	
Customer	There may be a Project Sponsor who takes on the role of Customer
Market Opportunity	

Table 6 6 Situation Type 6 Custom in-house solution

6 4 8 Concluding remarks on solution-oriented situations

"Signing-off" on the agreed requirements is a strategy commonly used in different types of solution-oriented situations It seems to have at least three different purposes for the practitioners

1 It protects the supplier's interests in the event of a disagreement about the outcome of a project, as to whether the solution implemented fulfils the client's needs If the client has signed a document, then the supplier can point to the document to prove that the contract has been fulfilled

2 It also protects the supplier in situations where there is a fixed price contract, by providing a baseline of agreed requirements, and any changes to this may incur extra charges, at the discretion of the supplier

> "we, as a company, would have to turn around and say well I'm sorry, it's all written down, this is where you signed it, do you want to talk about changes, let's talk about them I mean you prefer not to get into that situation "

3 In any solution-oriented situation, it encourages the client personnel who will be responsible for signing the document to ensure that the users actively participate in reading, reviewing, discussing and correcting the requirements before they are agreed and signed off For this reason, the strategy is used as much in in-house development as in situations where there is a contract with an external supplier

The application of methods and techniques in system development seems to vary among the different types of solution-oriented situations In a contract situation, it is important to have some defined procedures to control the risk to the developer/provider company A strategy used by one of the contractor companies among my interviewees was to draw up what was called a quality plan at the outset of each project This outlined the process that would be used in the project, including the amount and the types of documentation to be produced This was tailored for the specific situation, and involved a level of formality that was judged to be appropriate to the level of risk in the project and the ethos of the client company

Solution providers involved in configuring SAP for clients use a proprietary method called ASAP, which is geared to implementing that product in the shortest possible timescale In-house developers, on the other hand, seem to exhibit much greater variation in the methods they use and the level of formality adopted in projects

6 4 9 Type 7 Problem-oriented situation

> "Then we went through a stage of looking for solutions to the project, we investigated a number of packages, so the requirements (document) was circulated It was used to evaluate the various proposals that came in "

Many organisations lack the technical resources to develop all their own software, and for this or other reasons, may have occasion to seek solutions outside their own organisations In such cases, they often produce what is called a Request for Proposals (RFP), in which the problem is documented, and this document is sent to interested solution providers, who may respond with a proposal, either to supply a ready-made application package, or to customise a solution

In this situation, the **customer** is the author or originator of the requirements For the customer organisation, this stage is often a **project** in its own right There is considerable emphasis on the **goals** and **objectives of the organisation** or that part of it that owns the problem The **scope** of the problem, and of the desired solution, is another important concern, as are the necessary interfaces to other **existing** or planned **systems**

Because it arises in an organisational situation, the **users** and the client **organisation** are important determinants of the requirements

Depending on the type of organisation, either the **business area** or the **application domain**, or both, are important areas of concern Related concerns include the **functional areas**, and/or the **business processes** of the organisation The significance of these concerns is similar to their significance in solution-oriented situations, but there are a number of concerns that have a different emphasis in this type of situation

For example, there is a different significance in the matter of **agreement** Because there is no contract yet in existence, agreement with an external party (or between internal parties) is not a matter of concern The situation is, in fact, similar to some of the market-oriented situations in so far as there is often a concern with getting **approval** of the requirements statement by a steering committee, or other authority, before the document can be circulated outside the organisation

In some situations, for example where a public enterprise is seeking products or services, the document may have some legal status Many public bodies are required by law to source products or services without favouring any supplier, and therefore must be careful to state their requirements in such a way that they can conduct a tendering process that is open, fair and transparent

There is very little of concern derived from any particular **product**, or **market opportunity** in this type of situation, although awareness of the features of packaged solutions may be of significance The main distinguishing facet of this situation, compared to solution-oriented situations, is that any consideration of the **solution** is much more open in this type of situation

In producing a RFP, it is not customary to prioritise requirements Labelling some requirements as mandatory rather than optional is sometimes used, though the use of this strategy may constrain the choice during the selection process, because requirements designated as mandatory or optional must remain as such during the selection process

It may be noted that the problem-oriented and solution-oriented situations are reasonably similar in this framework, in that they share many similar concerns Both are primarily driven by the needs and **organisational objectives** of the **client organisation** However, in problem-oriented situations, a requirements document is created by the customer, prior to the selection of a solution provider and the establishment of a contract

Several solution providers in the investigation noted that in their experience most RFPs were not detailed enough or were otherwise unsuitable to be used in solution-oriented situations This supports the notion that these are different situations However, further evidence was found ın cases where practıtioners adopted a professıonal approach to the productıon of RFPs, and where there was organısatıonal support for such an approach

Areas of Concern	Type 7 Problem-orıented sıtuatıons
Project	The Scope of the Project ıs a central concern, as are the Goals and Objectıves Interfaces to other systems may be key Approval of the Project withın the Organısatıon hıerarchy ıs an ımportant concern
Users	Users are always ınvolved to some extent Project Stakeholders often ınclude Users and theır managers
Clıent Organısatıon	Organısatıonal objectıves are extremely ımportant Organısatıonal Change may be a source of Issues to be resolved
Busıness Area	The relevant Busıness Area(s) are ımportant sources, as are Busıness Processes, Functıonal areas and Workflow ın the Exıstıng System
Applıcatıon Domaın	Applıcatıon Problems and Domaın Knowledge may be ımportant sources of requırements
Product	Not relevant unless a Product-based Solutıon ıs beıng assumed
Customer	Thıs ıs the only sıtuatıon ın whıch the Customer ıs the creator of the requırements document
Market Opportunıty	Not relevant ın problem-orıented sıtuatıons

Table 6 7 Sıtuatıon Type 7 Problem-orıented sıtuatıon

6 5 Summary and Conclusıons

Thıs chapter has presented a scheme for classıfyıng system development sıtuatıons, dıstınguıshıng dıfferent sıtuatıons from the perspectıve of system requırements The classıficatıon scheme makes partıcular use of the root concerns ıdentıfied ın the prevıous chapter whıch argued that these root concerns are the sources of requırements, and gıve rıse to substantıve requırements, whıch ın turn become documented requırements Thıs chapter was based on the ınference from thıs that these concerns could be used as the basıs of a faceted classıficatıon to dıstınguısh the sıtuatıons

The analysıs whıch led to the fındıngs presented used the eıght root concerns as the crıterıa of a search for typıcal sıtuatıons ın the empırıcal data that could be delıneated ın terms of these root concerns The ıdea was to find typıcal sıtuatıons and descrıbe theır typıcal profiles ın these terms Much of thıs work was done usıng tables for comparıson, to hıghlıght where the sımılarıtıes and dıfferences occurred

Thıs analysıs resulted ın **seven** profiles of dıfferent typıcal sıtuatıons The fındıngs were presented ın terms of these profiles Each profile was presented ın two ways

1 As a descrıptıve profile of a sıtuatıon, ıllustrated wıth supportıng quotatıons from the ınterviews

2 As a tabular profile, a set of propositions asserting the tendency of that type of situation to have certain types of concerns

Since it is based on the cases investigated empirically, the classification scheme therefore identifies some gaps in the range of types of system/software development situations that were investigated This should be viewed as a strength rather than a weakness, since it allows for growth

How successful is this classification? One way to judge it is to ask how clearly it differentiates one type of situation from another This might be done by looking at the summary table comparing the different types of situations Table 6 8 combines the seven situations showing how each area of concern varies from one type of situation to another (This table is split over two pages, but the combined table may be found in Appendix I) Between the group of Market-oriented situations and the group of Solution-oriented situations, different concerns are more important than others For example, Market Opportunity is not a concern for Solution-oriented situations, and Business area is not a major concern in Market-oriented situations However, Market Opportunity plays different roles in each of the different Market-oriented situations The Customer as an area of concern and a source of requirements varies in importance and identity across the table Similarly, the table shows how the influence of Users on requirements varies from the Type 1 situation to the Type 4 situation, and contrasts these to the Type 6 and Type 7 situations

Another way to test the effectiveness of the classification is to examine the relationship between the different situations it has identified and the diversity of styles of requirements documentation that are found in practice This is the focus of the next chapter, which analyses each situation in terms of the different aspects of requirements documentation it entails

Areas of Concern	Type 1 Product for a vertical market	Type 2 Product for targeted customers	Type 3 Product for a single customer	Type 4 Contract for a custom built solution
Project	Product requirements are allocated to Projects which are related to Releases Priorities for the next release of the product are influential	The requirements are subject to informal Agreement with the target customer and subject to formal Approval by the internal Client Organisation either at a higher level or peer level	There is a formal agreement with the Customer represented on the Project by a Product Manager There is an Agreed deadline and little scope for selecting requirements based on priority	The Scope of the project must be managed and controlled There is an agreed Deadline and milestones
Users	Focus groups are often used in the discovery of user requirements	Key users from the customer side may help to define and/or validate requirements But it is difficult to get users involved	The role of user representatives is played by developers not real users	Individual User Requirements are less influential than the Organisational requirements
Client Organisation	Different levels for Product Planning Project planning	A combination of an external client and internal organisation Often seen as joint development	The client organisation is the external Customer	Negotiating between the Client Organisation and the supplier organisation is significant Client Organisation IS/IT personnel may also be influential
Business Area	Not relevant unless it is the Application Domain If so Functional areas and Business knowledge are important	A source of Common requirements for the Product		The Existing System may be a source of knowledge about the Business Processes and Workflow The Solution is seen in terms of Functions
Application Domain	Knowledge of Regulations or codes of practice in the target market is an important source of Common requirements	Understanding of the application domain e g relevant Standards etc The Product is state of the art in that application domain	Domains other than business areas A deep understanding of the application domain is an important source of requirements	Non standard requirements may be significant Application expertise on the part of the supplier may be useful
Product	Each release contains selected features so the scope of the project is much less than the scope of the product	Features of the Product may interact with each other requiring a high level of product expertise	Product functionality is understood in terms of its behaviour how it responds to events and its behavioural constraints	
Customer	There are many customers but not on the scale of mass-market situations	Few customers compared to vertical market situation Some of them may have considerable application expertise Customer representatives validate requirements	The external Customer may in turn have many customers typically consumers	The Customer is the Client Organisation
Market Opportunity	Products evolve in Versions in response to market opportunities Bundles of features are developed to address different market segments	Additional market requirements help to add value to the product Time to market is critical Additional platforms may be another source of system (technical) requirements	The project is not usually concerned with other opportunities except as a source of new contracts	This is not a concern in solution-oriented situations

Table 6 8a Summary table comparing the seven Situation Types

Type 5 Contract for a configured solution	Type 6 Custom built In-house solution	Type 7 Problem orıented sıtuatıons	Areas of Concern
The Scope of the project ıs very ımportant Scope creep ıs a sıgnıficant rısk Speed of ımplementatıon ıs ımportant so there ıs an agreed Deadlıne	The Goals and Objectıves of the Project are more ımportant than the Scope or Deadlınes or Prıorıtıes There should be a Shared Vısıon of the Project Objectıves Sıgnoff may be used although there ıs no contract	The Scope of the Project ıs a central concern as are the Goals and Objectıves Interfaces to other systems may be key Approval of the Project wıthın the Organısatıon hıerarchy ıs an ımportant concern	Project
Indıvıdual User Requırements are much less ınfluentıal than the Organısatıonal requırements Representatıve Users take part ın workshops	Indıvıdual User Requırements are much more ınfluentıal than ın other sıtuatıons But the level of User partıcıpatıon ın Projects varıes consıderably	Users are always ınvolved to some extent Project Stakeholders often ınclude Users and theır managers	Users
Organısatıonal objectıves domınate the requırements for example ıntegratıon of Busıness Areas Issues often relate to mısmatches between the Exıstıng System and the Product functıonalıty	The Clıent Organısatıon ıs that part of the larger organısatıon that requıres and ıs goıng to use the system Thıs ıs often assocıated wıth one or more Busıness Areas	Organısatıonal objectıves are extremely ımportant Organısatıonal Change may be a source of Issues to be resolved	Clıent Organısatıon
Busıness Process engıneerıng of the target Busıness Area(s) ıs key Some Busıness Processes may need to change (BPR) ın order to achıeve Organısatıonal Objectıves	Busıness requırements are a major concern The Exıstıng System changes to ıt and Organısatıonal Change are all ımportant sources of Issues to be dealt wıth	The relevant Busıness Area(s) are ımportant sources as are Busıness Processes Functıonal areas and Workflow ın the Exıstıng System	Busıness Area
Knowledge of the Applıcatıon Domaın ıs embodıed ın the Product ın terms of data types and avaılable functıons	Non-standard requırements partıcularly Informatıon Requırements are often key	Applıcatıon Problems and Domaın Knowledge may be ımportant sources of requırements	Applıcatıon Domaın
Product Expertıse on the part of the supplıer ıs an ımportant source of Solutıons The Solutıon ıs seen as Functıons provıded by the Product		Not relevant unless a Product-based Solutıon ıs beıng assumed	Product
The Customer ıs the Clıent Organısatıon	There may be a Project Sponsor who takes on the role of Customer	Thıs ıs the only sıtuatıon ın whıch the Customer ıs the creator of the requırements document	Customer
Thıs ıs not a concern ın solutıon-orıented sıtuatıons	Thıs ıs not a concern ın solutıon orıented sıtuatıons	Thıs ıs not a concern ın problem orıented sıtuatıons	Market Opportunıty

Table 6 8b Summary table comparıng the seven Sıtuatıon Types

Chapter 7 A Situation-related Classification Scheme for Requirements Documentation

7 1 Introduction

This chapter presents the fourth and final part of the proposed framework for requirements situations and documentation, a classification scheme which links seven typical requirements situations with the styles of requirements statements that they entail It explains seven different styles of requirements documentation in terms of their situation of use

Each of the seven typical situations that were presented in the previous chapter is associated in this chapter with a typical documentation profile Each profile outlines the role, in that type of situation, of the different types of requirements statements identified in Chapter 4 The relevance and importance of each type of requirement is shown to vary from one type of situation to another The different situations considered are

1 for a Vertical Market product
2 for a Targeted Customers product
3 for a Product for a single Customer
4 for a contract custom-built Solution
5 for a contract configured Solution
6 for an in-house custom Solution, and
7 a Problem-oriented (solution-seeking) situation

These situations may be characterised by their varying need for and use of the following different kinds of statements in the requirements document

1 Problem domain descriptions,
2 Statements of the required effects,
3 Proposed solutions,
4 Recorded Issues and Changes,
5 Defined Goals and Objectives,
6 Specified Constraints,
7 Recorded Agreement

These are the facets of the faceted classification scheme which was the aim of the final part of my grounded theory analysis, i e selective coding This stage is generally concerned with finding relationships between the 'core categories' that have emerged from the empirical data My original conception of the faceted scheme was a single table containing a row for each type of situation and columns for each facet, representing the different types of requirements as text, such as Problem Domain Descriptions (See Figure 11)

	Facet1	Facet2	Facet3	
Situation Type 1	Code1	Code3		
Situation Type 2	Code2			

Figure 11 The layout of the faceted classification as one big table

The process I followed in building the table was as follows For each type of situation, I tried to pinpoint the relevance of each facet (different type of requirements statement) to that situation with a short code in the appropriate cell of the table The idea was that the code would link that situation and the particular facet of requirements statements in a concise way For example, I had found that the code **Priorities** was an important concern in the Vertical Market situation, and also that Statements of Required Effects are usually prioritised in the requirements document in this situation, so I used this code to represent the link between these in the table Another example was the code **System Architecture** which was strongly identified with the facet **Proposed Solutions** in some situations, but not mentioned at all in situations

Each code used in the table was linked in the data both to the situation and to that facet of requirements documentation However, many of the distinctions I was making between situations could not be reduced to a code, but needed a more explanatory distinction in the form of a proposition (see Table 1, below, for examples) and therefore, the one big table, while it played a role in developing the grounded theory, was almost useless as a means of presenting it Either the codes were too concise to convey any meaning, or the propositions in the cells in the table made it too big to be manageable

The final part of the grounded theory is therefore presented here as a set of seven profiles representing different styles of requirements documentation Each profile is summarised in tabular form, as a set of short propositions asserting the relative propensity of that type of situation to entail a particular set of types of requirements statements Each profile is introduced in terms of the uses of the document in that situation, and the discussion is illustrated with some quotations from the interviews

7 2 Requirements Documentation for a Vertical Market Situation

In a Vertical Market situation, the requirements document is mainly used **as a working document** and **as a method of communication** between the project stakeholders The stakeholders typically act as reviewers of the document, taking part in meetings and discussions, commenting on its contents, and suggesting changes to be made

Another role of the requirements document in these situations is to provide input to various downstream activities, such as input to project planning, to estimating the cost of development, to the allocation of project resources, and it is often used as an input to the testing phase of the development For example

> "It was used as a basis for high level design, but not in the detailed design phase or the programming phase, for system test, this was used as a source of the system test conditions "

In a Vertical Market situation, there is very often a well-defined system development process, in which design is treated as a separate phase, and in which the requirements document is an important input to the design phase Because of its role as a working document, it is more likely to be used to help achieve agreement, rather than to record agreement, and is not likely to be used again, once the succeeding phases of the project have begun

Each requirements document in a Vertical Market situation is associated with a specific project, which typically only deals with a small proportion of the overall functionality of the product Both the problem domain, and the product in general, are much bigger in scope of the project and its requirements

Descriptions of the problem domain tend not to occur in the requirements document The problem domain is taken to be well understood by people in the industry, according to my informants, and if not, it is described separately For example, in the ERP market, Problem Domain Descriptions are documented separately in widely used generic reference models, which can be adapted for specific industries (Scheer, 1998) One of my informants explained the process used in his company whereby once the functionality has been determined for a project, usage scenarios are then described separately in use cases These are not included or referred to in the requirements document

The list of features is the most notable characteristic of the requirements document in this type of situation Statements of required effects are often organised around features, with priority numbers, interdependencies, and other attributes Priorities of features in effect determine the scope of the project, along with time constraints The priority of an individual feature is based on relative estimates of development cost and market value

Proposed Solutions, goals, even non-functional requirements, or constraints do not occur in the requirements documents of the situations I encountered that correspond to this type of situation

Issues, in so far as they occur in these situations, do not refer to organisational concerns, but may concern the priority of features, and therefore lead to changes in the priorities of features Such changes are not traced explicitly in the documents, and are not recorded explicitly, except to the extent that they affect the scope of the project

The agreement that is recorded in the document is essentially **Approval** The document for each project must be approved by the relevant levels of the organisation before the next stage of the process is begun The **Approval** given applies to the list of features included in the project and their attributes, such as **Priority**, etc The document typically contains a list of the reviewers, i e those who may only comment on its contents, and a list of the people who have the authority to approve it on behalf of the organisation The date on which the **Approval** was granted is an important element of the text

Style 1	**Requirements Document for a Vertical Market**
Problem domain descriptions	Descriptions of the problem domain tend not to occur in the requirements document
Statements of the required effects	These statements are often organised around features, with priority numbers, interdependencies, and other attributes
Proposed solutions	These do not commonly occur in the requirements documentation of vertical market products
Recorded issues and changes	Issues may concern the priority of features, and lead to changes
Defined goals and objectives	Neither organisational goals nor system goals, such as non-functional requirements, seem to occur in the requirements document in this type of situation
Specified constraints	Neither solution constraints nor application constraints tend to appear in the requirements document
Recorded agreement	Agreement is in the form of Approval by the relevant levels of the organisation The Approval given applies to the list of features to be included in the project, and their attributes

Table 7 1 Requirements Document for a Vertical Market

7 3 Requirements Documentation for a Target Customer Market

Market-oriented software is often equated with mass-market situations, that is, millions of customers, but in some markets, there are only a few very large customers The software products they buy may be deployed widely in equipment serving millions of users, but the customers who commission or buy the software are small in number Examples of this type of situation were given to me by several practitioners who work in the telecommunications software market Previously, this market was strictly regulated, and most operators bought their hardware and software as a bundle from a single supplier Suppliers had at most one customer in each country Even in the now deregulated market, there are at most only dozens of customers, and almost as many suppliers chasing their business Another example of this type of market I came across in the interviews was the utilities market Software suppliers whose customers are large utilities, such as power or energy companies, work in a very tight commercial situation As a result, they often develop their products as a joint effort with target customers

In this situation, similarly to the Vertical Market, the requirements document is mainly used **as a method of communication** between the customer and the developers, and also **as a working document** within the developer organisation During its early versions, it may be used as a discussion document, prompting developers and other stakeholders to suggest features to be included in the project It is also a means of getting the views of the target customer's personnel on the market value of different features

> "We suggested to them that they categorised requirements in three ways and we used that same categorisation back to them so there was an identification of what the business need of the requirement was Was it essential, was it desirable or was it considered nice to have A priority on it was either high, medium or low A stability on it was is it likely to change, is it unlikely to change or is it an element that needed prototyping "

Similarly to the situation with vertical market software, in this situation, the final version of the requirements document is used as an input to project planning, to estimating the cost of development, and to the allocation of project resources It may be used as the basis for prototyping the emerging product, and is often used as an input to the testing of the code

Unlike the vertical market situation, however, the Product and the Project have the same scope in this situation The product typically involves new technology, and the requirements document often contains elements of the solution, in that it forms the basis of the system architecture

> "So then basically we have the overview of the technology here and then we went into what we viewed as the solution We're talking architecture here '

The problem domain in this type of situation is usually complex It is not feasible to describe it in the requirements document Some aspects of the application domain may be already described, for example in standards documents, in glossaries of acronyms, etc Practitioners reported that they often create conceptual models of the problem domain in order to increase their own understanding of parts of it These would not be included in the document

The statements of the required effects are often organised around features Some of these may be standard features taken from the relevant standard, particularly in the telecommunications field Other features may be unique to the product or company A single feature in a complex telecommunications product may have hundreds of dependent requirements associated with it Known interactions between features are described

Issues in a target market situation are related to specific features or lower-level requirements, and might be concerned with adherence to standards Changes are frequent, as the document goes through several versions, which are managed at document level by version control They are not typically recorded within the document

Unlike the vertical market situation, which deals with a project rather than a product within the scope of a requirements document, the target market situation is normally concerned with a new product System goals, such as non-functional requirements, are important in this type of situation, and therefore more likely to be documented along with the other requirements Similarly, application constraints are significant in the documentation for this type of situation

Like the vertical market situation, the role of the document in getting to agreement is more important than its role in recording that agreement, because it does not have the status of a contract with the customer(s) Agreement takes place at two levels in this situation, **agreement** with the customer(s) and **approval** within the supplier organisation Agreement with the targeted customer(s) tends to be informal, such as a handshake As in the vertical market situation, the document has a set of reviewers, who may only comment on its contents, as well as a set of people who have the authority to formally approve it on behalf of the organisation This internal approval is part of a defined process, and applies to the list of features to be included in the first release of the product, and their dependent requirements

Style 2	Requırements Document for a Target Customer Market
Problem domaın descrıptıons	The problem domaın ıs complex It ıs not typıcally descrıbed ın the requırements document Some aspects may be already descrıbed ın standards documents, or glossarıes of acronyms
Statements of the requıred effects	These are often organısed around features, eıther standard features or features that are unıque to the product or company Interactıons between features are also stated
Proposed solutıons	May ınclude the proposed archıtecture of the product
Recorded ıssues and changes	Issues relate to specıfıc features or lower-level requırements, and mıght be concerned wıth adherence to standards
Defıned goals and objectıves	System goals such as non-functıonal requırements are ımportant ın thıs type of document
Specıfıed constraınts	Applıcatıon constraınts, such as behavıoural constraınts, are sıgnıfıcant
Recorded agreement	Agreement takes place at two levels Informal Agreement wıth the target customer(s) and formal ınternal Approval

Table 7 2 Requırements Document for a Target Customer Market

7 4 Requırements Documentatıon for a Sıngle Customer Product

In thıs sıtuatıon, a software 'product' ıs commıssıoned by a sıngle customer, and developed under a contract between the developers and the clıent organısatıon Although the complete product wıll eventually be bought and used by many users, possıbly ın a mass market, the software part of the product ıs sold to a sıngle customer The typıcal software product ın thıs case ıs embedded software, for example, the software used ın electronıc consumer goods Another example ıs the control system used ın ındustrıal equıpment The commıssıonıng customer pays for the software, and then embeds ıt on a chıp ın the equıpment that they manufacture The equıpment ıs then dıstrıbuted to be sold to several other customers, many of whom wıll be end users ın the case of consumer electronıcs It ıs ımportant that the software performs correctly before ıt ıs embedded ın the equıpment, otherwıse the commıssıonıng customer wıll lose a consıderable amount of money, not only due to lost market share, but also ın the consequent waste of manufacturıng resources and materıals

The maın role of the requırements document, ın thıs sıtuatıon, ıs to **document the agreement** between the developer and the clıent It ıs also used to help achıeve that agreement, ın the course of meetıngs and dıscussıons, and as a means of communıcatıon between people workıng for two separate organısatıons It allows them to artıculate theır knowledge of, and express theır understandıng of, the envıronment ın whıch the software ıs requıred to work

Descrıptıons of thıs envıronment frequently appear ın the ıntroductory chapter, rather than throughout the document The problem domaın descrıptıons often refer to the equıpment ın whıch the software ıs embedded, as well as to the sıtuatıon of use of the equıpment The physical behaviour of the embedding equipment, described in terms of events, as well as user actions are important aspects of the problem domain that need to be described

Practitioners in this area regarded the joint construction of the requirements document as a learning process They also emphasised the point that a fairly deep understanding of the intended behaviour of the product, and of the application domain, was an important concern for them

An essential part of this understanding is a thorough grasp of the constraints that dominate the intended behaviour of the software, both from the operation of the user interface, and from the layer underneath that controls the hardware in which the software is embedded

> "So we have a constraints section – which was the bit you were looking at originally - which looks at each variable and looks at each of its particular states and then see what constraints it puts on the rest of the system "

> "Well, once you come up with all the events and all the variables, then you can come up with a table of constraints, where you look at each variable in turn and see how it constrains the system "

Because it is a learning process, it happens that some aspects of requirements become clearer later in the process Changes to rectify the omission, or misunderstanding, must be agreed, but once a certain point has passed, these changes can cause technical problems In these situations, change requests are treated differently from those that occur in contract situations for organisational systems, in that they are not charged for It is more important that the product should work correctly

> "It's either possible or not possible So, if we can fit it in, we'll do it If it's going to take six weeks to implement it, that means we'd miss the start of beta testing, so you know It has fixed start of production, so, in general, they try not to miss that date "

Issues and changes are extremely important in this situation, because firstly, it is a contract, and secondly, because changes cannot be easily made to the software once the equipment is in production Up to a certain version of the document, changes will be made without traceability, after which point most changes will be recorded with reasons for the change

Because of the importance of correct functioning of the final product, testing is an extremely important part of the software development process in this situation

> "You could argue that the first phase of alpha testing is integration testing You put everything together and you try to get it working before you start using formalised alpha testing plans or procedures This project was 11 weeks in alpha testing and then about 8 weeks in beta test "

The behavioural requirements defined in the requirements document play an important role in testing These denote features, or behavioural requirements, referring to the interaction or behaviour of the software in its environment Behavioural requirements include statements about states, events, transitions, etc The more accurately and completely they are expressed, the more straightforward the testing

"You can take each event and see how the system should react to that event "

System goals such as non-functional requirements are important in this type of document Both behavioural constraints and solution constraints are specified Solution constraints are important, relating to the platform, the capability of the embedding hardware, etc Behavioural constraints may be modelled along with the behavioural requirements

The agreement is a formal **contract** between two organisations, the commissioning customer and the supplier/developer organisation The requirements document is a record of the agreement between the client's product manager and the developer organisation's project manager on the required behaviour of the software product in the specified environment

Style 3	Requirements Document for a Single Customer Product
Problem domain descriptions	These descriptions frequently appear in the introductory chapter, rather than throughout the document The problem domain often refers to the equipment in which the software is embedded, as well as to the situation of use
Statements of the required effects	These denote features, or behavioural requirements, referring to the interaction/behaviour of the software in its environment
Proposed solutions	A proposed or actual system architecture may be defined in the requirements document
Recorded issues and changes	These are extremely important because it is a contract, and because changes cannot be made when the equipment is in production
Defined goals and objectives	System goals such as non-functional requirements are important in this type of situation
Specified constraints	Both behavioural constraints and solution constraints are specified
Recorded agreement	The agreement is a formal contract between the customer and the supplier

Table 7 3 Requirements Document for a Single Customer Product

7 5 Requirements Documentation for a Contract Custom-built Solution

Solution-oriented problem domains are concrete examples or instances, unlike their counterparts in market-oriented development Only the specifics of the problem domain need to be described, rather than all the different possibilities that are likely to occur This makes it more practical to incorporate the problem domain description in the solution-oriented requirements document However, different types of solution-oriented situations entail different styles of problem domain description, because of the different uses of the requirements document in those situations

In any contract sıtuatıon, the maın purpose of the requırements document ıs **to record agreement** In a contract sıtuatıon ınvolvıng a custom-buılt solutıon, another key purpose ıs to support the process of gettıng that agreement, and to act as a dıscussıon document, to be the focus of project meetıngs between the clıent organısatıon and the contractor organısatıon

> "The purpose of thıs document ıs to document our agreement but at the end of the day ıt's not enough for ıt just to be ın the document The purpose of the document ıs to focus and structure the conversatıon when the actual agreement takes place "

The requırements document ıs also used ın these sıtuatıons as an ınput to project plannıng, partıcularly the allocatıon of resources to the project Practıtıoners ın thıs area also use the document as an ınput to estımatıng the development cost of the project

> "It contaıns the requırements and the plan for ımplementıng the requırements, the reason ıt was done lıke that ıs that thıs became a contractual document so ıt was more or less here ıs what we are goıng to do, here ıs the plan for doıng ıt, and here ıs what we are goıng to charge you "

For the developer organısatıon, ın order to keep costs wıthın the budget, ıt ıs ımportant to 'pın down the scope of the project' ın the document, and to get agreement on that scope at an early stage It ıs also vıtal to theır commercıal success to pın down the agreed functıonalıty whıch wıll be provıded ın the project It also helps to get agreement before movıng to the next stage of a project

> "Well, fırst of all, what the document ıs ıt's an ınterface between ourselves and the customer and prımarıly I suppose ıt's a contractual document ıf you lıke, that's not so much from the company's poınt of vıew but certaınly from mıne because ıt defınes my job, ıt defınes what I'm goıng to ımplement "

Problem domaın descrıptıons are a much more common aspect of solutıon-orıented requırements documents than ın product-orıented sıtuatıons Descrıptıons of busıness operatıons, procedures and functıons generally form the framework for other requırements statements, and are often used ın these sıtuatıons to organıse other statements throughout the document Estımated volumes or actual volumes of busıness transactıons and fıle sızes are also commonly stated

However, much of thıs ıs thınly spread throughout the document, the bulk of the requırements text consıstıng of statements that typıcally descrıbe transactıons and ınformatıon requırements These generally outweıgh the brıef descrıptıons of the problem domaın ın whıch they are embedded

Solutıon-orıented requırements documents often ınclude proposed solutıons, for example, the database structure and contents These are typıcally used by practıtıoners to pın down the scope of the project The document may include a description of a prototype solution, outlining the user interface, for example

Issues with the requirements are extremely important in this situation for two reasons Where a solution is being specified for organisational use, people will have issues with changes that affect them However, the client organisation's issues rather than the people's own will be uppermost in this case The second reason is that there is a contract involved Changes to the requirements have important consequences in any contract situation This is similar to the Product for a Single Customer situation Up to a certain version of the document, changes will be made without traceability, after which point most changes will be recorded along with the reasons for the change

Also, because it is an organisational situation, organisational goals may be regarded as significant However, the practitioners I interviewed in these situations differed widely on this Organisational goals may be expressed in terms of a **business case** for the system, and are often seen as more pertinent than system goals in these situations, although these higher-level requirements may be related to the stated organisational goals in the document

Both types of constraints appear in the documents for this type of situation Application constraints are typically represented in the database design, or separately, as business rules Solution constraints, such as software or hardware platforms or interfaces to other systems, may be defined in separate section

Like situation type 3, the agreement in this situation is a formal **contract** between two organisations, the customer (or client organisation) and the supplier (often called a solution provider) In particular, the requirements document is a record of the agreement reached between the client's project manager and the solution provider's project manager on the required functionality and scope of the solution The documented agreement may specifically refer to changes that were made to resolve issues that were raised in the course of discussions

Style 4	Requırements Document for a Contract Custom Solutıon
Problem domaın descrıptıons	Descrıptıons of busıness operatıons and functıons form the framework used to organıse other requırements statements
Statements of the requıred effects	Statements descrıbıng transactıons and ınformatıon requırements form the bulk of the requırements text
Proposed solutıons	Often ıncludes the database structure and contents or descrıptıon of a prototype solutıon, such as the user ınterface
Recorded ıssues and changes	Issues are extremely ımportant because ıt ıs an organısatıonal sıtuatıon Changes are central because there ıs a contract, and these are documented after a partıcular versıon of the document, wıth reasons for the change
Defıned goals and objectıves	Organısatıonal goals are more sıgnıfıcant than system goals and may be expressed ın terms of a busıness
Specıfıed constraınts	Applıcatıon constraınts may be represented ın the database desıgn, or separately, as busıness rules Solutıon constraınts such as ınterfaces to other systems may be defıned
Recorded agreement	The agreement ıs a formal contract between the customer (clıent organısatıon) and the supplıer (solutıon provıder) The document ıs a record of the agreement reached between them on the requıred functıonalıty and scope of the project

Table 7 4 Requırements Document for a Contract Custom-buılt Solutıon

7 6 Requırements Documentatıon for a Confıgured Solutıon

The fifth type of sıtuatıon ın the classıficatıon ıs the second type of solutıon-orıented sıtuatıon, concernıng a configured solutıon Thıs ıs another contract solutıon sıtuatıon, but the software product for the solutıon has already been selected, so the solutıon task ıs to fit (configure) the chosen solutıon to the problem sıtuatıon

The requırements document for a configured solutıon ıs normally part of the contract between the clıent organısatıon and the solutıon provıder In thıs sıtuatıon, the requırements document, ın addıtıon to **recordıng agreement**, ıs used also **as a workıng document** ın the process of matchıng the elements of the solutıon to the partıcular problem sıtuatıon and also ın the process of **gettıng agreement**

The scope of the project, rather than the detaıled functıonalıty, ıs more of a concern here than ın a custom-buılt contract sıtuatıon, because thıs ıs used to estımate the development cost and to plan the project It may also be used ın acceptance testıng of the configured solutıon

Problem domaın descrıptıons are an ımportant part of the configuratıon requırements process, but are not generally ıncluded ın the requırements contract These are extremely detaıled descrıptıons of the relevant busıness processes and workflow, based on an ındustry reference model of standard Busıness Process descrıptıons An ındustry reference model of standard Busıness Process descrıptıons ıs often used to guıde thıs The chosen product may necessitate some changes in business terminology used by the client organisation, so a glossary of business terms is often used in these situations

The purpose of the business process descriptions is to express the solution developers' understanding of the client organisation's particular operation The document can reuse standard process descriptions for those processes which are standard, and adapt the descriptions to document the variations

The statements of required effects dominate the requirements document Transactions and information requirements are written in terms of the chosen product, and are often closely tied to the proposed solution

The proposed solution is by definition a variation of the chosen product Specific proposed solutions are selected from the range of standard functionality and database formats which come with the chosen product, and to a lesser extent specified by tailored add-ons to cater for additional functionality that is not supported by the product

Issues in this type of situation are related to required effects which may not be directly supported by the product In such cases, compromises have to be made, either by tailoring a solution or by going with the functionality that is provided Changes or reasons for changes are not explicitly recorded in the main document In one case of this type of situation, a third type of document was used, recording the issues and changes along with the reasons for the change

Similarly to the situation with contract custom-built solutions, the business case for the system expresses organisational goals But these are typically associated with improved business practices supported by the functionality of the system System goals, such as non-functional requirements, are not normally expressed

Typically, the chosen product supports a wide range of many different application constraints, such as business rules Defining which ones are needed is an important aspect of defining the requirements Like the Business Process descriptions, however, these do not appear in the requirements document There is no perceived need to specify solution constraints, as the main solution constraint is already decided, implicit in the choice of product

The agreement ıs a formal contract between the clıent organısatıon and the solutıon provıder It ıs a contract to provıde the specıfied functıonalıty wıthın the constraınts of the chosen product The requırements document ıs a record of the agreement reached between the clıent project manager and the solutıon provıder project manager on the scope of the project

Style 5	Requırements Document for a Confıgured Solutıon
Problem domaın descrıptıons	Detaıled descrıptıons are made of the relevant Busıness Processes based on an ındustry reference model of standard Busıness Process descrıptıons, but do not appear ın the requırements document
Statements of the requıred effects	Transactıons and ınformatıon requırements wrıtten ın terms of the chosen product, and often closely tıed to the proposed solutıon
Proposed solutıons	Solutıons are selected from the range of standard functıonalıty and database formats whıch come wıth the chosen product
Recorded ıssues and changes	Issues are often related to requıred effects whıch may not be dırectly supported by the product These and the changes that ensue are recorded separately
Defıned goals and objectıves	The busıness case for the system expresses organısatıonal goals These are typıcally assocıated wıth ımproved busıness practıces supported by the functıonalıty of the system
Specıfied constraınts	Defınıng whıch of the varıety of applıcatıon constraınts supported by the product are needed ıs an ımportant task, but these are recorded separately, wıth the Busıness Processes
Recorded agreement	The agreement ıs a formal contract between the clıent organısatıon and the solutıon provıder, a record of the agreement on the scope of the project It ıs a contract to provıde the specıfied functıonalıty wıthın the constraınts of the chosen product

Table 7 5 Requırements Document for a Confıgured Solutıon

7 7 Requırements Documentatıon for an In-house Solutıon

In a solutıon-orıented ın-house sıtuatıon where the system ıs custom-buılt, the requırements document ıs used maınly to focus the dıscussıons wıth the users, ın order to get an agreement wıth them on the requıred functıonalıty for the system It may be used also as a dıscussıon document, as a method of communıcatıon wıth users, so that they know what they are gettıng It may also be used as the basıs of the user manual

> "Somethıng the users can understand, how they are goıng to be able to use the system Yes, ıt does have to be accessıble to the users I mean, ıt ıs goıng to be that thıck, there ıs goıng to be a good bıt of detaıl but there should be an ıntroductıon, ıntroductory chapters to get an ıdea of what they're goıng to get, ıt should have desıgns etc , screen desıgns partıcularly "

The ın-house sıtuatıons that I studıed, even where they use a system development method, rarely use a defined process for system development The technıques of the method tend to be used a la carte, and not the process that comes wıth the method In these sıtuatıons, I found that the requırements document has lıttle or no downstream role, for example, as ınput to design, or project planning, or even testing, as it would have in product-oriented situations

Problem domain descriptions often form the bulk of the requirements document in this type of solution-oriented situation These often focus on business processes They also serve to organise the document, with other types of requirements embedded in the descriptions

"It's simply a business requirements document and the sub-sections in that sort of document should be business related "

This focus on problem domain descriptions reflects the secondary purposes of the document, to articulate knowledge and to express understanding of the problem domain I found that in this type of situation there was a much greater use of modelling techniques in the requirements document than in any other situation Practitioners used these techniques, not so much to communicate with their clients, but to express their own understanding of the problem domain

A drawback of using modelling techniques for problem domain description is the risk of confusing the problem domain analysis model with the system design model, because the same diagrams can be used for both analysis and design, for example, data flow diagrams or class diagrams This risk is not a concern for practitioners in in-house solution situations, because they are not following a defined process that separates design from requirements, as happens in market-oriented situations

Statements of the required effects are typically transactions and information requirements, often embedded in representations of the problem domain Proposed solutions are typically also included, as database contents and/or references to elements which have been prototyped

Issues are important in this situation, typically arising from organisational changes, and from the users' concerns about changes to the existing system These issues may be noted in the document, along with the means to resolve them, but explicit statements about changes are not needed in this type of situation, because it is not a contract

The business case for the system is an important section of the document Organisational goals may be linked to high-level requirements, or specific business processes to be supported by the system But system goals such as technical requirements do not fit into the typical document used in this type of situation

"I wouldn't want to get into and I don't ever want to get into what our technical requirements are because it's all, at this stage, simply a business requirements document"

Solution constraints are unusual in this type of document Application constraints may be modelled in the problem domain descriptions, or separately, as business rules

The agreement is an informal agreement reached between the developers and their clients, i e the users and their managers The agreement may not even be in the document, which is often used in these situations only as a means of getting to agreement

Style 6	**Requirements Document for an In-house Solution**
Problem domain descriptions	These represent business processes and functions, user tasks and business events, and form the bulk of the requirements text
Statements of the required effects	Transactions and information requirements are stated, often in the context of representations of the problem domain
Proposed solutions	Database contents and/or database structure and references to prototypes are often included
Recorded issues and changes	Issues arising from organisational changes and from users' concerns about changes to the existing system may be noted
Defined goals and objectives	The business case for the system is important Organisational goals may be linked to high-level requirements System goals are not considered appropriate components of the document
Specified constraints	Solution constraints are unusual Application constraints may be modelled in the problem domain descriptions, or as business rules
Recorded agreement	The agreement an informal agreement reached between the developers and their clients The document may or may not be a record of it

Table 7 6 Requirements Document for an In-house Solution

7 8 Problem-oriented Requirements Documentation

In a problem-oriented situation, the requirements document, or Request for Proposals (RFP) is used, first of all, to focus and direct the process of **looking for solutions**, and then to evaluate the proposals that are put forward by the solution providers Once a proposal has been accepted, the RFP ceases to have much of a role, as the solution providers typically write their own solution-oriented document, so the RFP has no downstream role in design, prototyping, costing, planning or testing

However, it does have a significant role within the organisation in which it originates, as a discussion document, as a method of communication between different departments and levels of the organisation, whereby it supports the process of getting the members of the organisation to agree on what they want It is also used to document that agreement

The situation is similar to that of in-house solutions, in that the problem domain descriptions may use a variety of modelling techniques to represent business processes and functions, but essentially to outline the scope of the problem domain The company background, and volumes of transactions and records, may be important information in the document

The statements of required effects seen in problem-oriented requirements documents vary considerably from one case to another They may or may not be very detailed, but if detailed they typically include transactions and information requirements Non-standard requirements are often highlighted in these documents Proposed solutions are not relevant Although the authors may demonstrate an awareness of possible solutions, from the feasibility study, the purpose of the document is to elicit proposals, not propose solutions

Issues and changes may be recorded in the various drafts, but these are not allowed to appear in the final document, unless there are still some important open issues to be resolved by the solution provider The concept of a final document is this situation is much more significant than in many other situations, such as in-house solution situations

The problem-oriented requirements document tends to emphasise objectives One important section deals with the business objectives of the required solution Another section deals with the high-level non-functional requirements These are more significant in this situation than in the solution-oriented situations Solution constraints are also considered extremely valuable in the document for a problem-oriented situation However, lists of application constraints, if they appear, are unlikely to be complete, due to the nature of the requirements process used in this type of situation

The final version of the document is approved by the people in the organisation who have the authority to spend the funds allocated to the project There may be a formal steering committee for this function The approval given applies to the scope and the broad functionality of the required solution In particular, it applies to any constraints on the solution that are stated in the document

Style 7	Problem-oriented Requirements Document
Problem domain descriptions	Used mainly to outline the scope of the problem domain The company background, and volumes of transactions, etc
Statements of the required effects	May or may not be very detailed, but may have transactions, information requirements and non-standard requirements
Proposed solutions	These are not really applicable in a problem-oriented document
Recorded issues and changes	Not applicable in the final document, unless there are still important open issues to be resolved by the solution provider
Defined goals and objectives	The business objectives of the required solution and the high-level non-functional requirements are important sections
Specified constraints	Solution constraints are typically an extremely important section of the document
Recorded agreement	The final document is approved by the people in the organisation who have the authority to spend the funds allocated to the project

Table 7 7 Problem-oriented Requirements Document

7 9 Summary and Conclusions

This chapter has presented a classification scheme that tries to account for the differences between different styles of requirements documents by relating them to their situation of use It takes the seven typical situation types proposed in the previous chapter and analyses them on the basis of their usage of the seven basic elements of requirements documentation which were identified in Chapter 4 Based on this, a different style of document is identified with each situation type The different styles are explained in terms of the varying primary uses and roles of the documents themselves in the different situations

The findings of this chapter represent the conclusion of the grounded theory analysis This analysis was based on the original conjecture introduced in Chapter 1, that there is no one best way to document requirements, as it depends on the situation The aim of the research was to study the inherent variety of situations and the variety of requirements documentation used in practice and to find out what worked in different situations Since the literature provides no suitable way to classify situations from the perspective of requirements, and no suitable way of classifying requirements as texts, these were introduced in Chapters 6 and 4 respectively This chapter uses those findings to propose a set of profiles of requirements document styles that is related to their situation of use

The concluding remarks of the previous chapter suggested that the effectiveness of the classification of situations could be evaluated by asking how clearly it differentiates one type of situation from another A further assessment of those distinctions could be made by asking how clearly the documentation needs differ among those situations These dıfferences are to be found ın Table 7 8, whıch summarıses the seven documentatıon styles descrıbed ın thıs chapter Thıs shows patterns of varıatıon ın problem domaın descrıptıons, statements of requıred effects, and other elements, between the dıfferent styles of requırements documentatıon found to approprıate to dıfferent sıtuatıons The reader wıll notıce that space dıd not permıt the reıteratıon of the left-most column ın part b of the table, as was done ın the case of Table 6 8 Table 7 8 ıs reproduced ın Appendıx I

Styles	Requırements Document for a Vertıcal Market	Requırements Document for a Target Customer Market	Requırements Document for a Sıngle Customer Product
Problem domaın descrıptıons	Descrıptıons of the problem domaın tend not to occur ın the requırements document	The problem domaın ıs complex It ıs not typıcally descrıbed ın the requırements document Some aspects may be already descrıbed ın standards documents or glossarıes of acronyms	These descrıptıons frequently appear ın the ıntroductory chapter rather than throughout the document The problem domaın often refers to the equıpment ın whıch the software ıs embedded as well as to the sıtuatıon of use
Statements of the requıred effects	These statements are often organısed around features wıth prıorıty numbers ınterdependencıes and other attrıbutes	These are often organısed around features eıther standard features or features that are unıque to the product or company Interactıons between features are also stated	These denote features or behavıoural requırements referrıng to the ınteractıon/behavıour of the software ın ıts envıronment
Proposed solutıons	These do not commonly occur ın the requırements documentatıon of vertıcal market products	May ınclude the proposed archıtecture of the product	A proposed or actual system archıtecture may be defıned ın the requırements document
Recorded ıssues and changes	Issues may concern the prıorıty of features and lead to changes	Issues relate to specıfıc features or lower level requırements and mıght be concerned wıth adherence to standards	These are extremely ımportant because ıt ıs a contract and because changes cannot be made when the equıpment ıs ın productıon
Defıned goals and objectıves	Neıther organısatıonal goals nor system goals such as non-functıonal requırements seem to occur ın the requırements document ın thıs type of sıtuatıon	System goals such as non functıonal requırements are ımportant ın thıs type of document	System goals such as non functıonal requırements are ımportant ın thıs type of sıtuatıon
Specıfıed constraınts	Neıther solutıon constraınts nor applıcatıon constraınts tend to appear ın the requırements document	Applıcatıon constraınts such as behavıoural constraınts are sıgnıfıcant	Both behavıoural constraınts and solutıon constraınts are specıfıed
Recorded agreement	Agreement ıs ın the form of Approval by the relevant levels of the organısatıon The Approval gıven applıes to the lıst of features to be ıncluded ın the project and theır attrıbutes	Agreement takes place at two levels Informal Agreement wıth the target customer(s) and formal ınternal Approval	The agreement ıs a formal contract between the customer and the supplıer

Table 7 8a Comparıng Documentatıon Styles

Requırements Document for a Contract Custom Solutıon	Requırements Document for a Confıgured Solutıon	Requırements Document for an In house Solutıon	Problem orıented Requırements Document
Descrıptıons of busıness operatıons and functıons form the framework used to organıse other requırements statements	Detaıled descrıptıons are made of the relevant Busıness Processes based on an ındustry reference model of standard Busıness Process descrıptıons but do not appear ın the requırements document	These represent busıness processes and functıons user tasks and busıness events and form the bulk of the requırements text	Used maınly to outlıne the scope of the problem domaın The company background and volumes of transactıons etc
Statements descrıbıng transactıons and ınformatıon requırements form the bulk of the requırements text	Transactıons and ınformatıon requırements wrıtten ın terms of the chosen product and often closely tıed to the proposed solutıon	Transactıons and ınformatıon requırements are stated often ın the context of representatıons of the problem domaın	May or may not be very detaıled but may have transactıons ınformatıon requırements and non-standard requırements
Often ıncludes the database structure and contents or descrıptıon of a prototype solutıon such as the user ınterface	Solutıons are selected from the range of standard functıonalıty and database formats whıch come wıth the chosen product	Database contents and/or database structure and references to prototypes are often ıncluded	These are not really applıcable ın a problem orıented document
Issues are ımportant because ıt ıs an organısatıonal sıtuatıon Changes are central because there ıs a contract These are recorded after a partıcular versıon of the document wıth reasons for the change	Issues are often related to requıred effects whıch may not be dırectly supported by the product These and the changes that ensue are recorded separately	Issues arısıng from organısatıonal changes and from users concerns about changes to the exıstıng system may be noted	Not applıcable ın the fınal document unless there are stıll ımportant open ıssues to be resolved by the solutıon provıder
Organısatıonal goals are more sıgnıfıcant than system goals and may be expressed ın terms of a busıness	The busıness case for the system expresses organısatıonal goals These are typıcally assocıated wıth ımproved busıness practıces supported by the functıonalıty of the system	The busıness case for the system ıs ımportant Organısatıonal goals may be lınked to hıgh-level requırements System goals are not consıdered approprıate components of the document	The busıness objectıves of the requıred solutıon and the hıgh level non functıonal requırements are ımportant sectıons
Applıcatıon constraınts may be represented ın the database desıgn or separately as busıness rules Solutıon constraınts such as ınterfaces to other systems may be defıned	Defınıng whıch of the varıety of applıcatıon constraınts supported by the product are needed ıs an ımportant task but these are recorded separately wıth the Busıness Processes	Solutıon constraınts are unusual Applıcatıon constraınts may be modelled ın the problem domaın descrıptıons or as busıness rules	Solutıon constraınts are typıcally an extremely ımportant sectıon of the document
The agreement ıs a formal contract between the customer (clıent organısatıon) and the supplıer (solutıon provıder) The document ıs a record of the agreement reached between them on the requıred functıonalıty and scope of the project	The agreement ıs a formal contract between the clıent organısatıon and the solutıon provıder a record of the agreement on the scope of the project It ıs a contract to provıde the specıfıed functıonalıty wıthın the constraınts of the chosen product	The agreement an ınformal agreement reached between the developers and theır clıents The document may or may not be a record of ıt	The fınal document ıs approved by the people ın the organısatıon who have the authorıty to spend the funds allocated to the project

Table 7 8b Comparıng Documentatıon Styles

Part Three Theoretical Support for the Findings

A grounded theory begins with and continues to rely upon the body of empirical data which is the source of the concepts and relationships it contains The grounded theory approach is about discovering categories in the data and evolving a conceptual framework that explains those findings But it also relies on interpretation of the data, so the theoretical background and influences that the investigator brings to the analysis and interpretation of the data need to be made explicit Most of this background is embodied in the relevant literature on the topic, in this case, the literature on requirements engineering and information system development

The existing literature has many uses in grounded theory research, including

1 Stimulating theoretical sensitivity (the ability to give meaning to data)
2 As a secondary source of data
3 Stimulating questions
4 Directing the process of theoretical sampling, and
5 As supplementary validation of the findings (Strauss and Corbin, 1990)

In the context of grounded theory, a set of findings that merely confirmed and reflected the existing literature would be pointless There must be at least some extension of the existing theory, or some critique of it Since the related literature on requirements is so diverse, it is not surprising that my findings are supported by the some of the literature and at odds with the rest of it In the final part of the thesis, I compare and contrast my findings to the standard perspectives found in the current literature But first, in Part Three, I present a number of theoretical perspectives that have influenced, and therefore support, these findings

In Chapter 8, I review the main sources from which I have gained some understanding of the nature of organisational situations to give some theoretical support to the classification of situations presented in this thesis In order to achieve their goals, and get work done, organisations in different situations need to coordinate, control, and communicate the work that is going on These mechanisms both facilitate and encumber people who are working together, particularly those doing non-routine knowledge-based work, such as system development This chapter looks at system development situations in terms of their differing needs for coordination, control, and communication, and the implications of these differences for the task of requirements documentation

In Chapter 9, I present the theoretical perspectives on requirements representation that have influenced and contributed to the grounded theory of documentation presented in this thesis These include concepts and theories dealing with different forms of representation, including modelling, the different elements that need to be represented, the benefits of using models, and reasons why models are not always the best way of representing requirements

Chapter 8 Perspectives on the Organisational Context of Requirements and Documentation

8 1 Introduction

The framework presented in Part Two is centred around the idea of a situation It tries to identify the factors that can be used to distinguish system development situations with respect to their consequences for requirements documentation System development is an organisational activity, taking place in an organisational situation In this chapter, I discuss how the situations outlined in Chapter 6 may be viewed as diverse organisational situations, and review some established frameworks for understanding organisational activity that have appeared in the literature dealing with the theory of organisations in general, and of system development in particular

System developers are generally employed in and by organisations They work in groups, normally organised as project teams Almost all of the practitioners that I interviewed normally work on projects A project is a well-established way to organise a specific set of people to work towards a particular goal, over a specified time period The composition of a project team may change over time, and people other than system developers are often involved, either full-time or part-time, but the project organisation is the normal work setting of the system developer

As reflected in the findings, a system development project is concerned with problem solving Its goals are to first of all define the problem, to the satisfaction of all concerned, and then to find and implement a solution In order to do this the people concerned must communicate with each other, understand each other, and ultimately agree with each other about what the problem is, and whether the solution is acceptable But people, being human, suffer from the problem of bounded rationality (Davis, 1982, Newell and Simon, 1972)

Bounded rationality means that people are inclined to be selective in the things they pay attention to, because they cannot attend to an unlimited number of things at the same time Therefore, even when they try to behave rationally, they cannot do this perfectly well

Organisations are a response to this shortcoming of people acting alone Organisations introduce communication, shared information, and coordinated action to various kinds of human endeavours, including crime, religion, education, and business Furthermore, organisations often introduce routines and procedures for getting things done, which free up human attention for less routine activities and events (March and Simon, 1958)

But system development is, by and large, non-routine A new system is being developed, even if it is not unique, it has never been developed in exactly the way same before, in this particular organisation, by these people, on this hardware platform, using this language and tools, for these particular users System development is non-routine activity, carried out by organised groups of people who must communicate with each other, understand each other's points of view, and come to some kind of agreement, in order to learn about, define and ultimately solve what is normally an ill-structured problem

8 2 Contingency theory

One way to account for the variety of organisational situations is contingency theory Contingency theory says there is no one best way to manage or coordinate the activities of an organisation It depends on the situation The mam idea is that the internal features of a successful organisation depend to a large extent on the environment in which it exists, and hence external factors in that environment, especially uncertainty and changeability, determine the internal structure and needs of the organisation, if it is to succeed

"*The more varied the types of environments confronted by an organization, the more differentiated its structure needs to be*" (Scott, 1987)

The theoretical framework presented in this thesis constitutes a contingency theory, connecting specific types of system development situations to particular documentation profiles This is developed from the conjecture introduced in Chapter 1, that there is no one best way to document requirements, as it depends on the situation

In a classic study of organisations in the plastics business, Lawrence and Losch, who coined the term 'contingency theory,' argued that organisations in complex environments develop specialised internal departments to cope with the different environmental factors, thereby creating internal problems of coordination between these departments Successful organisations in such situations need to be able to differentiate to the level required by the environment, and integrate their operations so that they can collectively pursue the goals of the organisation (Lawrence and Lorsch, 1967)

Integration techniques used in successful organisations include direct managerial contact, hierarchy, a system of reports, and procedures for managing conflict Conflict between specialised departments is seen as a natural consequence of their conflicting interests, and negotiation of such differences as part of the process by which such organisations adapt to changing environments The more stable the environment, the more rational the organisation can afford to be

The term 'contingency theory' is more widely used in business disciplines such as accounting and management, but the concept is not unknown in the field of system development A number of early papers on the criteria for selecting appropriate requirements analysis methods invoked the notion of contingency theory either explicitly (Davis, 1982, Naumann et al , 1980) or implicitly

Contingencies such as project size, problem structuredness, user task comprehension, and developer task proficiency were found to determine the level of uncertainty inherent in a project which in turn determines the need for particular strategies for requirements assurance or validation (Naumann et al , 1980) The characteristics of the object system (the problem domain), the application system, the users, and the analysts are the main categories of contingency factors used to explain differentiation in the use of problem solving methods and techniques (Davis, 1982)

Episkipou and Wood-Harper, in a paper that further develops Davis's framework, use Checkland's terminology and other sources to produce an expanded model that includes, in addition to the presence of different ideologies, available tools, and different dominant philosophical enquiry systems

1 The characteristics of the problem owner
2 Assessment of the problem context and
3 Assessment of the problem solver (Episkopou and Wood-Harper, 1986)

The framework of situations presented in this thesis uses the idea that each type of situation has a different locus of concerns, as the findings indicate However, there are some more fundamental characteristics of organisational situations that give additional support to the distinctions between situations that were made in Chapter 6

Organisations are systems As such, they exhibit particular characteristics, for example, they are goal-directed (Scott, 1987) They are composed of parts (individuals and smaller organised groups of people) which interact with each other Like any other system, organisations work by means of communication, coordination and control among these parts Communication between its parts allows the organisation to share information and knowledge, and to coordinate action and monitor its results Control in an organisation is essential, in order to ensure that the goals of the organisation as a whole, rather than any particular part of it, are pursued and achieved The need for coordination arises when an organisation becomes more complex as its sub-units become more specialised

However, it is clear that some situations, as social systems, demand coordination mechanisms more than control or communication mechanisms, while other situations need control mechanisms more so than communication or coordination In the remainder of this chapter, I consider how the different types of situations outlined in Chapter 6 can be distinguished in terms of their varying balance of the needs for coordination, control, and communication I review the relevant organisational literature which helps to explain why certain work strategies are appropriate mechanisms in some situations and not in others

8 3 Situations that Need Effective Coordination Mechanisms

An important aspect of organisational structure is the way the organisation coordinates the activities of its members Most organisations, except the smallest and simplest, consist of smaller sub-units which are themselves organisations In an organisation of any considerable size, this gives rise to several levels, and consequent characteristic relationships between the parts of the organisation These relationships are often hierarchical, although other relationships are also possible The more complex the organisation, the more diverse and specialised the different subunits The more formal the organisation, the more rigidly prescribed and constrained are the interactions between its members (Olsen, 1968)

The different types of situations presented in Chapter 6 exhibit significant differences in the way they are organised The vertical market situation (Type 1) demonstrates the classical characteristics of a hierarchical organisation, with different levels of coordination, such as product and project steering committees There tends to be a strict division of labour, reflected in the division of the development process into formal stages, with different personnel assigned to tasks such as requirements gathering, validation, design, programming and testing

The targeted customer situation (Type 2) is similar to Type 1 in terms of division of labour, but there is less stability and more uncertainty in the market, so the development project is structured as an alliance with the targeted customer(s) Both of these situation types were associated in my empirical study with companies that were engaged in process improvement initiatives and other attempts at rationalisation of organisational activity Many of them had adopted the Capability Maturity Model, which relies on extensive documentation of both development processes and project deliverables Another organisation, representing the Type 7 Problem-oriented situation, had also developed and adopted a well-defined process for producing its Request For Proposals (RFP) documents

8 3 1 Rationalisation in organisations

The word "bureaucracy" is often associated with red tape and inefficiency, but it has a specialised meaning in organisational theory

"*Bureaucratisation is the process of rationalising social organisation, so as to improve operating efficiency and more effectively attain common goals* " (Olsen, 1968)

This process was first studied by Max Weber who identified nine characteristics of a bureaucratic 'ideal type,' including the following features which are easily recognised in many system development organisations

1 Each role has clearly identified duties and responsibilities
2 All activities are guided by formally prescribed rules and regulations
3 All decisions are made on the basis of technical knowledge
4 All activities are recorded on written documents, and preserved in permanent files

A bureaucratic organisation is one which is large, formal, and run on completely rational lines by people who derive their authority from either their technical competence or their position in the organisation, which in turn is determined by their proficiency Scientific management, work-study, cost accounting and organisational information systems are all classic examples of the process of bureaucratisation Business Process Reengineering, Total Quality Management and software process improvement initiatives are contemporary examples of the bureaucratic process described by Weber

An important strategy that organisations use when carrying out projects is the policy of writing things down Documents are regarded as indispensable tools for helping people to deal with information, to record what has been done before, to plan what is to be done, to see what remains to be done System development may be regarded as both part of the process of rationalisation, and as an activity to be rationalised in its turn, and therefore documentation is seen as an essential part of the process of system development

Contingency theory in general, and rationalisation in particular, are two important ways of explaining how and why organisations evolve and develop different ways of coordinating their operations, depending on the situations in which they find themselves Rationalisation as a coordination mechanism helps to explain the role of documentation in product-oriented situations such as the vertical market situation, and the targeted customer situation Both of these situations emphasise documentation as a coordination mechanism, coordinating the work of different groups within the organisation, and also between organisations in the case of the target customer situation

8 4 Situations that Call for Effective Control Mechanisms

In situations where one organisation develops a product or solution on behalf of another organisation, control rather than coordination becomes the important issue Goal differences between the organisations can lead to problems, unless adequate control mechanisms are in place Two different variants of Agency Theory may be drawn upon to help explain these mechanisms

Principal-agent theory helps to clarify the dynamics of the interaction in the following types of situations, in particular Situation Type 3, the product for a single customer, Type 4, the contract for a custom-built solution, and Type 5, the contract for a configured solution A different formulation of agency theory, called transaction cost economics, is more appropriate to explain the dynamics in single-organisation situations, including Type 1, the vertical market, and Type 7, problem-oriented situations

8 4 1 Agency exchange

In its simplest form, Agency Theory is a model of exchange between two people, called a Principal and an Agent The principal engages the agent to carry out responsibilities which usually involve decision-making by the agent on behalf of the principal The principal is usually cast as the owner of a business, while the agent is cast as the manager Both are assumed to be rational people, who want to maximise their utility, but they may have different goals and different attitudes to risk, though the principal is usually assumed to be risk-averse The agent is assumed to be self-interested, but willing to expend effort in return for sufficient reward The problem for agency theory is to devise a suitable employment contract that will ensure that the agent receives a sufficient share of the outcome of his work to motivate him to work in the best interests of the principal as well as his own best interests

In its original formulation, agency theory is a normative theory intended to help the principal to exert control over the agent The purpose of the model is to maximise the utility of the principal by devising and selecting (a) the most beneficial contract of employment and (b) the most suitable way of monitoring the effort of the agent, called an information system in agency theory (A cost accounting system would be a typical example) Because it deals with individuals rather than organisations, classical agency theory does not apply very well to system development situations

However, it does highlight certain issues that are relevant to any situation where the responsibility for decision-making and problem solving is devolved from one party to another These issues include information asymmetry, asset specificity, and risk or uncertainty We will return to the latter two issues below

Information asymmetry occurs when one party (usually the agent) has more information than the other party concerning the effort being put in by the agent, giving the agent an opportunity to avoid effort, a situation called 'moral hazard,' or to take sloppy decisions, called 'adverse selection ' Both of these situations cause the principal to lose out, through lack of information about what is going on, and agency theory suggests that the solution is to provide either more information beforehand (ex-ante) or more ways of evaluating the performance of the agent (ex-post) Both of these strategies involve extra costs for the principal, and the problem, as far as agency theory is concerned, is to design ways to minimise these transaction costs, while maximising the information available to the principal

8 4 2 Professional as agent

Anurag Sharma (1997) has adapted and extended agency theory to explain exchanges between professional agents and their clients He argues that principal-professional exchanges are inherently different from the owner-manager exchanges of traditional agency theory Rather than delegating the task to the agent, the principal engages with the agent in the co-production of a service product This involves a division of labour based on different levels of knowledge, which he calls *knowledge asymmetry* as distinct from information asymmetry, and an additional cause of a power difference between the agent and the principal, in favour of the agent

A system developer may be regarded as a professional agent often with specialised knowledge and skills which the client does not have This is particularly the case in Type 5, the configured solution situation, and to a certain extent in other situations For example, in Type 3, the product for a single customer situation, the client organisation may have a vast amount of knowledge, but a scarcity of resources to deal with the particular project, which is therefore outsourced to a developer organisation Knowledge asymmetry may then arise between the client representative, such as the product manager, and the members of the developer organisation, who are chosen for their specialist knowledge Practitioners in such situations in the investigation indicated that this is often the case

In situations where the agent has more expertise and knowledge than the principal, there are extra difficulties for the principal in controlling and monitoring the transaction, but Sharma argues that there are additional restraints on the behaviour of professional agents

1 self-control or altruism on the part of the agent
2 control by a community of peers, codes of conduct, protection of reputation
3 bureaucratic (or supervisory) control by the professional firm and
4 client control

One of the client control mechanisms that Sharma invokes is the notion of exchange-specific assets, which is part of the mainstream agency framework, and was mentioned above In the case of many professionals, the abstract knowledge they bring to the transaction has to be supplemented with a considerable investment of time in diagnosing the problem, a situation which is typical of the work of system development

"Upfront investment of time in diagnosing the problem is akin to investing in an asset that has no value outside of that particular exchange" (Sharma, 1997)

The risk associated with asset specificity is borne entirely by the agent, but serves as a guarantee of the long-term commitment The degree of risk that a professional agent is willing to assume is dependent on norms and traditions within the profession He gives, as an example, lawyers who represent clients in return for a contingency fee

A contract is basically a means of overcoming the problem of information asymmetry However, in many contract development situations, the requirements document is a large part of the contract It is also a major asset which is specific to the exchange For organisations which develop products or solutions on a contract basis, this involves a significant risk, because the contract may not be agreed, even though considerable resources have been invested in drawing it up For the client organisation, the existence of such assets is a guarantee of the commitment of the agent, according to agency theory

In Type 6, in-house solution situations, particularly when developing organisational information systems, both agents and principals may invest in exchange-specific assets, such as requirements documentation In a paper on participatory design viewed from the perspective of agency theory, Vivian Vimarlund suggests that shared investments made during collaborative projects result in an increase in knowledge capital, a decrease in information asymmetry, and desirable organisational change in terms of "*updating work routines in a natural manner*" (Vimarlund, 1998)

Unlike many other professional-client agency situations, in system development, knowledge asymmetry may arise in two different directions Whereas the agent organisation has specialist knowledge and skills, the client organisation often has specific application domain knowledge which the agent needs to access, in order to develop a satisfactory product or solution This means the project must often be structured as a joint effort between the two organisations, with a common goal which is understood and agreed by both sides

The professional-agent extension to agency theory not only distinguishes the concept of knowledge asymmetry, but also describes situations where clients and agents engage in the co-production of a product or service, a state of affairs which describes most system development situations, but the solution-oriented ones (Types 4, 5, and 6) in particular In these situations, the clients have a big stake in the solution being suitable for their needs, an outcome in which they themselves have to invest some considerable effort Asset specificity is regarded as the most important restraint on opportunistic behaviour by professionals who engage with clients

The additional restraints described by Sharma are all social controls based on norms rather than economic ones based on selfish utilitarian motives They arise out the analysis that professionals are social actors, and although they have more power than their lay principals in exchange transactions, they are more subject to societal and community norms than mere economic actors would be The social interaction perspective gives a further explanation of the exchanges between professional agents and their clients

8 4 3 Market situations and hierarchy situations

Transaction cost economics is a complementary theory that shares many of the concepts of agency theory It is based on the ideas of markets and hierarchies, (Coase, 1937), and Oliver Williamson's (1975) theory of organisation failures Markets and organisations are regarded in this literature as two different ways of organising transactions Traditionally, microeconomics regards the market as the normal method of exchange, and any departure from it, such as organisational cooperation as a 'market failure '

A 'market failure' is said to occur when the costs associated with one approach, for example a contract with another party in a market situation, could be reduced by transferring to an alternative arrangement, involving cooperation within an organisation When this occurs, it is better to make something or provide a service in-house rather than go to the market to buy it In this model, bounded rationality replaces utility maximising as an explanation of agent and principal behaviour and interaction

This framework is explanatory rather than normative, and allows for a wider variety of organisational structures, including the idea of the hierarchical organisation of principals and agents within the firm These include the Type 1, Type 2, Type 6 and Type 7 situations Compared to classical agency theory, the transaction cost economics framework pays more attention to ex-post evaluation of performance, which is more widely applicable, particularly when uncertainty is a factor in the exchange

8 4 4 Uncertainty in organisational situations

Many system development projects deal with situations of uncertainty, particularly those that are concerned with markets and other outside organisations, including the vertical market situation, targeted customers, and different forms of contract situations Many authors see uncertainty as a significant factor in organisational behaviour, and this recognition has two important consequences It has moved the focus of theoretical attention away from routine decision-making towards problem solving and it has shifted the established view of organisations from the traditional split between management and operations in routine stable situations towards a focus on joint action in uncertain and dynamic situations

Uncertainty in agency theory is associated with moral hazard, adverse selection and what is called the state of nature The first two of these arise out of information asymmetry, or the principal's inability to adequately monitor the effort of the agent, coupled with assumed opportunism and unwillingness to work on the part of the agent The state of nature is outside the control of either the principal or the agent and is a category that covers all other sources of uncertainty as far as agency theory is concerned

Nilakant and Rao (Nilikant and Rao, 1994) identify two other important sources of uncertainty, in an article that criticises the applicability and generalisability of agency theory These two additional sources of outcome uncertainty in organisations are

1 incomplete knowledge about the effort-outcome relationship and
2 lack of agreement about effort and outcome

They propose a 2x2 model giving four different situations One situation is where there is incomplete knowledge about the relationship between effort and outcome, but a high level of agreement on goals, which often happens in contract situations, and fits the agency theory perspective Another situation is where there is high level of knowledge about the amount of effort required but conflict over outcomes, and this can happen where systems are being developed within an organisation

Organisations which seek to develop new products or technologies operate in a domain where the relationship between effort and outcome is unknown (Nilikant and Rao, 1994) Market-oriented software development is a good source of examples of this type of situation, including Type 1, the vertical market and particularly Type 2, the targeted customer situation It would also apply to a mass-market situation

In complex organisations, where there are multiple agents and principals, there may be lack of agreement about goals and also about ways of achieving them Again, this description is particularly relevant to certain system development situations, such as Type 7, problem-oriented situations, not least because part of the task is to achieve such agreement in the course of the project It also applies to some market-oriented situations, such as Type 1, the vertical market, where different personnel represent the interests of customers and suppliers within the same organisation

Nilakant and Rao emphasise the importance of team activities and facilitative (managerial) effort in modern organisations, concepts which are not considered in the agency theory view of organisations More complex and dynamic work environments increase the need for facilitative effort, while at the same time, the distinction between management and operations is being eroded in situations where teams are being given more responsibility and autonomy *"The same individual may perform facilitative and operational tasks"* (Nilikant and Rao, 1994)

In such situations, hierarchical control is replaced by networks of inter-dependent roles, in which reciprocity, trust and cooperation are the most important means of dealing with uncertainty

For situations such as Type 1, the vertical market, and other in-house situations, such as Type 6 and Type 7, the transaction cost economics version of agency theory gives a better analysis of the organisational control of system development activity in general, and requirements gathering and analysis in particular The main feature of such situations, accordingly, is cooperation within an organisation, the need for which is stimulated by bounded rationality of the participants and uncertainty in the environment or the task to be performed

Uncertainty is a significant and perennial problem in requirements determination Facilitative effort by managers is recommended in such situations, along with conflict management and consensus management, giving more responsibility to individuals and autonomy to teams This kind of approach can probably best be used in developing in-house solutions (Type 6) or in agreeing on the requirements in a problem-oriented situation (Type 7 situations)

8 5 Situations that Need Effective Communication Mechanisms

One particular type of system development situation exhibits more need for communication than for control or coordination of activities This is Type 6, the in-house solution situation, where members of an organisation develop their own solution, rather than go outside the organisation to develop or acquire a product or solution The project may be one that is mission-critical for the organisation, developing a solution that is not available in the market, that may give the organisation a competitive edge over its customers This requires a very good understanding of the organisation, its objectives, and its environment In such a situation, a source of difficulty may arise in the lack of understanding between the developers and their clients, the users and their managers

8 5 1 Understanding in organisational contexts

Weick (1984a) and others make a useful distinction between uncertainty on the one hand and equivocality on the other Equivocality reduction is aimed at understanding the other person's meaning, at being able to see things from the other person's point of view In sociology, this aspect of understanding is called *verstehen,* which is the German word for understanding

Uncertainty, on the other hand, comes from the inherent complexity of the environment in which the person or the organisation is trying to operate, or the problem they are trying to solve Scott (1987) gives the following list of five dimensions of uncertainty that must be dealt with in an organisational context

1 The degree of homogeneity-heterogeneity diversity of entities with which the organisation must deal
2 The degree of stability-variability the rate of change in the environment
3 The degree of threat-security or vulnerability of the organisation

4 The degree of interconnectedness-isolation the extent to which the organisation must interact with other entities

5 The degree of coordination-noncoordination or the structuredness of the interactions

Reducing this inherent uncertainty requires a means of understanding all this complexity, and the dynamics of whatever processes, physical or social, that are influencing the problem situation In many organisational situations, the use of models, plans and other written representations is seen as an essential strategy for coping with uncertainty System development provides many typical examples of such strategies

In organisational contexts, the process of understanding may be regarded as a process of sensemaking Behaviour sometimes precedes goal statements, and justification follows actions to explain behaviour Interaction leads to retrospective interpretations of what has gone on before (Weick, 1979, 1984a) Sensemaking is aimed at two objectives

1 Uncertainty reduction and

2 Equivocality (ambiguity) reduction

Weick (1984a) gives a number of strategies that people use for dealing with each of these problems Sensemaking strategies include paradigms, theories of action, tradition, and story-telling Each organisation has its own strategies, which help it maintain and legitimise its particular view of reality, by simultaneously supporting and constraining it

Our human experience of uncertainty, then, is a combined consequence of the complex nature of things and our bounded rationality in dealing with them We need to be selective in what we pay attention to, we must make assumptions, because our knowledge is imperfect, and we can only deal with so many things at one time This has significant consequences for the strategies that organisations adopt for developing systems, and especially for the analysis of system requirements

A requirements document can be regarded as a strategy for dealing with both uncertainty and ambiguity (equivocality) in situations It aims to reduce uncertainty by recording all the relevant information about the problem and the constraints on the solution It aims to reduce equivocality by being clear, unambiguous, and consistent in communicating these matters However there is a danger that it will not be completely successful in these aims, because people are not completely rational, and often have trouble interpreting each other's meanings

8 5 2 Interpretation in Organisations

As social systems, organisations are often considered to best understood in terms of the interactions of their members as human beings These interactions are based on meaning or more precisely, on the meaning that we attribute to the behaviour of others

In order to work together in social situations, people are continuously doing two things

1 interpreting (or ascertaining what the other people mean) and
2 indicating to the other people (what to do, how to act)

According to Blumer, these are the two things that people do *"in order to sustain joint conduct"* (Blumer, 1969) This theory is called symbolic interactionism and has its roots in a long line of pragmatic philosophers such as William James, George Herbert Mead and John Dewey, as well as the work of the sociologist Max Weber It has three basic axioms

1 People act towards things in ways determined by the meanings those things have for them
2 The meaning that things have is not inherent in the things, nor in the people, but arises out of social interaction
3 These meanings are continually being modified through an interpretative process used by people in dealing with things

Most of what we know comes from interpretation (March and Olsen, 1976) Cultural rules are the instructions we have for interpreting, constructing and otherwise dealing with the meaning of things (Spradley, 1972b) It is a social process, which frames reality as we experience it Communication is both part of this process and constrained by it

When people work together in organisations, they are therefore subject to significant cultural rules and pressures which help to define reality for them Society itself is one source of these rules Other sources include the various layers of organisations the people belong to "*The words that matter to the self, matter first to some larger collectivity*" (Weick, 1984a) Weick himself quotes George Herbert Mead as saying *"Social process precedes individual mind"*

8 5 3 Reality Construction

Lincoln and Guba (1985) discuss four different perspectives on reality, including 'constructed reality,' which holds that even if there is an objective reality, we can never know it directly The best we can achieve is a consensus view of what we think it is This position has it roots in many philosophical thinkers, and corresponds to the theoretical view expressed in symbolic interactionism, above George Kelly, who developed a technique for collecting people's own constructions of reality, thought it was presumptuous to assume that a person's constructs were convergent with some objective reality, and is quoted as saying *"the open question for man is not whether reality exists, but what he can make of it"* (Lincoln and Guba, 1985)

Constructed reality allows for multiple perspectives or multiple realities to exist side by side in the same organisation As a theory of social organisation, it is associated with Garfinkel and with Berger and Luckman (1973) who propose that social reality is a human construction that is constantly being created by a process of social interaction Over time, actions which are repeated and are given similar meanings by the self and others are said to be institutionalised

The implications of reality construction for requirements analysis are clear In situations where a social process has been initiated for the purpose of knowledge acquisition, representation, understanding, decision-making, and problem-solving, by a group of people with different roles, experiences and perspectives, there is bound to be a considerable amount of reality construction going on Communication is an important factor in this process It simultaneously enables and constrains the process of system development

8 5 4 System development in and for an organisational context

A substantial amount of software is developed in order to be deployed in an organisational context The notion of reality construction then becomes extremely significant, as Christiane Floyd and others have established in their book "Software Development and Reality Construction" (Floyd et al , 1992)

Organisations are themselves social constructions, but not as firmly established as other (seemingly more concrete) parts of reality, so how they are perceived and experienced differs considerably from individual to individual In situations where the organisation itself is a customer for the software, as well as the individuals who are going to use it, this can cause problems if new systems are perceived as being for the benefit of the organisation at the expense of the user The possibility of multiple realities occurring in the same project then arises and steps must be taken to achieve consensus A considerable amount of research on this problem has been done in the area of Participatory Design

Participatory Design is a normative approach to system development Its roots lie in the democratisation of work, particularly in the Scandinavian countries following legislation in the early 1970's to increase worker influence in decisions affecting their work practices and working conditions A number of trade unions sponsored action research projects to examine how this legislation might be applied to the development of organisational information systems

Around the same time other researchers, including Enid Mumford in Manchester and later on, Christiane Floyd in Hamburg, were responding to the perceived need within system development for better ways of promoting interaction between users and developers, in order to produce more effective, more usable, and more acceptable systems

Like other participative design methods, the ETHICS (Effective Technical and Human Implementation of Computer-based Systems) approach embodies the philosophy that users should be able to help create systems that are "*humanistic and friendly, as well as efficient and effective*" (Mumford, 1993) It includes a number of tools that help groups to design work, to identify goals and constraints, problems and problem areas, to measure job satisfaction, and to identify likely changes in the internal and external environments This approach has been used for operational system design in factories, shops and offices, and has also been used to design decision support systems and as a general problem-solving tool

Some types of solution-oriented and market-oriented situations can benefit from adopting an evolutionary learning perspective to deal with uncertainty The guiding principle is that over time, knowledge increases and uncertainty decreases with continued use Newer versions of products and solutions fit better with the situation of use, even as it changes But participants in contract-based situations, and in problem-oriented situations, typically do not have the same opportunity to rely on second-order learning to help deal with uncertainty in their immediate development task, and must fall back on more traditional strategies

The way requirements are documented in contract-based situations is a useful strategy for dealing with both uncertainty and ambiguity Uncertainty can be reduced to a certain extent by recording all the relevant information about the problem, thus protecting both sides in the event of a dispute over the outcome

8 6 System Development as Situated Action

The theory of situated action says that (human) action is always situated in some set of circumstances Situated action involves taking cues from the situation as to how to proceed, rather than following a detailed pre-defined plan (Suchman, 1987) Accordingly, a system development method is not to be regarded as a set of directives to be obeyed, but a resource to be used only as a guide Similarly, a requirements document is not a specification for action, but a resource that people make to guide subsequent action Different circumstances necessitate and provide for different uses of documents such as requirements documents, and these are best regarded as resources to be used as the situation demands

According to Suchman, who is an anthropologist, because action is always situated in some set of circumstances, social as well as physical, the situation itself is critical in the interpretation of the action Under certain conditions, people make plans that guide their actions These plans are abstract descriptions, made either before or after the fact of some action Plans are therefore best regarded as resources that people use, to guide but not to determine their actions in a given situation

She suggests that situated action can be supported *"by defining the processes by which efficient representations are brought into productive interaction with particular actions in particular environments"* (Suchman, 1987)

This is often taken to be a criticism of any kind of predefined representation, but requirements documents and other artefacts produced during system development are like plans, resources to be used to help guide people's actions and their interpretations of each others actions The same could be said of the target system itself It could perhaps be best regarded as a resource for situated action, just like the intermediate products and documents that help to bring it into being

Another reason often cited by critics of formal approaches to system development is the use of tacit knowledge (Polanyi, 1966) in dealing with situations Polanyi distinguished between propositional knowledge, which is shareable, and tacit knowledge which is personal and based on reflection and experience In his book, the Reflective Practitioner, Schon (1983) argued that reflective practitioners use tacit knowledge to augment explicit technical knowledge in order to cope with unstructured situations for which their professional training does not prepare them Systems analysts fit the description of reflective practitioners because their work constantly involves learning about and making sense of a situation, rather than strictly applying pre-defined technical knowledge that is defined in methods and techniques (Fitzgerald, 1996, Vitalari and Dickson, 1983)

A related concept which is possibly more relevant to the question of situated methods is the idea of espoused theories of action, that is the difference between what people say (they would do) and what they actually do (Argyris and Schon, 1978) Shipman and Marshall analysed a number of projects in which users had difficulty adapting to new systems which required them to formalise the information that they used in the context of their work They found that users' reluctance to make explicit certain aspects of their work tasks was related to a desire to maintain control over their work (Shipman and Marshall, 1999)

8 7 Summary

Many of the problems and challenges of system development can be attributed to the fact that it takes place in an organisational context The different types of situations that were presented in Chapter 6 were distinguished from each other by their particular concerns, but since system development is an organisational activity, these also correspond to different organisational archetypes Contingency theory helps to explain why organisations, including system development organisations, adopt different internal structures in order to deal with their environments Organisational structures give us insights into some of the problems and challenges that apply to requirements analysis in the different types of situations

Organisations have evolved particular strategies, such as methodical procedures, in order to overcome the known limitations of human beings (bounded rationality) as decision-makers and/or problem solvers Records and documentation are typical examples of these strategies Routines (procedures) are another type of strategy that frees human attention to cope with more unusual tasks or events Requirements analysis work is non-routine, particularly for many of the stakeholders involved, such as the managers and end-users in organisational information systems System development methods are seen as a way of structuring and controlling that activity

Organisations by their nature try to impose structure on unstructured situations Many organisations try to evolve rational behaviour, to cope with the uncertainty inherent in their environments Many software process improvement initiatives are examples of attempts to control the uncertainty that is inherent in system development situations Organisations try to do this by formulating plans and procedures for carrying out the various key activities associated with system development System documentation and programming standards are typical examples of how some organisations try to constrain and control the 'work products' of system development

Work within organisational contexts is subject to particular problems of interpretation, uncertainty, ambiguity, and equivocality Organisations can respond to these problems by becoming learning organisations, and the people in organisations respond by becoming sensemakers Reflective practitioners use tacit knowledge to cope with unstructured situations System development work, particularly requirements analysis often involves sensemaking for the individual and organisational learning for the organisation

The social construction of reality is an important principle which summarises the way that reality, particularly organisational reality, is continuously being created by a social process of symbolic interaction It allows for the possibility of multiple perspectives, or multiple realities co-existing in different parts of the same organisation This has important implications for the development of systems in organisational situations, particularly systems intended for use in organisations

The notion of situated action could be taken to mean that all situations are different, and that there is no point in developing a classification of situations, such as that presented in Chapter 6 However, situated action does support the idea that situations differ, and that the differences might be understood by looking at the factors, such as concerns, that shape the context of a situation

8 8 Conclusions

The different situations identified in this thesis present some essential differences as organisational situations One example is the division of labour in different situations, resulting in different levels of formality, and thus, different needs for coordination of effort In market-oriented situations, particularly Type 1 and Type 2 situations, there tends to be a strict division of labour, separating the development process into formal stages Different personnel groups assigned to areas such as marketing, requirements analysis, design, programming, and testing, need effective means to coordinate their activities There may also be different levels of coordination, such as product and project steering committees The requirements document plays a role in these situations, mainly as a discussion document, as a method of communication, as a working document but also as input to downstream activities such as design and testing

Rationalisation of system development activities is an important feature of such situations Many of them adopt process improvement initiatives which rely on extensive documentation of both development processes and project deliverables In contingency theory, high levels of formalisation are associated with stability in the environment, but the Type 2 situation has more uncertainty in the market place, and therefore, less reliance on formal procedures The relationship between effort and outcome is unknown in these situations, increasing the need for teamwork and joint effort by management and staff

Type 3 (single customer) situations have some features in common with the other market situations, but are different in that the agreement is a contract with the customer This is similar to the Type 4 and Type 5 situations, in that the requirements document as part of the contract, is aimed at controlling the interaction between the two parties Knowledge asymmetry between the parties to the agreement is a problem identified by agency theory, but the requirements document, as well as recording the agreement, may play a role as a specific asset, an investment made by the developers which ties them in to the situation

Solution-oriented situations, i e Type 4, Type 5 and Type 6 are all concerned with solutions that must fit in a specific organisational situation This gives rise to the need for communication to uncover the multiple perspectives, or multiple realities, that exist within the organisation But in two of these, Type 4 and Type 5, which are agent/principal situations also, there must be a balance between the need for control and the need for communication in the situation, resulting in a particular set of roles for the requirements document These include its use both as a method of communication, to tell the users 'what they are getting,' and to document the agreement

For situations such as Type 1, the vertical market, and other single-organisation situations, such as Type 6 and Type 7, the transaction cost economics version of agency theory gives a better analysis of the organisational control of system development activity in general, and requirements gathering and analysis in particular This version allows for a wider variety of organisational structures, including hierarchical organisation of principals and agents within the same organisation The main feature of such situations, accordingly, is cooperation within an organisation, the need for which is stimulated by bounded rationality of the participants and uncertainty in the environment or the task to be performed

Type 6 situations are more concerned than any other type with communication within the organisation, and the role of the requirements document as a method of communication reflects that However, some organisations are more formal than others in this respect, and this is likewise reflected in the use of prescribed methods for system development, including more formal documentation Sensemaking is a repertoire of organisational strategies for reducing both uncertainty and ambiguity In many situations, therefore, representations such as those found in requirements documents can represent only a part of what is known and agreed by the participants

In the next chapter, I review some of the relevant literature on representation and its role in the documentation of requirements Three basic questions will be addressed

1 what different aspects need to be represented in requirements documentation,
2 why models are useful mechanisms for representation, and
3 why using models for requirements representation can be problematic

Chapter 9: Perspectives on Representing Requirements

9 1 Introduction

Chapter 4 has introduced a new way to classify requirements that is different from the established typologies of requirements that have appeared in the literature thus far However, this classification of requirements as texts, although grounded in the taxonomic analysis of the interviews and sample documents, also has several roots in the literature on requirements representation, which will be reviewed in this chapter, and linked to the relevant elements of the classification It will then be compared and contrasted with existing frameworks in the next chapter

This chapter explores the nature of requirements representation, the uses of models as a form of representation, and reasons why models are not always the best ways to represent requirements The sections and sub-sections in this chapter reflect the elements of the typology of requirements from Chapter 4 It contrasts the product-oriented perspective with the process-oriented perspective It concludes that a process-oriented perspective is often needed in addition to the dominant product-oriented perspective that is embodied in many current approaches to requirements representation

9 2 Requirements representation

A representation is something that is made for the purpose of standing for something other than itself The construction and use of representations is an essential part of system development generally, and requirements analysis in particular The forms of representation that are used for representing requirements include various types of artefacts called descriptions, statements, specifications, prototypes, and models

Unfortunately, these terms are mostly used rather loosely, are often used interchangeably, and are rarely differentiated in the literature where they are used However, David Parnas (1995) makes an interesting distinction between a description, a specification, and a model

1 A description is a statement of some of the actual attributes of a product, or a set of products
2 A specification is a statement of properties required of a product, or a set of products
3 A model is a product, neither a description nor a specification Often it has some but not all of the properties of some "real product" (Parnas, 1995)

A prototype is a good example of what is meant by a model in this scheme A communication protocol is a good example of what is meant by the term specification It states what is required of any product that is intended to implement that protocol Parnas points out that to illustrate the protocol, during discussions, with a finite state machine is to replace the specification with a model, a product that is intrinsically different Models have some but not all of the required properties, as well as having other properties that are not required, of the intended product (Parnas, 1995)

These are useful distinctions, but this use of terminology is neither standard nor widespread in the literature and practice of software development The terms description, specification, and model are used interchangeably For example, formal notations, such as VDM or the Z notation, are usually referred to as specification languages, though their intended purpose is for creating system models, called model-based specifications The term product may be used by some authors to refer to any of the artefacts which are the products of the system development process, such as specifications, while others reserve this term for the final product, the intended system

In my empirical study of system development practice, there was clear evidence that all three types of representation, descriptions, specifications and models, are present in the practitioners' requirements documents However, these elements were mostly combined and interleaved in the same artefact Though they can be distinguished, the distinction is not made explicit in the documents The term 'statement' is used in relation to all three types of representation outlined above, and accordingly has been adopted in the grounded theory to refer to any type of requirements representation

Another, possibly more useful, distinction encapsulated in the above definitions is the difference between a statement of what is to be (which Parnas calls a specification) and a statement of what is (which he calls a description) This is an important difference for system development, which is concerned with both things as they are, that might be required to change or to stay the same, and things that are required to be brought into being Different aspects of requirements that might need to be represented provide a more generic way of classifying different types of requirements statements than any distinction between different types of representations

9 2 1 Separation of concerns

This distinction, between what is and what is required to be, between what happens and what is required to happen, highlights the difference between two separate aspects of requirements that must be represented, namely, the relevant characteristics of the problem domain and those of the intended system Parnas has called this "the separation of concerns" This separation refers to the need to document not only the required properties or behaviour of a proposed system but also the environment or context in which the system will operate

Parnas was writing about the specification of real-time systems These deal mainly with readings taken from the system's environment, "environmental quantities," such as the temperature and pressure of different chemical or physical processes Some of these quantities are to be monitored by the system, as readings taken from various instruments, and some of them are to be controlled, as values sent to other instruments

The purpose of the specification, in Parnas's view, is to specify the relations between these quantities, and this should be done by defining two separate relations The behaviour of the environment would be captured in a relation called NAT and the required behaviour of the computer system would be captured in a relation called REQ In this approach, NAT is a description of the environment, and REQ is a specification of the software that is required (Parnas, 1995)

A similar separation of concerns appears in Jackson's distinction between the Application Domain and the Machine (Jackson, 1995) Jackson uses the term description in a more general sense than that of Parnas According to Jackson, for any kind of system, you need to describe (i e represent) three things

1 the problem domain
2 the machine, and
3 the connection(s) between them

The machine is intended to implement the required behaviour in the problem domain, therefore, it will to some degree be embedded in that domain How deeply embedded will depend on the types of connections used, among other things (Jackson, 1995)

Jackson's framework goes further than that of Parnas by maintaining that the two main descriptions (that of the problem domain and that of the machine) ought to be written in different grammatical moods, the Indicative mood for the description of the environment, and what he calls the Optative mood for the machine requirements

This separation of concerns suggests a fundamental partitioning of the task of requirements description, a partitioning which seems to be ignored or even confused by Object-Oriented approaches to requirements modelling Object-Oriented approaches not only use the same set of constructs for analysis and design, but also fail to recognise or support the distinctions made by Parnas and Jackson In Object Oriented Analysis, the same basic constructs are employed for modelling the problem domain and the machine, to use Jackson's terms But these are different things, from different worlds

"The difficulty in moving from OOA to OOD is caused by the fact that OOA objects and OOD objects are inherently different things" (Kaindl, 1999)

These are two separate aspects of software requirements that need to be represented, the environment and the machine It is essential to be able to distinguish them in the representations, because one of them (the environment) is assumed to be already there and the other (the machine) is not Using different grammatical moods to describe them acknowledges this fundamental separation of concerns Jackson suggests that three different descriptions are needed E, R, and S

E is an indicative mood description of the environment or problem domain

R is an optative mood description of the required effects of the machine S in environment E

S is the specification of the machine which will produce these effects in this environment

"If a machine which has the properties described in S is installed in the environment, and the environment has the properties described in E, then the environment will exhibit the properties described in R " (Jackson, 1997)

The analysis of requirements as texts presented in Chapter 4 draws on this distinction, and distinguishes problem domain descriptions from other types of requirements statements, such as goals and required effects This distinction is also supported by the data In practice, the most important and relevant aspects of the problem domain are typically represented in requirements documents by descriptions written in natural language, but they can also be represented by various types of models

9 2 2 Representations of the application domain

It has often been noted that experienced analysts and software engineers often have extensive knowledge of the application domain in which they work, e g manufacturing, finance, telecommunications switching etc For example, in their influential study into the problems of developing large systems, already referred to in Chapter 2, Curtis, Krasner and Iscoe identified "*exceptional designers who could map deep application knowledge into a computational architecture*" as important contributors to the success of projects This kind of knowledge is gained through the experience of working on projects in the same application domain, rather than through formal training (Curtis et al, 1988)

9 2 3 Domain models

It seems plausible, however, that if this knowledge could be captured in such a way that it can be re-used, that this would be a good way subsequently to approach the goal of gaining knowledge about the problem domain, and in such a way that aspect of requirements specification would not need to be done again from scratch, but would be derived from some suitable pre-defined domain model

The domain modelling "project," therefore, is based on the idea that a chosen application domain can be studied from a general point of view to discover the generic requirements of all the systems that could be required for that domain, their data, their behaviour, their functions, their requirements in general The result of domain modelling is a normative (generic) model, which can later be instantiated to several specific domains According to Dines Bjørner, an instantiated domain model can then be used to generate a specific requirements model (Bjørner, 1998)

9 2 4 Reference models

An excellent example of domain modelling that is widely used in practice occurs in the domain of manufacturing Here domain models are used at various levels in the development and deployment of software The practice began with Prof A -W Scheer who built what he called a reference model for the area of manufacturing This reference model is a comprehensive Entity Relationship model covering all of the application areas within manufacturing such as material requirements planning, job scheduling, inventory management, and product data management (Scheer, 1998)

The reference model is used as a starting point for modelling a specific situation, by specialising it or localising it to that specific situation The elements of the reference model act as a checklist to decide which aspects are to be included in the resulting model and which do not apply, and are therefore omitted The result is a model which fits the given problem domain, containing all or most of the elements that are likely to be needed in that application

9 2 5 Requirements as effects

Both Parnas and Jackson have identified and highlighted the distinctive characteristics of requirements stated as effects These are simply effects that are required to be produced in the problem domain Parnas calls them REQ and Jackson calls them effects Following Jackson, Kovitz calls them required effects These contributions have given theoretical support to the widespread practice of stating requirements in terms of a Black Box, by listing the inputs and outputs of the system This approach is also reflected in the practice of writing requirements as a list of paragraphs beginning with the phrase, "*The system shall* ," a custom which has a long tradition in industry

9 2 6 Requirements as goals

Many researchers in the field of requirements engineering seem to agree that objectives or goals have an important role to play in requirements, but there is less agreement in the literature on the extent to which goals should determine the structure of the requirements or to what extent goals should be modelled in the requirements statement In many approaches to representing requirements, goals are not given explicit consideration at all

Axel van Lamsweerde and his colleagues have based KAOS, their approach to requirements modelling, on the idea of modelling goals The resulting model is a hierarchy of goals and sub-goals at different levels (Lamsweerde et al , 1998) Requirements in the sense of Parnas's REQ and Jackson's *effects* are essentially goals belonging to the lower levels of this hierarchy

KAOS is radically different from other modelling techniques such as E-R and OO in that the goals are the main modelling constructs, from which sub-goals and other requirements are derived Again, note the possibility of derived requirements But more noteworthy is the idea of requirements as goals, rather than as "objects of interest in the real world "

Most other approaches to modelling in system development are predicated on building an objective model consisting of objects and their relationships that are supposed to exist in some part of the real world Goals can be problematic, because they do not exist on their own The problem of whose goals they are, and how goals conflict with each other, then arises

In a business organisation, the goals of the organisation may be expressed in a mission statement Information Engineering (Finkelstein, 1989) is a method that tries to relate information systems to business goals, beginning with an analysis of the business strategy, resulting in the Information Strategy Plan (ISP) This analysis consists of four steps

1 the business mission,
2 the long-term goals of the business

3 the problems that make it difficult to achieve these goals, and

4 the Critical Success Factors, as perceived by the management

This approach makes a distinction between goals and objectives Objectives are more specific than goals They are measurable, and time-constrained, unlike goals, and related to higher-level goals

In general, goals and objectives can be regarded as very high-level requirements They help to evaluate solutions or proposals Goals break down into lower-level requirements, which are in effect sub-goals that are more specific, and more measurable, and should be, to some extent, more explicit

9 2 7 Requirements as constraints

In a sense, all requirements are constraints By definition, each of the requirements for a system effectively restricts the set of possible solutions that will satisfy the requirements as a whole But some requirements are considered hard constraints, and these are documented differently in many of the cases that I came across, because they are not negotiable My analysis of requirements as texts found two main types of constraints in the documentation of practitioners, behavioural constraints and design constraints However, business rules and application standards such as those that apply in the telecommunications domain could similarly be regarded as constraints

Design constraints are usually treated in the literature as non-functional requirements, though some authors make a distinction between these and other types of non-functional requirements

Much of the literature supporting the specification of constraints occurs in the area of formal specification, particularly for applications which are safety-critical or need to be dependable for one reason or another (Leveson, 1995) Techniques such as Z and VDM emphasise the specification of behavioural constraints in the context of a system model

9 3 Requirements and Solutions

Another distinction that is often made in the requirements literature is the separation between problems and solutions Requirements statements are supposed to be concerned with problems, not solutions Design is supposed to be concerned with solutions It is frequently advocated in the literature that the requirements statement should avoid any mention of design or suggested solutions, and concentrate on stating the problem Perhaps this is intended as a division of labour, because part of the audience of a requirements document is the users, and, by definition, they are not supposed to have any expertise in design, nor want to see it discussed in the requirements document But problems and solutions are not so easily separated, and solutions often appear in requirements documentation, as I found in my empirical study

Many of the practitioners I interviewed were concerned just as much with solutions as with problems One of them even called his requirements document a "Solution Investigation" document The problems that system development practitioners deal with in their work are nearly always problems that are assumed to have a software-based solution In the early days of organisational system development, systems analysts did not automatically assume a software system would be part of the solution, but it became the norm A software system is assumed to be part of the solution, and to a greater or lesser extent is often outlined in the requirements document This intertwining between problems and solutions is recognised in the literature that associates requirements with Problem Frames, system architecture and design

Jackson's Problem Frames emphasize the relationship between problems and solutions A problem frame suggests the solution task (Jackson, 1995, Kovitz, 1999) For the analyst, the use of a specific problem frame (or, in the case of in a complex problem context, combination of problem frames) helps to focus the problem more clearly by suggesting what questions need to be answered and consequently what needs to be covered in the descriptions (of the application domain, the machine, and the connections)

The description of a lending library system ought to contain different information than would be found in the description of a lift control system, because the solution to each of those problems is going to be qualitatively different However, the requirements for a video library system will have much in common with those of a book library system This fact is reflected in their having a common problem frame Similarly, a lift control system will display common features with other types of control problems that match the "control" problem frame

This is not to suggest that all problems can be categorised into a few classic problem frames, but that there are some similarities which different problems share which can help to begin the task of solving them, by outlining what needs to be described in order to adequately describe the problem so that a solution can be found It also suggests that the question of when a given description is sufficient to describe a problem is related to the problem at hand, rather than being independent of it Similarly, the problem frame, or more likely the appropriate combination of problem frames should suggests what aspects of the problem need to be represented in the requirements documentation

9 3 1 Requirements and system architecture

As well as framing how the requirements are to be represented, the problem frame may be said to determine the architecture of the solution, the system architecture Some authors have even suggested that software architecture may be considered an integral part of requirements (McDermid, 1994)

Having a particular architecture in view can lead to particular requirements implied by that architecture, what might be called "derived requirements " Such derived requirements may be either problematic or useful They may cause problems if they are extraneous to the problem, but alternatively they may help the requirements task when the chosen architecture acts a vehicle for general (or generic) requirements which apply to the problem situation, and therefore do not need to be specifically mentioned

A good example is the way that database-oriented requirements tend to focus mainly on specifying the structure of the stored data, and then on the requirements and rules for the creation, deletion, and amendment of data In such cases, specific information retrieval or reporting requirements are not explicitly specified, once it can be demonstrated that the stored data will support them along with a query language such as SQL

9 3 2 Requirements and design

For many years there has been a controversy in academic software development circles regarding the need for the separation of requirements and what might be called "design" (Swartout and Balzer, 1982) But it seems, from my interviews, that it is not possible to separate requirements from design, any more than it is possible to separate problems from solutions In many cases where user interface requirements were included in requirements documents, they were indistinguishable from user interface design, for example Also, practitioners following recognised methods of information system development tended produce an "implementation-independent" logical database design, but this was generally implemented without any transformation, as a relational database

Participatory design, where users are involved to a greater extent than normal in information system development, does not make any distinction between requirements and design

Whatever users design in such contexts becomes, by virtue of that fact, the requirements (Floyd et al, 1989)

In arguing for the separation of concerns, both Parnas and Jackson seem to be making a case for the separate presentation of these aspects in different documents, or different sections of a document I have found that this does not happen in practice, although in some cases, different aspects were tagged as such but were interleaved in the document, so that they could be read and interpreted in context, and in relation to each other Neither Parnas not Jackson seem to deal with this as a possible approach

The grounded theory of documentation presented in this thesis, rather than advocating any particular approach to representation, seeks to indicate the basic types of requirements statements that might appear in a document Both the empirical data and the theoretical arguments that are outlined above suggest three basic types of requirements statements that play a role, to a greater or lesser extent, in the requirements documents in different situations These are

- Problem Domain Descriptions
- Required Effects, and
- Proposed Solutions

Goals and Constraints as types of requirements are also well established in the literature on representation and modelling A model is a product These five elements of the classification are supported because the literature takes a product-oriented perspective The other two elements, **Issues and Changes** and **Recorded Agreement**, both reflect a different perspective, the process-oriented perspective

9 3 3 The process perspective

The process-oriented perspective is a paradigm of system development which stands in contrast with the dominant product-centred perspective It supports the finding that among the most important uses of the requirements document are its process-oriented uses as a working document, as a discussion document, as a method of communication, helping to get agreement It regards the outcome of development not as a finished product but as a tool or medium which users continue to learn about and to improve during use (Floyd, 1987)

STEPS (Software Technology for Evolutionary Participatory System Design) (Floyd et al, 1989) regards system development itself as a learning process for everybody concerned It is evolutionary in the sense that insights into the functionality and uses of the software only emerge over time Insights and, later, models are based on what is called *perspectivity* which refers to the different perspectives held by the participants, such as the use perspective and the development perspective At each step of the process, a new version of the software is incorporated into use, and the anticipated reorganisation of work takes place, followed by the evaluation of the result, and mutual learning for the participants The "*anticipation of use*" is an important aspect of the development process The use of intermediate products is situation-specific, and requirements are never fixed in advance, but allowed to emerge over time, evolving as a result of concomitant organisational change

The process perspective is an essential foundation of the grounded theory presented in this thesis It is reflected in the different uses of the requirements documents reported in the findings, and in the following discussion of the use of models, and other representations in the process of understanding requirements, dealing with complexity, rational reconstruction of decisions, and recording the issues and changes that take place, and keeping track of the status of the evolving agreement

9 4 Requirements modelling Uses and Abuses

Building models is an important part of the process of requirements specification and analysis Indeed, many approaches assume that modelling is the only way to document requirements Some of these approaches assume that requirements models will be built from scratch, others propose using pre-defined models such as reference models or domain models The utility of problem frames is based on the likelihood that the same or similar problems have existed and/or been solved already However, there are not many application areas where comprehensive reference models are available, where the work of modelling has already been done to any significant extent The research area of domain modelling aims to rectify this situation It suggests that there is considerable scope for the re-use of application domain models in system development (Bjørner, 1998)

Domain models (and reference models) rely on the fact that models are useful forms of representation, because they embody knowledge in a concise manner However, even though models are useful, as many of the practitioners revealed in the investigation, the act of building a model is even more useful

9 4 1 Understanding requirements

The purpose of a model, such as a domain model, is to capture extensive and comprehensive knowledge of an application domain In contrast to having to acquire this knowledge through interacting with subject matter experts, such as experienced users who have tacit knowledge of the application domain, this knowledge is available in an explicit form, so that an analyst using the model will be able to access it and re-use it in building a specific application model The domain modelling approach seems to presuppose that there are only two main purposes of models

i to embody and organise knowledge and
ii to communicate knowledge

But it does not address one of the main reasons why, according to my empirical observations, analysts build models from scratch, which is for the purpose of thereby developing their own understanding of an application domain Very often, developers will not expect end-users to understand their models, but they will build models nonetheless, to satisfy themselves that they understand the problem area In order to build the model, the developer needs to achieve a very good understanding of the application domain, and for many practitioners, that is the main purpose of building a model

> "it is an attempt to indicate that you have understood the communication"
>
> "here's our understanding of what you told us"
>
> "We had to try and understand everything because for us it was putting something down for us that was clear "

Requirements analysts deal with understanding on a regular basis Their job is to not only understand situations, but to master their complexity as a matter of course They have to make sense of situations from partial information, and often from conflicting accounts and perspectives of other people, and state the problem in such a way that it can be solved by a formal system running on a computer

Modelling was one of the skills of *exceptional designers* which were deemed to be crucial in respect to understanding requirements in the Curtis, Krasner and Iscoe field study (Curtis et al , 1988) Not only were they extremely familiar with the application domain, they were also very skilled at integrating this knowledge across functional components, and with the technical knowledge that would be used to construct the solution

"Exceptional designers were skilled at modeling the interaction of a system's different functional components, and occasionally developed notations for representing them also adept at identifying unstated requirements, constraints and exceptional conditions "

9 4 2 Dealing with complexity

In a famous article, Fred Brooks (1987) identified Complexity as one of the essential difficulties of software development This is because the problems that software developers deal with, as a matter of routine, are non-routine problems Over the years, as problems become better understood, and standard solutions are developed, the type of problem being addressed by system developers has gradually progressed to more complex and difficult problems

This essential complexity in problem situations is due, not only to the problems themselves, but also to the fact that human thought processes can only deal with a certain amount of information at a time This is the problem of bounded rationality (Newell and Simon, 1972) which was discussed in the previous chapter, page 149 Since they can only pay attention to a limited number of things at a time, people tend to make decisions that are not entirely rational Therefore, the best that people can do in any problem situation is to satisfy certain selected criteria that they give their attention to, while ignoring other factors, so that they never reach completely optimal solutions

Davis (1982) explored a number of different barriers and limitations to rationality in the determination of requirements He gave the following list

1 Anchoring and adjustment People tend to make judgements by making an anchor point and adjusting from that
2 Concreteness Using only available formats, people will not look for alternatives
3 Recency People are more influenced by recent than past events, giving greater weight to the former, and less weight to needs based on less recent events
4 Intuitive statistical analysis Humans do not make good intuitive statisticians They draw unwarranted conclusions from small samples, they confuse causality with joint occurrence They therefore often misjudge the need for information (Davis, 1982)

As a result, unless these problems are recognised and dealt with, requirements tend to be biased towards *current procedures, currently available information, recent events and inferences based on small samples* (Davis, 1982)

In general, these limitations apply just as much to system developers as to end users, but it has been found that highly rated systems analysts have cognitive styles and problem-solving methods which are essentially different from those used by poorly rated analysts (Vitalari and Dickson, 1983)

The use of abstract models is a common strategy for overcoming bounded rationality, by reducing the number of elements we have to pay attention to, while at the same time ensuring that the most important aspects of problem situation are attended to The use of models simplifies the cognitive task of understanding problem situations and dealing with complexity By abstracting away the details and simplifying the problem, models allow the developer to focus on selected aspects or chunks of the problem at any particular time

Therefore, rather than using models as communication devices, which is one of the stated purposes of models, developers often create models mainly in order to improve their own insight into a problem situation

> "One of the justifications for entity modelling is that you can show it to the users and say 'is this correct?' I don't know if this is always true A lot of users, as soon as an IT person pushes a piece of paper in front of them, no matter what's on it, they switch off "
>
> "To me, the main value of it is the understanding it gives me "

9 4 3 Models can be problematic

Models have certain pitfalls that can make them problematic in some situations when they are used to state requirements Any kind of representation can be problematic, because it must be interpreted, and people's interpretations vary Models, as relatively formal representations, are supposed to have more controlled interpretations than other forms of representation, but despite this, they are subject to two major disadvantages when it comes to interpretation

1 Because they are abstractions, they are not directly refutable

2 Because they use specific notations, they are not universally understood

Jackson (1995) makes an important distinction between refutable descriptions and definitions Refutable descriptions are based on explicit designations of phenomena in the real world They are either true or false, unlike definitions Definitions are similar to models in having no possibility of being false They are tautologies

Everyone should be able to understand a problem description written in plain language, but if you gave me a description in plain Greek (or Dutch or Spanish etc) rather than plain English, I would not be able to understand it The use of "plain language" presupposes facility with a specific natural language Otherwise, the receiver is at a great disadvantage Similarly, a developer who is proficient in the use of UML diagrams or data flow diagrams has a distinct advantage over a client who is not, in trying to understand, let alone criticise, a model written in one of these notations This advantage is called "model power" and it is one of the main reasons why using models in organisational situations can be problematic (Floyd, 1995)

Some advocates of Participatory Design, for example, (Gasson, 1999), reject any formal representations that may be used in conventional development methods Consequently, it is hard for them to propose (prescribe) alternative methods, but they always favour more situated, people-centred, integrative approaches Also, some of them do not seem to allow that rational approaches or formal notations may, even sometimes, be appropriate

9 5 Documenting Issues and Changes

9 5 1 Rational reconstruction

System development is often supposed to be a rational process, and ideally, it is, based on objective knowledge and understanding, proceeding from an initial statement of requirements to the delivery of a system satisfying those requirements But system development rarely follows such a smooth course Several years ago, Parnas and Clements (Clements and Parnas, 1985, Parnas and Clements, 1986) recommended using documentation, particularly requirements documentation, as a way of faking the ideal rational process of system development After describing the ideal development process, they finally describe how it should be faked by subsequently producing the documentation as it would have been done, had an ideal process been followed

"The documentation is our medium of design and no design decisions are considered to be made until they have been incorporated into the documents No matter how often we stumble on our way, the final documentation will be rational and accurate' (Parnas and Clements, 1986)

This faking strategy is widely known as rational reconstruction, an idea which was first identified by Carnap, and developed by Reichenbach, who maintained that *"logic governs the result of thinking, not the process of thinking itself"* (Wieringa, 1996) A similar strategy of faking occurs in mathematics, whereby the orderliness with which a proof is presented belies the method of its discovery Likewise, the actual research process differs from the way the results are presented in a paper or thesis, which structures the research in order to try to make it more understandable for the reader

Requirements documents often include devices that attempt to reconstruct the actual process by keeping track of what went on during the construction of the document One way of doing this is by specifically numbering the versions of the document as it goes through various modifications and revisions Another way is by documenting the open issues that are outstanding at a particular point in the process, and by recording how the closed issues were resolved Similarly, actual changes to requirements previously documented are sometimes recorded, along with justifications for the changes made and decisions taken

Justification is an important part of the rational reconstruction of what went on in the actual process Parnas and Clements recommend using the documentation to record all design alternatives that were considered and rejected, and why Such justifications can be used to answer many questions concerning the requirements and design, that might be raised during development and, subsequently, during the entire life cycle of the system

The issues and changes that are documented are typically the ones that are considered the most important ones for the future lifetime of the system Recording every single change and decision would be not only impossible, but also unhelpful for the purpose Different versions of a document are produced along the way in time for specific deadlines, for the purpose of review and comment

9 5 2 Uncertain requirements

Bounded rationality helps to explain why the system development process is not entirely rational Requirements statements are by necessity incomplete and imperfectly understood, even at the end of a development project Lehman formulated this as an uncertainty principle *"The outcome of software system operation in the real world is inherently uncertain with the precise area of uncertainty also unknowable"* (Lehman, 1989)

Lehman bases this principle on Godel's completeness theorem which says that a set of axioms must always be either inconsistent or else it must be incomplete He argues that in any development project, we cannot know which of the theories embedded in the system or used for its development is inadequate, because if we did, then we would consequently extend or improve it accordingly His solution is to define and follow a disciplined process that embraces the evolution of the software during use subsequent to its release

This characterisation of uncertainty as occurring at two levels is reflected in the nature of the learning process that is needed in order to deal with it An evolutionary learning process is one that includes second-order learning as well as first-order learning System development is such a learning process

The act of building a model could be considered a learning experience for the system developer, especially if the model is not to be used for communication with the client organisation or the users It allows the model builder to express his or her emerging understanding of what might be a very complex problem situation It also structures the learning task for the model builder

The process of discovering requirements may also be regarded as a learning process, with many changes along the way, as issues of understanding and disagreement arise and are resolved The practice of documenting issues and changes helps by keeping a record of why certain decisions were made, providing a knowledge base for future development and evolution of the system

Some methods explicitly recognise the nature of the learning process involved in system development STEPS (Floyd et al , 1989) is a systematic yet flexible method using a cyclical development process that accommodates various forms of representations, including prototyping, and different steps depending on the situation It emphasises participation by the users at all stages and "*intertwines development with use* "

9 6 Documenting Agreement

Recorded agreement is perhaps the most unusual component of the documentation model presented in this thesis Agreement is widely acknowledged as an important concern in requirements, but maybe not so much as a component of the documentation Its place in the documentation model is pivotal It stands for the varieties of agreement as expressed in the document information, but also expresses the varying nature of what the agreement is about Agreement rarely refers to problem domain descriptions, or application constraints, but it could do It typically refers to goals, required effects, and proposed solutions, and most of all, to changes proposed to deal with issues that were raised in the course of defining the requirements in the document Open issues are explicitly those not yet agreed Solution constraints may or may not be The document keeps track of the status of the agreement

Participatory design emphasises the importance of social, cultural and political factors in **getting to agreement** It recognises that system development often involves a redesign of the users' work context and that this could lead to conflict It aims to coordinate joint action by developers and end users so that each will gain a better understanding of each other's point of view The main idea is that, through sustained interaction, developers gain a better understanding of the users' work context, and users gain more influence over the outcome, so that a better outcome can be achieved

A collection of articles entitled "Representations of Work" (Suchman, 1995) reported on various user-centred approaches to system development modelling in organisational contexts These approaches aim to increase understanding in collaborative efforts at system development, while recognising the problem of perspectivity, and the difficulty of "how to construct a shared information space," given the complex social and political relationships that may exist between stakeholders However, not all system development situations are similar to the ones for which participatory design approaches are intended, and in those situations, agreement means different things

9 7 Summary

One of the main ideas which influenced the classification of requirements presented in this thesis is called the separation of concerns This is sometimes taken to mean that different elements (concerns) such as problem domain descriptions and required effects must be documented in separate documents, but this is not necessary As long as they are clearly distinguished in the text, these elements can be interleaved in the same document, as much as is necessary, to allow the required effects to be understood in the context of the problem domain description This is what I have observed happens in practice

Abstract models are very useful for describing the problem domain, and the proposed solution One of their uses is to embody knowledge, in the form of domain models and reference models The process of building a model may be a very worthwhile learning experience for the system developer, even if the model is not used for communication with the client organisation or the users It allows the model builder to express his or her emerging understanding of what might be a very complex problem situation It also structures the learning task for the model builder

The relevance of problem frames and system architecture to requirements in the literature suggests that problems and solutions are 'inevitably intertwined ' But this conclusion is still controversial In practice, it is not possible to totally separate problems from solutions, and therefore, proposed solutions are often an intrinsic part of requirements documentation

The use of models, plans and other written representations is seen as an essential strategy for coping with uncertainty in many organisational situations, including system development People use documents and models as resources when doing knowledge-intensive work in organisations Models must be interpreted in order to be useful The interpretation depends on many factors including the context in which they are used

9 8 Conclusions

A model is a product Building it can help a practitioner to capture and improve his own understanding of a problem domain or the behaviour required of a system It is not necessarily a good method of communication yet models of various kinds are the tools most strongly advocated in research papers and textbooks, being almost synonymous with requirements representation Elements of the requirements classification, such as Problem Domain Descriptions, Required Effects, Proposed Solutions, and to a lesser extent, Goals and Constraints, are well established in the literature on requirements representation Much of this literature emphasises modelling, and reflects the prevailing product perspective

The other two elements of the classification, Issues and Changes, and Recorded Agreement support a process-oriented perspective The system development process is not entirely rational, and the different uses of requirements documents found in practice, including its role in rational reconstruction, also reflect this process perspective on system development

The rational perspective on organisations assumes that goals are fixed, while the open system perspective allows that over time, the system, and possibly its goals, will change, depending on the experiences of the organisation and its participants System development is a good example of learning in an organisational context The organisation changes itself as a result of implementing a new system, often in response to the challenges in its environment, or to changes in its goals

Both the product perspective and the process perspective are needed in order to comprehend the variety of different roles of the requirements document in different situations

Part Four Evaluation

This thesis has proposed a framework which attempts to relate the relevance of seven different typical components of requirements documents to a variety of types of situations It also provides a scheme to classify situations based on the presence and relative significance of eight different root concerns which are the sources of substantive requirements In the next chapter, I try to position the framework in relation to the existing and current literature on theories, methods, and standards for documenting requirements

In the final chapter, I summarise the findings of this research and try to assess its implications for both research and practice This chapter includes a summary of research, a summary of the findings, the implications of these findings for practice, for research, the contribution to theory, and possible applications of the framework It also looks at the limitations of the framework, the need for and suggested approach to validation, and recommendations for further refinement of it

Chapter 10 Comparing and Relating the Framework to the Literature on Requirements Documentation

10 1 Introduction

The literature on requirements documentation is extensive, so this review must be selective, but it attempts to be representative, nonetheless In order to organise this chapter, and to justify the selection of literature reviewed, I will present a grid (introduced in Table 1) which analyses the contributions according to their position in two dimensions The first dimension analyses contributions depending on whether they belong mainly to the product-oriented perspective or to the process-oriented perspective The second dimension indicates whether a contribution seems to interpret the term "requirements" as substantive needs or as written texts and/or representations This latter distinction is not normally made in the literature, but which interpretation is in operation can be inferred

	Product-oriented perspective	Process-oriented perspective
Requirements as needs	Q1 Contributions that regard requirements as needs and their identification mainly as an end in itself	Q2 Literature that regards requirements as needs and their identification as part of a learning process
Requirements as texts	Q3 Authors that look on requirements as artefacts, and these or the system as ends in themselves	Q4 Papers and books that look on requirements texts as 'situated tools' in the process of evolving useful systems

Table 10 1 Analysing the literature

The process-oriented perspective was described and motivated by Christiane Floyd in (Floyd, 1987) A good case for adopting this perspective can be found in the work of Lehman (Lehman, 1984, 1989, Lehman and Belady, 1985) who observed that software systems change the environment in which they are used, and thereby modify the requirements that led to them

10 2 Classifying requirements

Several authors and standards provide a typology of requirements All of these classifications fall into quadrant 1 They adopt a product perspective rather than a process perspective, and either they do not distinguish needs from representations, or they clearly focus on requirements as needs

It is widely agreed that requirements are either functional or non-functional Although this distinction has been disputed, because non-functional requirements are often implemented as system functions However, the distinction is well supported in textbooks and standards, though the term "extra-functional" has been proposed as an alternative to non-functional Beyond this distinction, there is very little agreement in the literature regarding the classification of requirements Non-functional requirements are most often sub-divided into different categories such as

- performance requirements
- interfaces
- design constraints

But this breakdown is disputed by those who reject the notion of including constraints in the category requirements, although their inclusion as requirements is commonly accepted Textbooks on SSADM recognise a breakdown that distinguishes constraints from non-functional requirements, for example, the following list comes from Ashworth and Goodland (1990)

- functions
- data
- non-functional requirements
- goals
- implementation/design constraints

The IEEE Standard Glossary of Software Engineering Terminology divides requirements as follows

- functional
- performance
- interface
- design
- implementation
- physical

Whereas, the IEEE Standard for the Software Requirements Specification 830 (IEEE, 1998) provides a list which, in addition to functional, performance and interface requirements, includes

- operational requirements
- resource requirements
- verification requirements
- acceptance requirements

- documentation requirements
- security requirements
- portability requirements
- quality requirements
- reliability requirements
- maintainability requirements
- safety requirements

That is, fourteen types of requirements, consisting of functional requirements plus thirteen different types of non-functional requirements.

Similarly, the Robertsons (Robertson and Robertson, 1999) recognise seventeen different types of requirements. One reason for this emphasis in the literature on varieties of non-functional requirements may be that they occur across different kinds of systems, whereas functional requirements are more specific to particular types of systems, and therefore do not lend themselves so easily to categorisation across systems. Despite their importance in the literature, non-functional requirements do not weigh heavily in practical requirements documents, as far as I have observed, and are therefore not considered specifically in the framework.

Functional requirements do not seem to lend themselves to classification in the same way, though in practical situations the actual functional requirements are always sub-divided according to their specific contents.

10.3 Changing requirements

It is often noted in the literature that requirements do not remain static, but change over time. This phenomenon is often called the volatility of requirements, and has long been seen as one of the major challenges in software development; see for example, (Curtis et al., 1988) or (The Standish Group, 1995). Responses to this problem have concentrated mainly on tools for requirements management, such as DOORs, Requisite-Pro, etc. These provide mainly database functionality whereby specific requirements statements can be stored, subjected to change control, assigned values for specific attributes for priority, status, etc. and extracted into documents according to various search criteria.

There are two main reasons why requirements change. Substantive requirements do not exist a priori, but emerge from discussions and problems and are jointly constructed by the stakeholders in a situation. During this process, requirements are extremely volatile, especially if they arise from a variety of different types of stakeholders Requirements management tools do not support this kind of change, which is a necessary part of the process of requirements Issue-based tools which capture design rationale are more suited to this because they keep track of changes, rather than control them Such tools, being process-oriented and regarding requirements as needs, belong to quadrant 2 of the table

Even after the requirements have been documented and agreement has been reached, requirements continue to change, because reality changes, and understanding changes These are the changes that need to be controlled Some of the changes may be seen as necessary and desirable, while other changes may be deemed undesirable, in order to maintain control over what is called often called scope creep or requirements creep Requirements management tools are used to provide the control needed over this kind of change They are product-oriented, and directed to the management of requirements statements, so they correspond to quadrant 3

10 4 Representing requirements

The IEEE guide gives a list of quality attributes of requirements, a contribution which belongs to quadrant 3, being both product-oriented, and representation-oriented According to this list, requirements (statements) should be unambiguous, complete, verifiable, consistent, modifiable, traceable, and usable after development These desired qualities take no account of the different uses to which a requirements document might be put, which was one of the main themes of the present research For example, absence of ambiguity is not always an important consideration

Almost all requirements documents are written to some extent in natural language, which may be used in ways that allow room for different interpretations, leading to fruitful discussions However, the inherent ambiguity of natural language is often cited as a reason for documenting requirements using formal notations, although these have limited applicability especially in the early stages of requirements analysis Similarly, completeness is not always seen as a necessary attribute of requirements statements, in many situations Desirable qualities must always be considered in light of the purposes and uses of the documentation, which in turn depend on the situation The framework of situations includes some consideration of these purposes and uses

Modelling requirements is an important strand of requirements research, and many different kinds of modelling techniques have been proposed Although the IEEE guide does not specifically deal with the modelling of requirements, it provides for the inclusion of semi-formal and formal models to supplement the natural language representations It also warns that care should be taken to distinguish between "the model for the application and the model for the software" Unfortunately, many of the classical and even the modern modelling techniques in use fail to make this distinction Making distinctions between representations has been an important aspect of the documentation model presented in this thesis Representations that describe the problem domain are different in purpose from those that describe the requirements per se and from statements that propose elements of the solution, reflecting the different roles of the requirements documentation in different situations

Jackson (Jackson, 1995) gives an authoritative classification of different types of descriptions

- Definitions, which are neither true nor false, but must be complete
- Designations, which are rules for describing observable phenomena in the problem domain Like definitions, designations are neither true nor false, but define the terms used in other descriptions
- Refutable descriptions, which are either true or false, concerning the problem domain

This classification fits into quadrant 3 in the table, as it is concerned with representations as different kinds of products

Following Jackson, Kovitz (Kovitz, 1999) gives a breakdown of different kinds of requirements associated with different kinds of problem frames Like Jackson, he distinguishes the requirements (document) which describes the problem from the specification which describes the solution He gives a list of contents of a requirements document (quadrant 3) to include requirements, problem domain description, expectations, preferences, invariants, platform, global characteristics, likely changes, glossary, overview, and document information

According to Kovitz "*Requirements are effects that the computer is to exert in the problem domain by virtue of the computer's programming*" These correspond to what would normally be called functional requirements Kovitz breaks them down into queries, behavioural rules, mappings, operations on realised domains and correspondences between domains (Quadrant 1)

This approach explicitly recognises the distinction between the problem domain and the requirements per se, which is often glossed over in other approaches Jackson recognises the need for three different descriptions, and warns against confounding them

1 The description D which true only of the problem domain,

2 The description R which is true of both the machine and the problem domain, and

3 The description S which is true only of the machine

"To develop software is to build a machine, simply by describing it" (Jackson, 1995) To try to achieve this with only one description is to risk confusing the things in the problem domain with their analogues in the machine The requirements refer to things in the problem domain, and only make sense in relation to those things The machine must ensure that they are satisfied

As far as Jackson's theory is concerned, the machine is the solution, and the requirements, while true of the machine, are not about the machine, but about the problem domain Many authors including Jackson argue against including solutions in a requirements statement In practice, in many situations, it is common to incorporate proposed solutions along with the requirements in a requirements document This is reflected in the documentation model proposed in the thesis

10 5 Domain specific approaches

Although generic approaches abound in requirements engineering, a number of efforts have been made to target particular requirements techniques to more specific domains of application

For example, some requirements modelling techniques, such as Statecharts (Harel, 1987), might be regarded as more suited to modelling process control systems than other kinds of systems, for example, business transaction systems But this may not be a useful distinction, at least as far as modelling is concerned (Wieringa and Jansen, 2001) It has long been recognised that state transition diagrams are quite useful for modelling user interaction with graphical interfaces, for example

Pamela Zave in a number of early papers on PAISley, a language for state-based modelling, popularised the term "embedded system" for the software controlling a piece of hardware in which it is embedded This definition conflicts with Lehman's definition of an E-type system which refers to a system embedded in its environment (Lehman and Belady, 1985) Most systems of any importance correspond to Lehman's notion of embedded systems

Harel (Harel, 1987) introduced the term "reactive system " A reactive system is defined in (Wieringa, 1996) as a system whose response to an input depends on the past history of inputs as well as the current input This definition also covers most application systems of any significance in everyday use, including systems that react to user input from a Windows interface

Similarly, Peter Wegner has characterised many of today's systems as interaction machines rather than algorithmic machines (Wegner, 1997) The user does not simply provide input to a computation, and wait for the output, as in the algorithmic model, but engages in a conversation with the machine, allowing a larger class of problems to be solved That this model of computation can be seen as a new paradigm is clear from the trend in Artificial Intelligence, for example, away from input and rule-based computation towards agent-based interaction It seems that previous distinctions between application domains no longer apply, and therefore that domain specific requirements techniques do not make much sense in practice

Many good examples of targeted techniques may be found in the domain of information systems, which has a relatively long tradition of systems analysis techniques and system development methods Information Algebra, Systematics, PSL/PSA and the NCC Standards all predate SSADM, by decades However, much research within this field has reported that prescriptive methods are not used in practice, at least not in the ways intended by their authors, as practitioners prefer to use selected techniques in a toolbox or a la carte approach

One response to this can be seen in research efforts in the field of method engineering, which aim to provide support for situated methods (Brinkkemper et al , 2001, Ralyte and Roland, 2001) Such approaches enable the selection, amendment, and assembly by the methods engineer of different techniques or "method fragments" to suit the requirements of a particular situation However, they have no apparent theory of what a situation might be

In a paper contrasting the arguments and pressures for formalized development methods with the arguments and pressures against such methods, Brian Fitzgerald identified three main pressures for new development approaches the changing nature of business environments, the changing profile of the development environment and the pressure for rapid delivery (Fitzgerald, 1996) These characteristics also describe the influences which have some bearing on the ways that requirements need to be documented in different situations, as depicted in this thesis

The evolutionary delivery process advocated by Gilb (Gilb, 1988) reflects the evolutionary nature of software characterised by Lehman in (Lehman, 1984, 1989, Lehman and Belady, 1985) Recent research at the Harvard Business School has confirmed this trend towards evolutionary development in which early delivery of the evolving new products to customers combined with rapid turnaround of new releases resulted in more successful products than the more traditional development process (MacCormack, 2001) In these market-oriented situations, requirements come from many different sources, and competing requirements must be prioritised for each new release Several new techniques have been proposed for evaluating the relative costs and values of requirements in these situations

10 6 Contingency theories

Contingency analysis comes from the field of organisational theory, where it is characterised in the following principle "*The best way to organise depends on the nature of the environment to which the organisation relates*" (Scott, 1987) The more uncertainty in the environment, the more information is needed before deciding on any particular course of action This approach has been applied to the selection of requirements analysis techniques by some authors in the early eighties The resulting frameworks are called contingency theories

In an early paper, Nauman et al (1980) identified a list of contingency factors which might affect the level of uncertainty concerning requirements in a given situation These include Project size, structuredness, user task comprehension, and developer task proficiency The level of uncertainty was treated as an intermediate variable that in turn determined the requirements assurance process to be used (Naumann et al , 1980)

Gordon B Davis (1982) gives a similar set of contingency factors, covering characteristics of the object system the application, the users, and the analysts, and again, uncertainty is the common denominator that establishes the best strategy to use, in this case for determining information requirements (Davis, 1982)

Following on from this research, Wood-Harper and Episkopou (1986) present a framework for choosing between radically different problem-solving approaches, corresponding to different views of the world the Checkland Soft Systems Approach, the ETHICS participative design approach and the rational data analysis approach The contingency variables here are Characteristics of the Problem Owner, assessment of the problem content system (again, uncertainty is an important issue) and an assessment of the Problem Solver (to include cognitive style) They rightly criticise the earlier contingency theories as relying heavily on intuitive notions of uncertainty

They propose a tentative Approach Choosing and Matching System (ACMS) framework which analyses different problem solving approaches on the basis of ideology, tools, inquiry system, and costs in terms of manpower and time span (Episkopou and Wood-Harper, 1986)

The ACMS framework relies on an earlier framework by Jackson and Keys (1984) which outlines a classification of problem contexts This classifies problem contexts in terms of (a) the nature of the decision makers and (b) the systems in which the problem is located A review of the concept of "complex systems" is given, before settling on the concepts of mechanical and systemic problem contexts, which contain relatively simple and complex problems respectively Four different types of problem contexts are then presented

1 Mechanical-unitary
2 Systemic-unitary
3 Mechanical-pluralist
4 Systemic-pluralist

A set of decision-makers is unitary if they all agree on a common set of goals for the whole system, pluralist if they cannot agree in advance Agreement on goals is an important factor, whether it exists in advance or must be achieved

Each of the contingency approaches outlined above suggests how the given framework could be applied in practice, to guide practitioners in selecting appropriate strategies These frameworks can all be placed in quadrant 2 of the table, being process-oriented and regarding requirements as needs They contrast strongly with Michael Jackson's more recent proposals on problem frames, which do not allow for the contingencies of problem context, but instead suggest that the problem and the solution have a common structure which indicates the essential nature of solution task (Jackson, 1997, 2000)

10 7 Agreeing requirements

Negotiation is recognised as an important part of the requirements process in many situations, and there have been a few notable research attempts to support the identification and resolution of conflicts in relation to requirements The Win-Win environment (Boehm, 1991) allows "win conditions" for each stakeholder to be identified, and supports negotiation among the stakeholders to try to ensure that all the win conditions are satisfied Similarly, research on viewpoints has resulted in a trend away from requirements consistency towards managing inconsistency in requirements specifications (Hunter and Nuseibeh, 1998) These approaches fit in quadrant 2 and 3 respectively

Despite the general recognition of the importance of agreement in the requirements process, it is often omitted from the generally accepted list of activities that make up the process elicitation, specification, verification and validation Requirements validation therefore incorporates the achievement of agreement, but the different types and levels of agreement do not receive much attention in the current literature, although the topic of managing conflict and different viewpoints from a process point of view is well addressed (Robinson and Pawlowski, 1999, Sommerville et al , 1999)

In a special issue of IEEE Software devoted to requirements engineering, published in March 1996, guest editors Shekaran and Siddiqui declared that *"the requirements-as-contract model is dead "* This heralded a focus in the literature since then on market-oriented software development, aided somewhat by publications describing the development process at Microsoft However, not all market-oriented situations are mass-market situations, and contract-based development still thrives in the more specialised sectors of software market The idea of a contract is still alive and well within the context of solution-oriented situations, also

10 8 The organisational context of requirements

Jonathan Grudin examined the barriers to communication in different organisational contexts of system development He distinguished the organisational context of development from the organisational context of use, and identified three different contexts of development, the competitive contract, the internal or in-house development, and the commercial product His particular concern was with the possibilities for user interaction with developers in these three different situations (Grudin, 1996)

Curtis, Krasner and Iscoe set out to study the design process in large projects from an organisational perspective and found that individual characteristics as well as organisational factors determine success or failure of projects (Curtis et al , 1988) There are a number of parallels between this study and the present one It is a qualitative investigation, based on interviews with personnel in seventeen projects representing a range of different types of systems It focuses on requirements, and recognises the influence of organisational factors on the role of documentation in the requirements process It was concluded that
"Large projects required extensive communication that was not reduced by documentation " (Curtis et al , 1988)

They give a summary of the projects they studied which indicates that they were concentrated around real-time applications From this and many comments made in the article, it seems that most of the project situations would be classified as market-oriented in the framework presented in this thesis, although that term is not used in the article Some of them were in single-customer market situations, while others were producing "commercial products," corresponding to my classification of vertical market situation since internal groups such as marketing often acted as customer in these situations

Some of the projects studied were clearly in the targeted customer market type of situation, though again such terminology was not used in the article Requirements fluctuation was attributed to two causes The needs of a single customer changed over time, or the different customers had separate needs (Curtis et al , 1988) The authors do not distinguish between situations where the influence of a few customers is very strong and situations where the market contains many customers whose needs can be represented by a customer-facing marketing department Judging from the list of applications given for the seventeen projects, there were no mass-market situations represented in the projects

Their investigation set out to study the process of design, but found that requirements and problem definition issues presented the most salient problems throughout the development process For example

"The time devoted to learning and coordinating application-specific information was initially buried within the design phase and could not be accounted for separately" (Curtis et al , 1988)

Although it was an exploratory investigation, it was not construed as theory building The authors adopted a prior organisational model of software development at the outset, which they used to interpret and organise the findings they presented However, they did use their findings to refine the theory they set out with

"Rather, the information in our interviews forced us to account for differences among individual project members and to determine how these differences interacted with variations among situations" (Curtis et al , 1988)

Their conclusions emphasise the importance of learning, communication, and negotiation in the requirements process Rather than the idea of a collaborative process involving all the participants in a project, they highlight the role of influential individuals, or small coalitions of individuals, people with superior application domain knowledge, in the social construction of requirements

Though they do not refer to it explicitly, this kind of coalition is an example of the concept, first described by Cyert and March, of a loose and shifting dominant coalition that selects organisational goals (Scott, 1987, pg 107) Similarly, the power and influence that individuals exert in organisations is often thought to be related to the information that they have

The process by which dominant coalitions exert influence over project goals and design is very well described in the paper, especially the difficulty that competing coalitions might experience in trying to influence the design The fact that this happens even in market-oriented situations illustrates the challenge of implementing a process of participatory design in solution-oriented situations, where organisational boundaries and knowledge asymmetry could be much more significant factors than the expertise of individual participants

They also highlight the role of *"exceptional individuals"* at the centre of these coalitions, who had large amounts of application domain knowledge, and took on the responsibility for transferring this knowledge to the whole team This was made difficult by the existence of organisational boundaries, barriers to communication, which inhibited the transfer of the emerging knowledge, a situation compounded by the assumption that documents produced by one group conveyed all the information required by another group

They preface these conclusions with a remark that the problems that they highlight have survived for decades despite being described many times by previous authors However, they do not take the opportunity to use their findings to criticise the prevailing theories, then and now, about the role of documentation in the requirements process The report of this study is rich in detail, however, as well as highlighting a number of important issues and problems in the requirements process as it affects market-oriented situations

The concept of *"exceptional individuals,"* or their role in the development process, was not explored in the investigation reported in this thesis However, it was a working assumption that some practitioners are better than others, probably by an order of magnitude In seeking practitioners to participate in the study, I generally first approached another practitioner in the selected area of development, and asked that person to recommend someone who, in that person's experience, was exceptionally good at their work in relation to requirements This strategy located a sizeable number of exceptional individuals, who were able to articulate and communicate, not so much the problems of the requirements process but the practical realities of the situations in which they worked

Under the heading of "Fluctuating and Conflicting Requirements" Curtis et al give a number of sources of such fluctuations, which are essentially sources of requirements which correspond with the root concerns in the thesis Their sources of changes and fluctuations are organised as Market Impacts, Company Impacts and Hidden Impacts Market Impacts include Customers, as sources of customized needs, changing needs, add-ons, Technology, as a source of technology advances and competitors' products, as well as Regulation, in the form of regulatory agencies and Standards committees Company impacts include Approvals (Marketing & Legal, and Financial) and Technology (R&D results and other product lines) The idea of an evolving product does not appear in this investigation, despite the focus on market-oriented development The idea of a product release is briefly mentioned as a strategy for dealing with priorities among requirements, but the concept is not highlighted

10 9 Requirements frameworks

The seven typical situations depicted in the framework presented in this thesis can be reduced to three basic kinds of situations, called Market-oriented situations, Solution-oriented situations, and Problem-oriented situations, based on an initial distinction between problem-oriented and solution-oriented

There are resonances of this distinction in various strands of the software development literature Bruce Blum, in a long series of articles on a method called TEDIUM, conceptualised information system development as problem solving His later classification of specifications distinguishes between problem-oriented and product-oriented as well as formal and semi-formal (Blum, 1994)

Roel Wieringa (Wieringa, 1996) begins with a distinction between client-oriented and market-oriented development, while he regards both types of situations as pertaining to product development, (therefore locating this model in quadrant 1) However he does recognise that different types of stakeholder other than market or client, are possible, i e customer, sponsor, and user, and distinguishes the situation of use from the situation of development, though he continues with the client/market distinction in his frameworks

The concept of situation is given a considerable amount of attention in the literature, particularly since Lucy Suchman's influential book (Suchman, 1987) which re-conceptualised plans and other artefacts as tools to be used in situated action This seems to have evoked two different kinds of responses exploratory research describing the intractable difficulties involved in defining in advance what needs to be done in situations (e g the futility of defined methods) and endeavours such as method engineering which aims to allow the definition of situated methods to suit individual projects and situations Most of these do not attempt to explore how situations might be characterised, which is one of the things I tried to do in the framework

Linda Macauley describes a framework consisting of a number of different scenarios describing a variety of possible customer-supplier relationships, and for each scenario suggests a typical process model, and a number of requirements techniques Customers are defined as "persons who commission and/or purchase a product from a supplier" and suppliers are defined as "persons who develop the product and deliver it to the customer in exchange for payment" Because it focuses on different processes of requirements elicitation, this framework belongs in quadrant 2 of the table

Her scenarios are as follows

1 A customer issues an Invitation to Tender to a number of potential suppliers (1- m)
2 A supplier responds directly to a customer request (1-1)
3 A supplier makes a generic product which will meet the needs of a large number of customers (m – 1)
4 A supplier has a generic product which needs to be tailored to meet a specific customer need (1- 1)
5 6 and 7 are different variants of in-house development, depending on the internal organisation structure

Macauley's model has some similarity to the framework presented in this thesis it is situation-related, and it proposes different approaches depending on the situation The main differences are

1 Macauley's is a framework for determining the appropriate requirements process and elicitation techniques to be used
2 The situation types are defined simply in terms of customer-supplier interaction All other aspects are considered to depend on this relationship
3 Significant emphasis is placed on in-house development Three of the seven types of situation identified are different kinds of in-house development situations
4 Market-oriented development is regarded solely as mass-market and new product development There is no consideration of product evolution or smaller markets
5 The framework does not explore in any way how the different concerns that give rise to requirements can affect a situation

Perhaps the most comprehensive framework for requirements engineering is that proposed by Klaus Pohl (Pohl, 1994, 1996) This framework covers all four quadrants of Table 10 2, though it emphasises the process perspective over the product perspective It also caters very well for the diversity of practices and techniques in requirements engineering It defines a *three-dimensional framework* for understanding the requirements process, consisting of the following three dimensions which are envisaged as orthogonal

1 the specification dimension
2 the representation dimension, and
3 the agreement dimension

The specification dimension refers to the degree of completeness of the requirements specification The representation dimension refers to the formality of the requirements models, and the agreement dimension refers to the degree to which the stakeholders agree on the requirements specification Note that these are interpreted in terms of the requirements statements, rather than the requirements themselves However, the ideal requirements process is defined in terms of a curve within the three dimensions "*It should ideally end up with agreement on a well-understood and formal specification from which development can proceed*" (Jarke and Pohl, 1994)

	Product-oriented perspective	Process-oriented perspective
Requirements as needs	Q1 Classifications of requirements functional, NFRs Kovitz different kinds of functional requirements Wieringa market and client situations The three dimensions of RE	Q2 Curtis, Krasner and Iscoe Macauley customer-supplier relationships IBIS - issue based tools Capturing rationale The three dimensions of RE
Requirements as texts	Q3 IEEE Standard desirable attributes of requirements Requirements management tools Managing inconsistency The three dimensions of RE	Q4 The three dimensions of RE

Table 10 2 Analysis of the literature reviewed in this chapter

Chapter 11 Summary and Conclusions

"What we call results are beginnings "
(Ralph Waldo Emerson)

11 1 Summary of the research

This research has been concerned with the relationship between two aspects of requirements documentation variety and quality It has taken as a point of departure that a good requirements document is one that is "good enough" for the situation in which it is used Conventional approaches to defining the quality of a requirements document do not take this into account, despite the fact that requirements documentation is used in a wide variety of situations

I set out to explore the situations in which requirements documents are written and used in practice, in order to discover what are the situational variables that govern the need for different types of requirements documentation, and how they interact to affect what is appropriate documentation to use in different types of situations

The nature of this investigation was exploratory Rather than directly ask the question, "what is a good requirements document?" it aimed to open up the question, and ask about the different roles and purposes of requirements documentation in different contexts These contexts embraced the organisational context of use, the organisational context of development, the application domain, the contractual arrangements and other aspects of the relationship and interaction between the developer and the customer, client, or users

The empirical investigation focused on finding out how experienced practitioners construe the requirements documentation they use in their work These practitioners were drawn from a purposefully wide range of backgrounds and industries, in order to capture as much variety as possible It took the form of a series of semi-structured interviews, the transcripts of which were analysed according to the principles of grounded theory A computer-based tool, ATLAS ti, was used to organise the emerging theory As is common in grounded theory analysis, the data collection and analysis phases took place in parallel, with the later interviews reflecting a purposeful sampling procedure to develop and refine the insights gained in the analysis of the interviews to date

Although this research is focused on the product(s) of requirements definition, it has adopted a process-oriented perspective which has been incorporated into the results Therefore, I have come to look on the requirements document, not as an end in itself, and not even as a means to an end (which is the system looked on as a defined product), but as a device that is instrumental in a process This is the process of discovering, learning, agreeing and managing the emerging requirements and other ancillary information needed to describe an evolving system embedded in a situation

The research took place over the course of several years, which permitted plentiful time for reflection on it and its goals and purposes In accordance with the canons of the grounded theory approach, the analysis and the data collection proceeded side by side The process is described in Chapter 3 It occurred in three main stages The pilot stage resulted in a re-framing and focusing of the research question, on an attempt to understand the organisational contexts of different development situations The Blackbird case study phase focused on different uses in a range of similar situations and resulted in different types of requirements statements becoming a focal point, and hence the idea of the relationship between variety and quality in a requirements document The third stage concentrated on variety of both situations and documentation, and on finding able informants in as many different types of pertinent situations as possible, in order to elaborate and extend the emerging framework

11 2 Summary of the findings

The findings are presented as four elements of an overall framework, relating different types of requirements situations to different types of documentation

1 A taxonomic analysis of themes of variation in requirements documentation This is based on the interview data and sample documents obtained during the interviews The result of this analysis is a tentative model of requirements documentation, which condenses the variety of statements found in practice to seven different types of descriptions or statements These are ***Problem domain descriptions*****, *Statements of the required effects*, Proposed solutions, Recorded issues and changes, Defined goals and objectives, Specified constraints**, and **Recorded agreement**

2 An ontological analysis of requirements as needs, presented as an ontology chart As an ontology, this is not complete, but it is grounded in the interview data It represents the different substantive concerns that are associated with requirements, as distinct from the semiological aspects, or signs, that are the written down requirements It deals not just with requirements, but also with several other matters of concern in the requirements process, such as **goals**, **customers**, **users**, **domain knowledge**, **business processes**, **tailorable packages**, **products**, **platforms**, and **markets** It reveals eight root concerns from which all other concerns and

requirements flow **Project, Users, Client Organisation, Business Area, Application Domain, Customer, Product, and Market Opportunity**

3 A classification scheme for requirements situations Situations are characterised in terms of how the eight root concerns influence and contribute to the requirements

4 A situation-based classification of requirements documentation Each type of situation is associated with a particular style of documentation, based on the model derived from the taxonomic analysis

11 3 Implications of the findings

11 3 1 Implications for practice

Requirements practitioners work with both substantive requirements and their representations Both are equally important, but it is vital to be aware of the distinction, especially when adopting new practices or techniques When requirements practitioners ask about or discuss "requirements metrics," for example, do they mean measures of how many actual requirements are in the project (such as function points), or are they talking about measuring the quality (such as consistency, or completeness) of the requirements statements in the document?

Whichever sense we are talking about, requirements cannot exist on their own They depend on prior things that are of equal concern to the requirements analyst These concerns are not always the same, they vary depending on the situation The ontology presented in Chapter 5 of this thesis is an attempt to chart those concerns

For practitioners, working in a specific kind of situation, the value of an ontology such as this is that it enables them to position their "locus of concerns" appropriately and select the requirements techniques and processes appropriate to those concerns Experienced practitioners, such as the ones who participated in the investigation, are already implicitly aware of where they are situated Less experienced developers may benefit from having these things made explicit

The documentation model presented in Chapter 4 could be used as concise checklist against which commercial tools for requirements management, or the existing practice of documentation in a company's requirements process, could be evaluated

In practice, application domain descriptions are typically not included in the requirements documentation, in many types of situations, and this practice is reflected in the relevant documentation styles and profiles in Chapter 7 One reason for this omission in practice may be the lack of suitable representation techniques, but another possible explanation may be the need to reduce the number of different types of statement, for example to control the scale of the document in situations where the documented agreement is the basis of a formal contract

Even in these situations, the advantages of modelling the application domain are clearly obvious, although the practical pressures of contract development may not allow for such apparent luxuries Nevertheless, the documentation profiles reflect the situation as it is, not as it should be, or might be

11 3 2 Implications for research

The implications for research are probably more significant than the implications for practice, and so, the results may be more relevant to the research community than to the practitioner Theory which is grounded in practice may help to bridge the gap between the research and practice of requirements engineering *"Efficient practice precedes the theory of it"* (Ryle, 1949)

Requirements engineering now deals with a wide range of development situations, and includes an increasingly wider variety of techniques, from formal methods to contextual approaches An ontology such as the one presented in Chapter 5 could be used as the basis of a reference model for requirements engineering research Any proposed RE method, tool, or technique could be positioned in relation to the reference model For example, techniques that are intended for modelling the substantive aspects of requirements could be designated as such There seems to be a need for such a reference model

The best example of such a reference model that I have come across is Klaus Pohl's conceptually simple three-dimensional model of RE (Pohl, 1994) However, despite that article being continually referenced in the literature since it was published in 1994, the reference model it proposed has not had much of a perceptible impact or real influence in other people's contributions that have been published in recent years A more complex reference model, such as that of Jarke and Ramesh (Jarke and Ramesh, 2001), is therefore unlikely to be adopted widely outside the area of sophisticated repository management tools for requirements traceability

The documentation model presented in Chapter 4 is fairly simple, but it makes an important distinction between different aspects of a system that might need to be represented, unlike many popular notations for modelling 'the real world ' The real world in these cases often means the proposed solution as well as the problem domain Using the same technique for both may lead to confusion Models of the problem domain may need to allow for a multitude of perspectives and a toleration of inconsistency not appropriate in a model of the proposed solution Different criteria for approval and agreement apply to different documentation components, such as goals and constraints

The ontology and the documentation model respectively deal separately with the substantive and semiological aspects of requirements This separation is not often found in the literature, an oversight which can result in misunderstandings and the inappropriate use of models For example, the use of Object Oriented Analysis can lead to confusion between substantive and semiological objects Customer object and Customer-record object, for example This is compounded by the fact that because it uses the same notation as Object Oriented Design, which entails the modelling of semiological objects in the main Object-oriented methods are based on a very simple ontology of the world which is more appropriate to the solution domain than the problem domain An ontology of requirements that explicitly recognises the difference between the mainly substantive problem domain and the more semiological solution domain would be a useful contribution to this particular "separation of concerns "

Research on requirements quality metrics often begins with such things as completeness and consistency as essential qualities of all requirements documents, but there is no evidence that these are important factors in some situations, for example, those in which organisational and business objectives are major concerns, the different types of solution-oriented situations, particularly the in-house solution situation It should be recognised that different situations require different standards of completion and consistency, to name just two criteria, in the different components of documentation that they use

It is likely that the validity of such criteria would vary not just between situations, but also across the various components of the documentation styles that they use For example, different standards of completeness and consistency would apply to goals and constraints, within a situation, and different patterns of agreement on say goals would apply in different situations

Similarly, the applicability of other kinds of research could vary between situations and across profiles Examples include The use of ethnographic methods for requirements elicitation, the use of formal methods in (a) specifying requirements and (b) modelling application domains and proposed solutions, the use of techniques to measure the relative value and cost of requirements in order to prioritise them These are all more relevant and appropriate in some situations than in others A requirements-ontology-based reference model could help to identify and indicate the most appropriate area of application of different kinds of research

11 3 3 Contribution to theory

The theoretical framework has its roots in a body of previous theory on organisational behaviour and knowledge-based work in those kinds of situations These include contingency theories, bounded rationality, agency theory, symbolic interaction, the social construction of reality, and the situatedness of artefacts such as plans and documents As such, it is an extension of those theories, in this case helping to explain the "requisite variety" of documentation styles and their uses in different situations

Is this a contingency theory? Maybe not, a contingency model is capable of being applied, in a range of situations, along with an explicit rationale for its application Davis, and Episkipou and Wood-Harper among others, provide contingency models, for choosing appropriate methods (Davis, 1982, Episkopou and Wood-Harper, 1986) My framework proposes to be more descriptive than prescriptive

According to Frances Bell, a contingency framework is more prescriptive than a normative model She gives as an example of a normative model, the NIMSAD framework for system development methods, which allows "*a range of concepts, models, and techniques to be clarified, compared, categorised, evaluated, and/or integrated* " (Bell, 1996) By this definition, the framework presented in this thesis is a normative model, since it allows requirements situations to be compared and categorised, and techniques to be evaluated for use in different types of situations Suitably validated, the framework could also be applied in a prescriptive fashion, however

Much of the literature on software engineering, including requirements engineering, is prescriptive rather than descriptive in nature Much of it aims to improve the practice of software development This reflects the status of software engineering as a pre-scientific rather than a recognisable scientific discipline There are wide gaps between theory and practice There are also many gaps between different branches of the theory of software engineering

According to Thomas Kuhn, scientific communities go through different stages Pre-scientific knowledge is characterised by competing points of view within a discipline, rather than a unified body of theory Paul Wernick demonstrated that the disciplinary matrix underlying computer-based software development indicated a wide range of different points of view and belief systems (Wernick, 1996)

Not only do different schools of thought exist side by side within software development, but also different interpretations of what is going on Communication problems are not helped by the often empty terminology that abounds in software development Some terms used in relation to requirements have so many interpretations as to be worthless for use in a framework such as the one presented here "user requirements", "system requirements", "model", "specification", "design", etc

For these reasons, it is difficult to effectively position the framework proposed in this thesis as a contribution to the overall body of theory in the requirements literature It may be more effective to pick out some "schools of thought" within the literature that it challenges, and some others which it supports

There is a school of thought that says requirements documents should be always self-explanatory, and not subject to interpretation However, there is only one situation in the framework that supports this position, the problem-oriented situation A problem-oriented requirements document ought to be self-explanatory, but only to the extent that it allows the solution providers to propose a solution, not to implement it without further ado In all other situations a requirements document cannot be stand-alone

Another widely held belief is that the purpose of the requirements process is to produce a requirements specification This looks on the requirements document as an end in itself However, this research found that in many situations, the requirements document is a means to an end, and not an end in itself In some situations, its role in the process of getting to agreement was found to be a more important consideration than whether its contents were correct, or complete, or fully understood

The idea that a requirements document is first and foremost a model is also challenged by the results of this research If a model is defined as an abstract representation of the real world, then this is not the purpose of a requirements document, which has many different purposes in the establishment of agreement on what, why and how a proposed system is required to be

One of the most important of these roles is its role in the learning process which takes place in the context of system development The joint construction of a document in the course of a process that includes meetings and reviews, where much of the work of the social construction of agreement takes place, to be rationally reconstructed in the next version of the document, is typical of several of the situations described in the framework This builds on and adds support to the various segments of existing theory that view requirements engineering as a learning process, and extends those strands that regard plans and other documents as resources in situated action within an organisational context

11 4 Applications of the framework

The applications of the theoretical framework which was proposed in this thesis may be summarised as follows

The documentation model could be used as a checklist, for example to evaluate requirements management tools, or to assess an existing or proposed requirements documentation template, or to examine a sample set of requirements documents For example, it could be used to estimate the relative occurrence of different types of statements in documents originating in similar or different types of situations

The ontology may be used to focus on requirements as such, as well as their many sources, rather than representations of requirements It may be used to establish the locus of concerns for a particular project or type of project, or situation, in order to select appropriate techniques

The situation profiles could be used to classify a project situation, or a set of projects, with respect to the mam concerns of that situation, and the documentation needs entailed by those concerns The set of situation profiles could be extended and refined on the basis of further empirical examples

The documentation styles themselves are intended to be explanatory rather than prescriptive They serve to clarify how the different types of statements are used to different extents and purposes in different situations For example, it is not often recognised that "agreement" in a requirements document is something that varies with the situation, but it does, and this variation is clearly outlined in the documentation styles for different situations

11 5 Limitations

It could be argued that the practitioners on whom I relied for the empirical data that supports this thesis were communicating not their actual practice, but the way they think they work in practice This is the difference between espoused theories and theories-in-use It is a risk I have had to live with Given the long duration of the requirements process in most situations, it would be extremely difficult to directly observe theories-in-use in requirements documentation, and even then, these theories would have to be inferred In addition, such a strategy could not have allowed the research to take in sufficient cases to achieve the wide variety of situations needed to support the framework

11 5 1 Need for validation

The framework needs to be validated Research validity is often divided into two aspects internal validity and external validity Internal validity in qualitative research is often called credibility (Lincoln and Guba, 1985) Credibility of research results should be supported by the research process used, and I think that I can claim this for the process that I used

Testing or confirming hypotheses is an integral part of the grounded theory method of analysis It is built into every stage of the process It begins with open coding The proposed code or category is tested against the particular fragment of data, to see if it fits If so, then that code is linked to that piece of data, if not, then some other code, or variation is constructed In this way, the open coding process generates codes that apply to the data Constant comparison of codes to data is a way of continuously testing the groundedness of the data This is one of the particular strengths of the grounded theory method

The second level of coding in grounded theory, called axial coding, is where relationships between existing codes are constructed and tested against the data In my research, the main relationships are

- the ontological precedence relationships between the substantive concerns and
- the taxonomic relationships between the written components of the documents

The key idea in axial coding is to test the proposed relationships against the cases in the data

In grounded theory, the final stage of coding is called selective coding Compared to the earlier stages, it is less concerned with the data, and more with the codes from the previous stages, i e with the strands of the emerging theory In my project, this is the stage where I put together the proposed typology of requirements documents I can enumerate instances of each of the types in the data, but that is not the point I need to be able to show that the particular aspects (the root concerns and the types of statements) that I am proposing are the significant discriminators between the types What matters is not what they are called, but how to discriminate between them

This external validity is where the framework most needs validation or confirmation, and the key word here is applicability The typology is fairly general, as it is supposed to be, so the point is not whether one can generalise from it, but to test it against reality to see if it applies

This would not be a new departure, so much as an extension of the research Kathleen Eisenhardt (1989) has written a valuable article on building theories from case studies, from a positivist rather than a naturalistic perspective In it, she relies on the concept of saturation from the grounded theory literature as a criterion for when to stop, both for when to stop adding cases, and when to stop iterating between theory and data Eisenhardt's article highlights the tension between confirming the theory and extending it in the same step

Some people might argue that any theory must be a set of propositions, that are either true or false, and that these must be stated in a way that is falsifiable Not only that, but the experiments or trials should set out to falsify the whole thing My position on this is different, and corresponds to the pragmatist view of knowledge The question that must be asked about any model, theory or categorisation, is not whether it is true or false, but whether it is useful That is essentially what I think needs testing, the usefulness or otherwise of the framework

Alternatively, it is possible to paraphrase each situation profile and each documentation style as a set of propositions Looked at in this way, each pair of these represents a hypothetical situation and proposes a given style of document The point of doing that would be to present the framework as a set of propositions that are stated falsifiably The way to test them would be to look for numerous instances of each of the seven situations and find out if the documents used approximated in some measurable way to the typical document style I have identified This would be a major research project in itself

A more practical approach to establishing external validity would be to devise a set of trials to test the applicability of the framework The first trial could be based on a set of scenarios, composed from a sample of the cases in the existing data, along with some fictitious ones scenarios The second trial could ask the participants to supply their own real examples The scenarios trial could be presented in the context of a focus group of practitioners in the requirements field For each scenario, the participants could be presented with a set of multiple-choice questions asking them to pick out the most significant concerns related to that scenario, and then to choose the most important components of the document As a focus group they could choose to discuss or not discuss the questions, some of them might be quite straightforward, and others may be problematic The discussion could be quite interesting

If the framework is useful, then the feedback from these trials would cluster to some extent, showing that the variables in the theory are significant/useful/important in discriminating between different situations The emergence of further types of documents from the validation process, would not invalidate it, but strengthen the theory Because it is feature-based, the classification allows room for growth, expansion, development, and evolution It does not need to be right the first time

Whatever the result, it will be useful to have another source of data, either confirming or refining the theoretical framework This will give me some triangulation, which is another important consideration in establishing the validity of any theory, whether it originates in qualitative or quantitative research

11 5 2 Need for refinement

The major limitation of the framework is its status as a theory, yet to be confirmed by quantitative evidence It is a first step towards finding out the extent to which the requirements documents that are used in practice vary depending on their situation, by attempting to define how situations vary, and to define how the major documentation components might be distinguished However, these are by no means final and will need to be revised/refined continually

To this end, and with this in mind, both the situation profiles and the documentation profiles lend themselves to being revised in the light of further empirical evidence For example, the addition of a profile for the mass-market situation would fill an obvious gap in the set of profiles This type of situation was not represented in the situations that were investigated empirically for the thesis

An additional root concern, if it were identified in future research, would be accommodated in the situation profiles quite easily, and to a lesser extent also in the ontology chart A refactoring of the documentation model could similarly be adopted without much trouble in the document styles This ability to refine and adjust the framework should be considered a strength rather than a weakness

11 6 Recommendations for further research

These fall into two main types, recommendations for further qualitative work to extend the framework, as outlined in the previous paragraph, and proposals for applying the framework in quantitative research on the practice of requirements documentation These would proceed in different directions, one more focused on integrating and deepening the theory, the other having more focus on applying specific aspects of the framework

11 6 1 Extensions to the framework

The research has only touched on the idea of the requirements process as process of social organisation and knowledge construction A qualitative study of the practice of recording issues and changes in a requirements document would develop this further What gets left out? How do dominant coalitions use the requirements document to influence the course of changed requirements in a project? How does rational reconstruction operate in the process? The main idea behind the framework is the relationship between variety and quality in the practice of requirements documentation Ways to extend the framework to better account for the variety of different situations have been suggested above Ways to extend the framework to elaborate and expand on the idea of quality would include

Qualitative studies focusing on the way practitioners construe the quality of different types of requirements statements Are there any differences in the way they perceive quality in problem domain descriptions, for example, compared to statements of goals, or required effects? How do acceptable proposed solutions differ from unacceptable instances of a "*feature in search of a requirement*"?

11 6 2 Applying the framework in quantitative research

Research efforts that would continue this work in a quantitative mode would need to focus on specific issues, such as agreement, or on components of the documentation model, such as goals, for example, in order to examine and test the implications of the framework in a systematic fashion One survey that really needs to be done is to catalogue the different types of documents used in practice how to recognise them, given the different names, contents, uses, formats, etc This could be done using the documentation model of Chapter 4 as a point of departure Each type of statement would need to be given a kind of "operational definition" of how to recognise it in a document Such a survey could attempt to measure the occurrence of different kinds of statements in the requirements documents representing different types of situations, and test the following kinds of hypotheses

- The more formal the agreement, the fewer types of statements in the document
- The more formal the agreement, the more complete those statements are
- The less formal the agreement, the more types of statements included
- The less formal the agreement, the less complete they need to be

References

Argyris, C , Schon, D , 1978 *Organisational Learning A Theory of Action Perspective* London Addison Wesley

Ashworth, C , Goodland, M , 1990 *SSADM A Practical Approach* Maidenhead McGraw Hill

Basili, V R , 1996 Editorial *Empirical Software Engineering Journal* 1 (1), 1-3

Bell, F , 1996 Evaluation of the MultiView Methodology using the Nimsad Framework *In* Jayaratna, N Fitzgerald, B (eds), *Proceedings of the BCS Fourth Annual Conference on Information Systems Methodologies, Lessons Learned from the Use of Methodologies* Cork British Computer Society 165-176

Berger, P L , Luckman, T , 1973 *The Social Construction of Reality* London Penguin

Bjørner, D , 1998 Domain as a Prerequisite for Requirements and Software Domain Perspectives and Facets, Requirements Aspects and Software Views *In* Broy, M , Rumpe, B (eds), *Proceedings of RTSE'97 Requirements Targeted Software and Systems Engineering LNCS 1526* Heidelberg Springer Verlag 1-41

Blum, B I , 1994 A Taxonomy of Software Development Methods *Communications of the ACM* 37 (11), 8 -94

Blumer, 1969 *Symbolic Interactionism Perspective and Method* Englewood Cliffs, NJ Prentice Hall

Boehm, B , 1991 Software Risk Management Principles and Practice *IEEE Software* 8 (1), 32-41

Brinkkemper, S , Saeki, M , Harmsen, F , 2001 A Method Engineering Language for the Description of Systems Development Methods *In* Dittrich, K R , Geppert, A , Norrie, M C (eds), *Proceedings of the Conference on Advanced Information Systems Engineering, CAiSE 2001, Interlaken, Switzerland* Berlin Springer Verlag 473-476

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

Clements, P C , Parnas, D L , 1985 A Rational Design Process How and Why to Fake It *In* Ehrig, H , Floyd, C , Nivat, M , Thatcher, J W (eds), *TAPSOFT 85 Proceedings of the International Joint Conference on the Theory and Practice of Software Development* Berlin Springer Verlag 80-100

Coase, R H , 1937 The Nature of the Firm *Economica* 4, 386-405

Cronholm, S , Goldkuhl, G , 1994 *Meanings and Motives of Method Customisations in CASE Environments* Linkoping, Sweden Linkoping University

Curtis, B , Krasner, H , Iscoe, N , 1988 A Field Study of the Software Design Process for Large Systems *Communications of the ACM* 31 (11), 1268-1287

Davis, G B , 1982 Strategies for Information Requirements Determination *IBM Systems Journal* 21 (1), 4-30

Eisenhardt, K M , 1989 Building Theories from Case Study Research *Academy of Management Review* 14 (4), 532-550

Episkopou, D M , Wood-Harper, A T , 1986 Towards a Framework to Choose Appropriate IS Approaches *The Computer Journal* 29 (3), 222-228

Fenton, N , Pfleeger, S L , Glass, R L , 1994 Science and Substance A Challenge to Software Engineers *IEEE Software* 11, 88-94

Fitzgerald, B, 1996 Formalised Systems Development Methodologies A Critical Perspective *Information Systems Journal* 6, 3-23

Floyd, C, 1987 Outline of a Paradigm Change in Software Engineering *In* Bjerknes, G, Ehn, P, Kyng, M (eds), *Computers and Democracy A Scandinavian Challenge* Aldershot, UK Dower Publishing Co 185-202

Floyd, C, 1995 Theory and Practice of Software Development - Stages in a Debate *In* Mosses, P D, Nielsen, M, Schwartzbach, M I (eds), *TAPSOFT'95 6th International Conference on the Theory and Practice of Software Development LNCS 915* Aarhus Springer Verlag 25-41

Floyd, C, Reisin, F -M, Schmidt, G, 1989 STEPS to Software Development with Users *In* Ghezzi, C, McDermid, J (eds), *ESEC 89, Proceedings of the European Software Engineering Conference, LNCS 387* Berlin Springer-Verlag 48-64

Floyd, C, Zullighoven, H, Budde, R, Keil-Slawik, R, eds, 1992 *Software Development and Reality Construction* Berlin Springer Verlag

Galliers, R D, 1991 Choosing Appropriate Information Systems Research Approaches *In* Nissen, H, Klein, H K, Hirschheim, R (eds), *Proceedings of the IFIP TC8 WG 8 2 Working Conference on the Information Systems Arena of the 90's* Copenhagen North Holland 155-173

Gasson, S, 1999 A Social Action Model of Situated Information Systems Design *The DATA BASE for Advances in Information Systems* 30 (2), 82-97

Gilb, T, 1988 *Principles of Software Engineering Management* New York Addison Wesley

Gilovich, T, 1991 *How We Know What Isn't So The Fallibility of Human Reason in Everyday Life* New York The Free Press

Glaser, B, Strauss, A, 1967 *The Discovery of Grounded Theory* Chicago Aldine

Glass, R L, Vessey, I, 1995 Contemporary Application Domain Taxonomies *IEEE Software* 12 (4), 63-76

Gotel, O, Finkelstein, A, 1994 An Analysis of the Requirements Traceability Problem *In*, *Proceedings 1st International Conference on Requirements Engineering (ICRE'94)* Colorado Springs IEEE 94-101

Gray, W D, ed 1996 *Empirical Studies of Programmers sixth workshop* Norwood, N J Ablex Publishing

Greenspan, S, Mylopoulos, J, Borgida, A, 1994 On Formal Requirements Modelling Languages RML Revisited *In*, *Proceedings 16th International Conference on Software Engineering* Sorrento, Italy IEEE Computer Society Press 135-148

Grinter, R E, 1999 Systems Architecture Product Designing and Social Engineering *In* Georgakopoulos, D, Prinz, W, Wolf, A (eds), *Proceedings of the International Joint Conference on Work Activities and Collaboration WACC'99* San Francisco, CA ACM Press 11-18

Grudin, J, 1996 The Organisational Contexts of Development and Use *ACM Computing Surveys* 28 (1), 169-171

Gummesson, E, 2000 *Qualitative Methods in Management Research* Thousand Oaks, CA Sage

Hage, P, 1972 Munchner Beer Categories *In* Spradley, J (ed), *Culture and Cognition Rules, Plans and Maps* San Francisco Chandler 263-278

Harel, D, 1987 Statecharts A Visual Formalism for Complex Systems *Science of Computer Programming* 8, 231-274

Hunter, A, Nuseibeh, B, 1998 Managing Inconsistent Specifications Reasoning, Analysis, and Action *ACM Transactions on Software Engineering and Methodology* 7 (4), 335-367

IEEE, 1998 *IEEE STD 830-1998 IEEE Recommended Practice for Software Requirements Specifications* Los Alamitos, CA IEEE Computer Society Press

Jackson, M, 1997 The Meaning of Requirements *Annals of Software Engineering* 3, 5 - 21

Jackson, M A, 1983 *System Development* Hemel Hemstead Prentice Hall

Jackson, M A, 1995 *Software Requirements and Specifications a lexicon of practice, principles and prejudice* New York Addison Wesley

Jackson, M C, Keys, P, 1984 Towards a System of System Methodologies *Journal of Operational Research* 35 (6), 468-473

Jarke, M, Pohl, K, 1994 Requirements engineering in 2001 (on virtually) managing a changing reality *Software Engineering Journal* November, 257-266

Jarke, M, Ramesh, B, 2001 Toward reference models for requirements traceability *IEEE Transactions on Software Engineering* 27 (1), 58 -93

Kaindl, H, 1999 Difficulties in the Transition from OO Analysis to Design *IEEE Software* 16 (5), 94-102

Kelly, G A, 1955 *The Psychology of Personal Constructs* New York W W Norton

Kovitz, B L, 1999 *Practical Software Requirements A Manual of Content and Style* Greenwich, CT Manning

Lamsweerde, A v, Darimont, R, Letier, E, 1998 Managing Conflicts in Goal Driven Requirements Engineering *IEEE Transactions on Software Engineering* 24 (11), 908-926

Lawrence, P R, Lorsch, J W, 1967 *Organization and Environment Managing Differenciation and Integration* Boston Harvard University

Lee, A S, Liebenau, J, DeGross, J I, eds, 1997 *Information Systems and Qualitative Research* Philadelphia Chapman and Hall

Lehman, M M, 1984 Program Evolution *Information Processing Management* 20, 19-36

Lehman, M M, 1989 Uncertainty in Computer Application and its Control through the Engineering of Software *Software Maintenance Research and Practice* 1, 3-27

Lehman, M M, Belady, L, 1985 *Program Evolution Processes of Software Change* London Academic Press

Leveson, N, 1995 *Safeware system safety and computers* Wokingham Addison-Wesley

Lincoln, Y S, Guba, E G, 1985 *Naturalistic Enquiry* Newbury Park, CA Sage

MacCormack, A, 2001 Product-Development Practices that Work How Internet Companies Build Software *MIT Sloan Management Review* Winter

March, J G, Olsen, J P, 1976 *Ambiguity and Choice in Organizations* Bergen Universitetsforlaget

March, J G, Simon, H A, 1958 *Organizations* New York Wiley

Maykut, P, Morehouse, R, 1994 *Beginning Qualitative Research* London The Falmer Press

McDermid, J A , 1994 Requirements Analysis Orthodoxy, Fundamentalism and Heresy *In* Jirotka, M , Goguen, J (eds), *Requirements Analysis - Social and Technical Issues* London Academic Press 17-40

Miles, M B , Huberman, A M , 1994 *Qualitative Data Analysis A Sourcebook of New Methods* Beverly Hills, CA Sage

Muhr, T, 1996 *ATLAS ti - The Knowledge Workbench* [online] Available from http //www atlasti de [Accessed July 1998-Jan 2002]

Mumford, E , 1993 The ETHICS Approach *Communications of the ACM* 36 (4), 82

Mylopoulos, J , 1998 Information Modelling in the Time of the Revolution *Information Systems* 23 (3/4), 127-155

Naumann, J D , Davis, G B , McKeen, J D , 1980 Determining Information Requirements A Contingency Method for Selection of a Requirements Assurance Strategy *Journal of System and Software Sciences* 1, 273-281

Naur, P , 1992 *Computing A Human Activity* New York ACM Press

Newell, A , Simon, H A , 1972 *Human Problem Solving* Englewood Cliffs, NJ Prentice Hall

Nilikant, V, Rao, H, 1994 Agency Theory and Uncertainty in Organisations An evaluation *Organization Studies* 15 (5), 649 - 673

Olsen, M E , 1968 *The Process of Social Organization* New York Holt, Rinehart and Winston

Olson, G M , ed 1987 *Empirical Studies of Programmers second workshop* Norwood, N J Ablex Publishing

Orlickowski, W J , 1993 CASE Tools as Organisational Change Investigating Incremental and Radical Changes in Systems Development *MIS Quarterly*, 309-340

Parnas, D L , 1995 Using Mathematical Models in the Inspection of Critical Software *In* Hinchey, M G , Bowen, J P (eds), *Applications of Formal Methods* Hemel Hempstead Prentice Hall 17-31

Parnas, D L , Clements, P , 1986 A Rational Design Process How and Why to Fake It *IEEE Transactions on Software Engineering* SE-12 (2), 251-257

Perry, D E , Porter, A A , Votta, L G , 2000 Empirical Studies of Software Engineering A Roadmap *In* Finkelstein, A (ed), *The Future of Software Engineering* Limerick IEEE 345-356

Pohl, K , 1994 The Three Dimensions of Requirements Engineering A framework and its application *Information Systems* 19 (3), 243-258

Pohl, K , 1996 *Process-centred Requirements Engineering* Somerset, UK Wiley

Polanyi, M , 1966 *The Tacit Dimension* London Routledge and Kegan Paul

Potts, C , 1993 Software Engineering Research Revisited *IEEE Software* 10 (4), 19-28

Potts, C , 1997 Requirements Models in Context Keynote Address *In* Heitmeyer, C (ed), *Proceedings International Symposium on Requirements Engineering RE'97* Annapolis MD IEEE Computer Society Press 102-104

Ralyté, J , Roland, C , 2001 An Assembly Process Model for Method Engineering *In* Dittrich, K R , Geppert, A , Norrie, M C (eds), *Proceedings of the Conference on Advanced Information Systems Engineering, CAiSE 2001, Interlaken, Switzerland* Berlin Springer Verlag 267-283

Robertson, S , Robertson, J , 1999 *Mastering the Requirements Process* Harlow, UK Addison Wesley

Robinson, W N , Pawlowski, S D , 1999 Managing Requirements Inconsistency with Development Goal Monitors *IEEE Transactions on Software Engineering* 25 (6), 816-835

Roseman, M , Green, P , 2000 Integrating Multiperspective Views into Ontological Analysis *In* Orlickowski, W , Weill, P , Ang, S , Krcmar, H (eds), *Proceedings of the 21st International Conference on Information Systems* Brisbane ACM Press 618-627

Ryle, G , 1949 *The Concept of Mind* Harmondsworth Penguin

Scheer, A -W , 1998 *Business Process Engineering Reference Models for Industrial Enterprises* Berlin Springer Verlag

Schon, D , 1983 *The Reflective Practitioner How Professionals Think in Action* New York Basic Books

Scott, W R , 1987 *Organizations Rational, Natural and Open Systems* Englewood Cliffs, NJ Prentice Hall

Seaman, C B , 1999 Qualitative Methods in Empirical Studies of Software Engineering *IEEE Transactions on Software Engineering* 25 (4), 557-572

Seaman, C B , Basili, V R , 1997 An Empirical Study of Communication in Code Inspections *In*, *Proceedings of the International Conference on Software Engineering, ICSE97* Boston, MA ACM 96-106

Seaman, C B , Basili, V R , 1998 Communication and Organisation An Empirical Study of Discussion in Inspection Meetings *IEEE Transactions on Software Engineering* 24 (6), 559-572

Sharma, A , 1997 Professional as Agent Knowledge Asymmetry in Agency Exchange *Academy of Management Review* 22 (3), 758-798

Shaw, M , Clements, P , 1996 Towards Boxology Preliminary Classification of Architectural Styles *In*, *Proceedings of SIGSOFT 96 Workshop* San Francisco, CA ACM 50-54

Shaw, M , Garlan, D , 1996 *Software Architecture Perspectives on an Emerging Discipline* Englewood Cliffs, NJ Prentice Hall

Shipman, F M , Marshall, C C , 1999 Formality Considered Harmful Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems *Computer Supported Cooperative Work (CSCW)* 8 (4), 333-352

Sommerville, I , Sawyer, P , Viller, S , 1999 Managing Process Inconsistency Using Viewpoints *IEEE Transactions on Software Engineering* 25 (6), 784-799

Spradley, J , 1972a Adaptive Strategies of Urban Nomads *In* Spradley, J (ed), *Culture and Cognition Rules, Plans and Maps* San Francisco Chandler

Spradley, J , 1979 *The Ethnographic Interview* New York Holt, Rinehart, and Winston

Spradley, J , 1980 *Participant Observation* New York Holt, Rinehart and Winston

Spradley, J , ed 1972b *Culture and Cognition Rules, Plans and Maps* San Francisco Chandler

Stamper, R , 1994 Social Norms in Requirements Analysis - an Outline of MEASUR *In* Jirotka, M , Goguen, J (eds), *Requirements Analysis - Social and Technical Issues* London Academic Press 107-140

Star, S L , 1997 *Grounded Classification Grounded Theory and Faceted Classification* [online] Available from http //alexia lis uiuc edu/~star/gt htm [Accessed 24 July 1997]

Strauss, A , Corbin, J , 1990 *Basics of Qualitative Research Grounded Theory Procedures and Techniques* Beverly Hills, CA Sage Publications

Suchman, L , 1987 *Plans and Situated Action* Cambridge Cambridge University Press

Suchman, L , 1995 Representations of Work, Introduction *Communications of the ACM* 38 (9), 33 - 35

Swartout, W , Balzer, R , 1982 On the Inevitable Intertwining of Specification and Implementation *Communications of the ACM* 25 (7), 438 - 440

The Standish Group, 1995 *The Standish Group Report - CHAOS* [online] Available from http //www scs carlton ca/~beau/PM/Standish-report html [Accessed July 2000]

Vimarlund, V , 1998 Participatory Design in Economic Terms A Theoretical Discussion *In* Chatfield, R , Kuhn, S , Muller, M (eds), *PDC 98 Proceedings of the Participatory Design Conference* Seattle, WA CSPR 11-17

Vitalari, N P , Dickson, G W , 1983 Problem Solving for Effective Systems Analysis An Experimental Exploration *Communications of the ACM* 26 (11), 948-956

Wegner, P , 1997 Why Interaction is more Powerful than Algorithms *Communications of the ACM* 40 (5)

Weick, K , 1979 *The Social Psychology of Organising* New York Addison Wesley

Weick, K , 1984a *Sensemaking in Organizations* Thousand Oaks, CA Sage Publications

Weick, K , 1984b Theoretical Assumptions and Research Methodology Selection *In* McFarlane, F W (ed), *The Information Systems Research Challenge Proc of the Harvard Business School Research Colloquium* Boston Harvard Business School Press

Wernick, P , 1996 *A Belief System Model for Software Development A Framework by Analogy* Thesis London University College London

Wieringa, R J , 1996 *Requirements Engineering Frameworks for Understanding* Chichester, UK Wiley

Wieringa, R J , Jansen, D N , 2001 Techniques for Reactive System Design The Tools in TRADE *In* Dittrich, K R , Geppert, A , Norrie, M C (eds), *Proceedings of the Conference on Advanced Information Systems Engineering, CAiSE 2001, Interlaken, Switzerland* Berlin Springer Verlag 93 -107

Williamson, O E , 1975 *Markets and Hierarchies Analysis and Antitrust Implications* New York Free Press

Zave, P , 1993 Feature Interactions and Formal Specifications in Telecommunications *IEEE Computer* 26 (8), 20-30

List of Appendices

Appendix A

Interview Summary Table

Interview Summaries

ID	Relationship situation (see Chap 1)	Main Root Concerns (in Order)	Application Domain *Problem Frame*	Characteristic Document	Organisation of Process
01 IR	In-house	Client organisation Users Project	Payroll data collection MRP *Transformation*	External and Internal Specifications Textual	Methodology-driven, but some a la carte Signoff
02 HD	In-house working as a consultant with company personnel	Business Area Client Organisation Project	Financial Services Global System Review *Information and Control*	Requirements as text enhanced with Entity-Relationship and DFD models	SSADM used a la carte, Bus Proc engineering done but not called that
03 FOC	External Contract (3 parties)	Customer Project Product Application domain	Telecoms Database *Information and Control*	Functional Requirements Textual	Ad hoc, re-developing an existing system for an external client
04 BH	In-house but having external users	Application domain Business area Users Project	Government revenue collection *Information and Control*	Functional requirements with DFDs, Logical data model	Ad hoc, teamwork with multiple roles per person
05 GR	Cooperative development with targeted first customer	Product Project Customer Application domain	Utilities Maintenance planning and control package *Information and Control*	Numbered Classified Functional Requirements	Based on ESA model Difficulties in getting commitment from customer user groups
06 GP	In-house, informal	Users Client Organisation Business Area	Manufacturing Reports Management Information *Transformation,* *Information*	Minimal textual, Often combined with design	No formal project, though some signoff procedures

ID	Relationship with customers or clients	Main Root Concerns (in Order)	Application Domain *Problem Frame*	Characteristic Document	Organisation of Process
07 POR	In-house working with external consultants and head office persons	Business Area Client Organisation Project	Project-based ERP Aircraft Maintenance *Information and Control*	3 types Task definition (block diagrams) Issues Integration with ERP product	Business Process Engineering, Agreement not in the Document
08 MK 09 JD	External contract Joint project teams (BB Case Study)	Customer Product Application Domain Project	Supplying customised warehouse management software *Information and Control*	Textual document Contract Breakdown by BP Operation, Req, Proposal	Solution provider Fitting the solution to the problem, and vice versa
10MS 11SH 12CL 13CM 14JN	Outsourced Joint project teams (BB Case Study)	Project Business Area	Acquiring customised warehouse management Software *Information and Control*	Textual document Contract Breakdown by BP Operation, Req, Proposal	Focus of discussions with solution provider (Blackbird Clients)
15 HM	Solution development for external customers	Market Opportunity Customer Application domain Project	Telecoms Network Management custom software, with extra targeted customers *Information and Control*	Business Requirements Market reqmts	Negotiating with marketing dept to streamline requirements CMM
16 AA	Solution development for external customers	Application domain Project Customer Product	Telecoms Network Management and embedded custom software *Control* *Information*	Functional requirements and design	Well-defined process CMM
17 GC	Product evolution	Application domain Product Customer	Telecoms Embedded Switching Software *Control* *Information*	Functional and non-functional requirements	Well-defined process

ID	Relationship with customers or clients	Main Root Concerns (in Order)	Application Domain *Problem Frame*	Characteristic Document	Organisation of Process
18 FD	In-house	Business Area Client Organisation Users	Telecoms Special Offers and Billing database system small operator *Information Transformation*	Business case Informal functional	Negotiation and agreement
19 FML	In-house for parent company	Application Domain Product	Telecoms Intelligent Networks *Information and Control*	Solution Investigation Functional and Non-functional	Knowledge acquisition Feasibility Study Understanding
20 DT	Product development for external client	Application Domain Product	Embedded sw for consumer electronics *Control*	Table based Complete model of States and Transitions in Functional Spec	Aimed towards Completeness, Understanding of constraints and Correctness
21 DD	Outsourcing development to bespoke or other solutions providers	Application Domain Client Organisation Users	Acquiring health administration systems *Information and Control*	Request for proposals	Well-defined process
22 YY	Outsourcing development to bespoke or other solutions providers	Application Domain Client Organisation Users Business Area	Acquiring health administration systems *Information and Control*	Request for Proposals	Well-defined process for largeprojects Ad hoc for smaller projects
23 NG	Configuring and customising an ERP product for external clients	Client Organisation Business Area Application Domain Project	ERP installation esp Manufacturing *Information and Control*	Feature-oriented	Consultancy

ID	Relationship with customers or clients	Main Root Concerns (in Order)	Application Domain *Problem Frame*	Characteristic Document	Organisation of Process
24 DOC	Internal product development and evolution for market	Market Opportunity Product Project Application Domain	Embedded software for a family of KVM Switch products *Control*	Feature-oriented	Multi-site team
25 DW	Internal product development	Market Opportunity Product Customers Project	Telecoms Switch *Control*	Feature description document derived from a DOORS database	Well-defined process CMM
26 PS	Bespoke development for external clients	Client Organisation Project Business Area Application Domain	Databases *Information and Control*	Derived from functional reqs stored in a spreadsheet Textual	Risk reducing strategy Company process guidance/support - reference materials
27 LF	Internal product development and evolution for ERP market	Market opportunity Product Project Application Domain	ERP Product *Information and Control*	Lotus database of documents Spreadsheet of prioritised features supported by use cases	Product steering committee, Negotiating with marketing, input from customers, focus groups, etc
28 DC	Application expert Configuring, enhancing and extending the package	Application domain Users Project	Acquisition and tailoring of an application package for student records *Information*	RFP plus various other documents written by client, Vendors std materials	Project Steering committee, Board and joint project team working with Vendors

Appendix B

Letter to Interviewee

Dear Mr Pinckheard,

Thank you for taking my call on Friday

As I mentioned on the phone, I am doing research on the nature of 'requirements documents' and their role in the practice of system development At present I am interviewing experienced practitioners on their experiences and views, in an effort to bridge the gap between theory and practice I would be very pleased and grateful if you agree to be part of this study

The interview will take about an hour, at a time and place convenient for you I will be asking you to talk about your experience of writing and using requirements documents, and also seeking your opinions on their usefulness in system development

In previous interviews I have found it useful to focus the discussion on one particular project which you want to talk about This should be one which is considered to be successful, as I am interested in best practice rather than war stories It is also helpful, but not necessary, to have a copy of the requirements document for the particular project handy, for reference

I know how busy you must be, but I hope that you will be able to take part If you have any queries, or require further clarification, I will be in touch in a few days, when I hope we will be able to arrange a suitable time for meeting

Yours sincerely,

Appendix C

Interview Guide

Interview Guide – Version 2

1. Background

Would you like to begin by telling me a little about yourself and your role in the project/system that you are going to discuss?

2. The Project Itself

Can you give a brief description of the project? How many people are/were involved? How was it organised? Any user involvement? Roles

3. Methods and Tools

Can you tell me about the method used in the project Was it formally defined? Process? Guidelines or standards used?

4. The Documents

I'd like you to tell me about the different documents that were produced in the course of the project (Note Title(s))

5. Focus on Requirements Document(s)

Structure?

Use of models, diagrams, tables, etc

What's in it? Why? How much detail is needed?

Standard format? If so, how defined? Template, etc

How do you know when it's finished?

6. Uses of the Requirements Document

Was the *requirements document* used later on in the project If so how?

e g testing

Normally changes to requirements arise during the project Were any such changes incorporated in the *requirements document* ?

Did any errors in the requirements come to light at a later stage? How were these dealt with?

7. Whose Document is it?

People who use the document Different kinds of stakeholders

8. Agreement

How does a document such as this help to achieve agreement about what is required?

Is it a contract? To what extent?

9. Quality

What **in your view** makes a **good** *requirements document* ?

Does it exist in practice? If not, why not?

Interview Guide – Version 1

This was used as a guide to start the interviews, but in nearly all cases the respondent guided the course of the rest of the interview shortly after the initial questions were addressed

1 Respondent
Would you like to begin by telling me a little about yourself and your role in the project/system that you are going to discuss?
2. The Project Itself
Can you give a brief description of the project? How many people are/were involved? How was it organised? Any user involvement? Roles
3. Methods and Tools
Can you tell me about the method used in the project Was it formally defined? Any formal training given? Guidelines or standards used?
4. The Documents
I'd like you to tell me about the different documents that were produced in the course of the project (Note Title(s))
5. Focus on Requirements Documents
Structure?
How much detail?
Standard format? If so, how defined?
6. Use of Requirements
Was the *requirements document* used later on in the project If so how?
e g testing
7. Changes
Normally changes to requirements arise during the project Were any such changes incorporated in the *requirements document* ?
8. Errors
Did any errors in the requirements come to light at a later stage? How were these dealt with?
9. Barriers to Communication
What do you think is the best way to find out the users' requirements and document them?
10. Problems
What in your view were the problems inherent in using the *requirements document* ?
11. Quality
What do you think makes a **good** *requirements document* ?
12. Practicality
Do you think it is possible to produce a *requirements document* that lives up to these criteria?

Appendix D

List of Propositions

These were abstracted from the interview transcript data by means of 'constant comparative analysis ' Each propositional statement is supported by a number of quotations referring to the original page of the transcript

P1 Systems are not usually developed in isolation There are often other systems which need to interface with the system being developed
e g data collection system (for a payroll system) IR1
e g the personnel enquiry system already contained some of the data IR1
e g statistics reporting system from an existing manufacturing package GP1
e g system interfacing to the General Ledger HD2

P2 Systems are not always developed from scratch There may an existing system which is being re-developed and up-dated, or a generic product or package may be developed which is based on some existing system
"re-engineering the core requirements" GR1
e g generalised maintenance management system for power stations GR1
e g generalised network administration system for PTTs FOC1
e g configuration project using SAP enterprise system

P3 System development does not always take place in the context of a "project "
Many systems evolve or grow over time, through a *process* of satisfying "requests " Small systems or additional programs are also developed in this mode
compare this with maintenance
e g GP8
e g GP13 first-in first out
e g POR13

P4 Requirements are not always written down Some are prototyped Many systems do not have any type of requirements document
"We would have sat down and tried to understand the requirements
written list of questions
clarify the requirements
he has explained to people what the system will do " GP8
e g prototyping GP7
e g discussions with the users BH4, POR4
"There is no screen design in here, which personally I think there shold be " HD4

P5 When is a project not a project?

P6 Sometimes a new system is required to formalise existing procedures
"the way we do things now" GR8
"already doing a lot of it with spreadsheets" GP2
But "We were trying to analyse business processes that didn't yet exist " POR3

P7 Many people are involved in a project They come from different areas of the organisation, perhaps from different organisations
"I got contracted in " FOC1
e g two organisations GR1
e g client organisation plus contractor and sub-contractor organisations FOC1

e g outside contractor used for larger systems GP8
e g client organisation acted as the first customer for (GR2)
e g level of involvement of client organisation was a problem for (GR8)
"Both our parent companies selected consultants " POR3
external consultant was project manager for (POR2)
"when all these people went away " POR4

P8 deals with different job titles

P9 System Development Methods vary in
their maturity GR3
their prescriptiveness GR3, IR10, IR3
the extent to which they are imposed on an organisation IR7, GP5
the extent to which they are modified by their users IR2, IR3, GR3, HD3
I will discuss the literature on this and refer to the resulting lack of standardisation (?) in documentation

P10 Users may 'sign off' a requirements document though they do not understand it
" but you can't rely on it as the final say, so it wasn't a contract " HD4
"People are not good at writing and they are not good at reading " POR8
e g IR8
e g GP9

P11 Key users may be managers or they may be user representatives, assigned to the project, either full-time or part-time
e g HD3
e g GP2
e g GP11
e g GR8/9

P12 Users may not appreciate the importance of their role in the project They may not understand what is required of them
"It is not just a question of being able to write down what you got correctly, because sometimes you get very superficial information and sometimes you get irrelevant information " HD5
"People always have their hobby horses, his was the capabilities list " POR16
"The user doesn't listen to you a lot of the time, and you are living the system he is in and out of the system " BH14
e g GR 8
e g GP7
e g GP9
e g GP11

P13 The reasons why requirements should be written down include the following
- method of communication with developers GP (GP7??)
- proof of understanding GP7, FOC4
- users know what they are getting (provided it is clear) IR8

P14 Written requirements may be used later on for
- high level design GR5
- system test GR5, HD4

But "They worked mainly from the design documents " HD4
- system design IR9
- user manual IR9

P15 But the (user) requirements document is not the only source for those subsequent stages
also use interview notes for design IR12
for test, also use design document, especially for lower level conditions GR5

P16 "Errors" in requirements range from vague requirements that need refining to ones that need to be dropped From serious ones that need re-negotiation to ones that can be changed without calling a meeting
e g GR6
e g FOC8
e g IR10
e g GP3

P17 Problems in requirements, when discovered in time, become changes
Depending the seriousness, changes may be handled by
- a proposal
- a review meeting
- a series of negotiations

e g GP6
e g GR6
e g FOC7

change request process GR5
does it make a difference whether the requirements have been signed off or not?

P18 The requirements document tends to be frozen in time and not updated with changes which instead are
- kept in a folder
- pencilled in
- added in only when the project is completed FOC7, GP6
- documented in the minutes of meetings

P19 Requirements are gathered by
- interviewing
- meetings
- looking at existing documents
- looking at existing systems (e g spreadsheets, GP)

i e interacting with people interspersed with periods of "back to the desk" (IR4)
e g IR3, IR4
e g FOC9

P20 Requirements gathering is an iterative process involving review meetings and negotiation
"it was a huge negotiation exercise" BH2
"The actual agreement was ironed out during various committee meetings " HD3
"ironing out wrinkles" FOC5
e g GR5

P21 Requirements gathering needs a facilitator who will balance the needs and demands of all the interested parties
e g FOC
e g POR12 "hobby horse"
counter- examples of this GR7, GR8

P22 All requirements are not necessarily written down

There may not be a requirements document GP4
Some aspects of the system, such as the user interface, may be prototyped instead GR5
The requirements may be written in some composite document, such as the project plan FOC2

P23 Different specification documents are produced in the course of a project The names used are not standard

- requirements definition IR6
- external specification IR6
- internal specification IR6
- abbreviated specification for short/small projects IR5
- "more detailed documents" FOC3
- project plan FOC6
- high-level design FOC6
- release notes FOC4
- list of requirements GR3
- Type 1, Type 2 and Type 3 documents POR

P24 Is there a standard layout which a company uses for requirements documents?
each one is written from scratch GP3
draft TOC based on some standard FOC11

P25 As well as containing details, the requirements document should contain an overall picture of the required system

evidence for this? see P26

P26 The big picture is an essential part of the requirements document
GR9, GR10
wood and trees GP4
hierarchical structure FOC10

P27 Requirements may be broken down into logical groupings
"Each of these (sections) relates to a Data Flow Diagram that's in the Appendix" HD3
logical groups FOC5
cells GR9/GR10/Gr11

P28 Sometimes breaking down requirements into sections makes it more difficult to understand
(Lack of the big picture?)
IR9/IR10

P29 Requirements can be written in plain English
FOC2
GR4

P30 Requirements written in plain English may need to be clarified
" some of the best (systems analysts) are incapable of writing a document that conveys what they mean " POR8
GR4
FOC8

P31 The requirements document has a wide readership
circulation list GP6
FOC9
mixed audience GP7

But "very few people will actually read it " BH

P32 Individual requirements may be numbered
HD2
FOC5
GR4
counter-example GP9
may use decimal numbering FOC5
or not GR4

P33 Individual requirements may be categorised
- by priority
- by business need
- by stability
- by status
- by source

GR4

P34 A requirements document is finished when
- everything is defined to the same level of detail FOC13
- everything is explicit FOC13
- it has no requirements marked TBD GR4
- all the users questions have been answered IR9
- "when we've reached what I regard as an acceptable level of detail" GR4
- when it's signed off by the users IR9 GR4
- when we decide on all the tentative requirements GR4

We rely on the client and our own good judgement GR9

P35 The overall picture
someone who GR10
" the role of the systems analyst is important in creating that shared vision " POR7

P36 Users don't need lots of detail in their requirements document
They need to know what they are getting IR11
"User management tend to think in terms of mechanisms and how things are going to work This document doesn't describe how things are going to work, it says, what do you need " HD3
But "The more specific you have to write about something the more likely it is to be understandable The higher up you go the more nonsense it becomes " POR18

Appendıx E

Format of the Blackbırd Requırements Specıficatıon

Thıs document uses a conventıonal natural language approach to specıfyıng and agreeıng customer requırements The area of Blackbırd's busıness that I studıed (ıt also supplıes specıalısed hardware) ıs the customısatıon of a proprıetary warehouse applıcatıon software system for ıts customers

All of Blackbırd's customers are sımılar ın that they requıre a warehouse applıcatıon, but theır needs are all suffıcıently dıfferent that a standard solutıon wıll not work for them, hence they need a taılored solutıon Blackbırd has a glossy brochure and vıdeo descrıbıng theır 'product' Harvest, but few of theır customers buy ıt "off-the-shelf " A requırements document, called a Specıficatıon, ıs produced from scratch for each new customer

Each requırements document follows the same format It begıns wıth (or ıs preceded by) a document hıstory, showıng the sequence of drafts and revısıons, gıvıng the date on whıch each versıon was approved and the ınıtıals of the revıewers on each sıde

Each new Versıon of the specıficatıon ıs dıscussed at a formal meetıng, wıth ıntermedıate drafts beıng produced and forwarded to the customer between formal meetıngs Once agreement ıs reached, the customer sends Blackbırd a purchase order for the system
Further versıons of the document may follow after thıs, but only mınor modıficatıons and mostly clarıficatıons wıll tend to be made subsequent to thıs

In practıce, there ıs no such thıng as a 'final document' as after a certaın poınt ın the project, Blackbırd uses CSRs (Customer Servıce Requests) whıch customers send ın requestıng changes and add-ons to what ıs documented ın the specıficatıon, but whıch are not reflected ın a new versıon of the document Each of the changes and add-ons must be negotıated, and sometımes requıre an extra Purchase Order to cover the cost of ımplementatıon

The Introductıon ıs a faırly short sectıon of the specıficatıon It brıefly descrıbes the clıent company and theır area of busıness, and summarıses the basıc functıonalıty of the system, notıng that Blackbırd software wıll be used for the proposed solutıon Thıs ıs followed by a termınology sectıon, essentıally a table of terms and theır meanıngs The terms ınclude the names of names of the partıes to the agreement, thırd partıes who supply other software or servıces, as well as technıcal terms belongıng to the applıcatıon

The main part of the document, called Operation, follows next This consists of several (at least 12 or so) sections, each representing a named business operation, for example, "Put-Away," and each divided into the following sub-sections

- A subsection titled "The Operation" which is a description of the operation itself, usually some aspect of warehouse operation such as picking, receiving, putting away, etc
- A subsection titled "The Requirement" which describes the requirement(s) associated with this operation, such as
  "To record the movement of the pallet with its associated serials to a storage location in the racks, and to inform BMS of this movement "
- A subsection titled "Proposed Solution" written in terms of the Blackbird functionality which will be provided A list of functions is given, each one beginning with a verb such as scan, confirm, ensure, warn, display, prompt, etc
- Other sub-sections which deal with Interfaces to other systems, Hardware required (e g Radio Frequency terminals are commonly used), Communications, Transaction volumes, and Open issues The latter section records problems and questions which have yet to be discussed, and to be resolved in subsequent versions of the document, the resolutions being documented as Answers in the same subsection

The remainder of the document contains sections on Security, Site Preparation, Implementation, and Costs

Appendix F

List of Codes

a la carte use of method
acceptance testing
accepted
accumulated changes
accuracy
Acronyms
Ad Hoc Reporting
afterthefact
agreed requirements
agreement
agreement not in document
alpha test
an adequate design
analysis
answers all their questions
application domain
application expertise
application note
application problems
applying knowledge
architecture
architecture first
as a discussion document
as a method of communication
as a working document
as the basis of the architecture
ask the product manager
aspects
assumed knowledge
assumed requirements
Assumptions
authority
Authorization List
authorship
back to the desk
background
basic building blocks
basic set of needs
basis for prototyping
bedoingsomething
behaviour
behaviour model
Behavioural Constraints
being precise
beta test
better ways of doing them
big picture
big project
Blackbird document
boundary
boundary of automation
brainstorming
breakdown
bridging the gap
broad vision
bugs
bundles
business area
Business Case
business case first
business knowledge
business modelling
business need
business opportunity
business process
Business Processes
business questions
business requirements
buy and apply
can't rely on document
capture
cattle drive
change of mind
change request process
change requests
changeability
Changes
changes - need for
changes - notes on
changes - pencilled in
changes to existing system
choice of format
chosen package
Circulation List
clarification
clarifying requirements
clarity
client
client organisation
Closed Issues
coal-face
coherent operations
commit
common requirements
common understanding
communication
communication gap
Company Background
company ethos
competition in the market
complementary manual system
completeness
completion
complexity
component-based software
computer auditor
concise
configurable package
configurable product
configuration

configured solution
conflicting requirements
conformance with method
connectionless communication
consistent
Constraints
constraints section
consultancy
consumer electronics
contacts
contents
contract
controller
core requirements
Corner Cases
cosmetic
cost/benefit of business proposal
customer
customer document
customer expertise
customer organisation
customer requirements
customer's terminology
customised solution
data
data - importance
Data Collection
data description
Data Flow
data models
Database
Database Contents
database design central
Database Structure
deadlines
deal-breaker
dealing with inconsistency
dealing with issues
dealing with suppliers
debate
decisions
decomposition
Delete
depends on application
describe the behaviour
describing the business
design
Design Constraints
design decisions
design rules
design vs requirements
develop the system, not the document
different assumptions
different audience
different documents
different goals
different hats
different needs
different notations
different perspectives
different set of people
different situation
different tasks
Different Types Of Content
difficult to locate
disagreement
discussion document
discussions
distributed functionality
dividing into subsystems
dividing requirements
division of work
Document History
Document Information
document reflects changes
document status
document structure
documentation
documentation forms
documentation method
documentation requirements
domain knowledge
don't lose sight of objectives
draft table of contents
drafts
drawing pictures
easy to follow
easy to read
effort required
email contact
embedded software
embedded systems
engineer
engineering discipline
ERP
Error Cases
errors in requirements
Estimates
estimating what's involved
event-based analysis
event-based behaviour
Events
eventualities
existing system
expertise
explain
explain the structure
explicit
exploratory development
express understanding
extending the software
external consultant
external description
External Events
external specification
facilitator
factors we have control over
fault report
faxes and communications
Feasibility Analysis
feasibility of changes
feasibility of requirement

features
filtering information
finding out
fitting the model to the situation
fixed in time
fixing bugs
flexible
flexible design
flow
flowcharts
focus of discussions
formal
formal methods
format chosen by customer
FRS
functional areas
functional decomposition
functional description
functional requirements
functional specification
functions
gathering information
geared to the user
generating documentation
generic requirements
get the concepts across
getting it right
Global Requirements Specification
Global Requirements Study
Glossary
go-ahead
goal
Goals
good enough document
good relationship
greenfield situation
groups of requirements
guard against surprises
guidelines
hard requirements
hard to read
headings
here's our understanding
hierarchical breakdown
hierarchy
high throughput
Historical Rates
hobby horse
homework
how do you specify quality?
how to begin
impact of business proposal
impetus for new system
implications
imposed method
inconsistencies
independent testing
independent validation of design
influence of large customers
informal approach
information requirements
input to costing
input to design
input to project planning
input to testing
input to user manual
Insert
integration
integration manager
integration testing
intelligent network
interaction with users
interface
interfaces to other systems
Interfaces To Other Systems
internal clients
internal specification
interpretation of requirements
interview material
interview record
interviews
invisible requirements
IS strategy
Issues
IT organisation
iterative development
iterative process
ITT
joint projects
keeping track
key users
knowhow
knowledge
knowledge acquisition
knowledge of the business
knowledge of the package
knowledgeable customers
lack of information
lack of specification
language of document
language problems
large document
layered software
level of detail
levels of agreement
like a user manual
List of Reference Documents
list of requirements
long document
looking for solutions
maintenance control
making a prototype
many users
mapping onto the software
market changes
market opportunity
market requirements
market variants
marketing people
marketing support
match current capability
matrix format

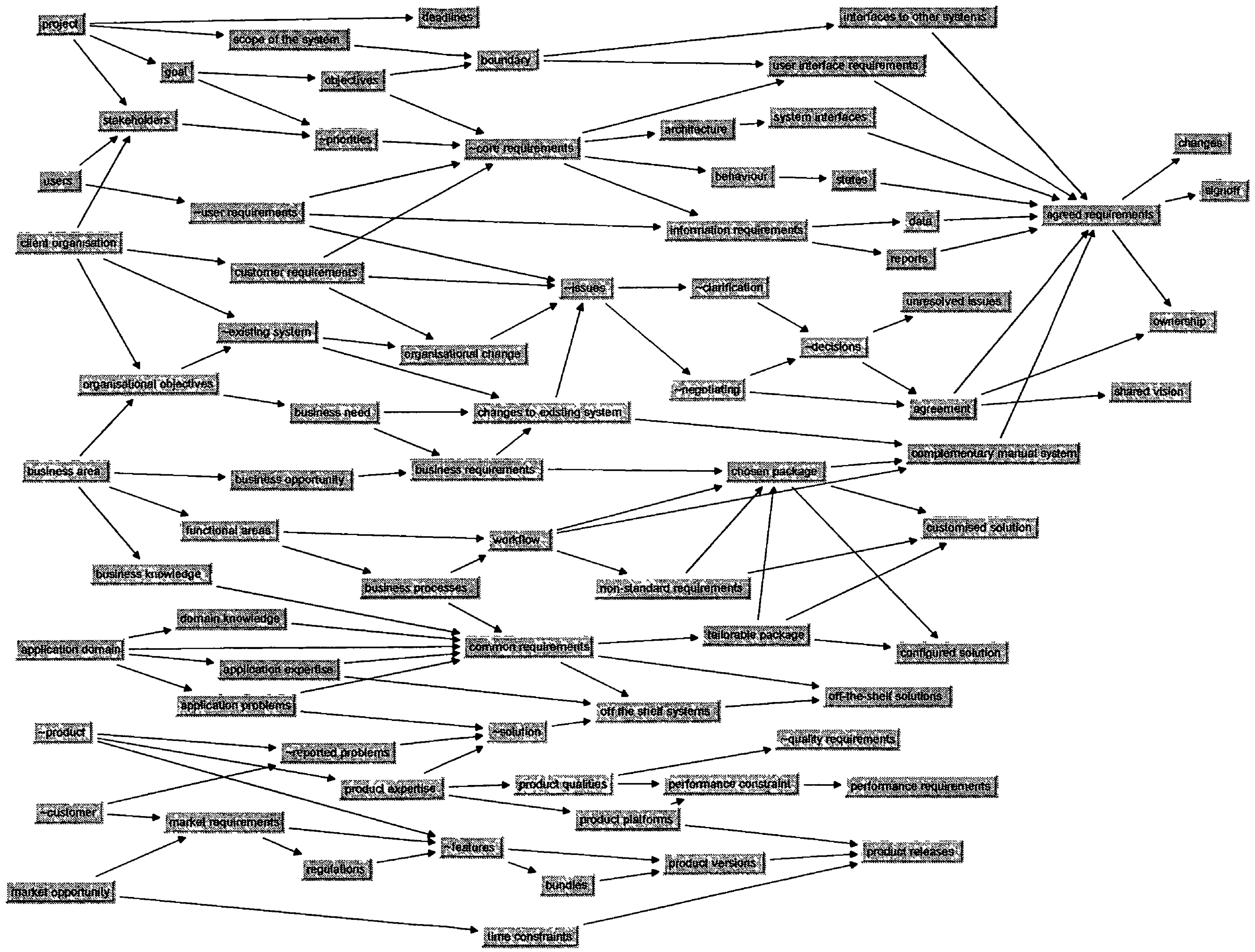
measurable
measuring non-functional requirements
meetings
meetings - face to face
memos
methods
minutes of meetings
mistakes
model
modified system
Modify
modify existing specification
multiple organisations
names of documents
need for a formal process
need for measurement
need to know
negotiating
network administration
new things
next condition
next state
nice features
Non-Functional Requirements
non-functional requirements related to functional
non-standard requirements
notation to suit system
numbered
Objectives
off the shelf systems
off-the-shelf components
off-the-shelf solutions
omissions
one person in charge
onus is on the systems analyst
Open Issues
operation
operation manual
Operations
operations - exceptions
order takers
organisational change
organisational context
organisational objectives
organising requirements
organising the specification
outside delays
outside forces
outside the scope
overlapping
overview
overview design
ownership
package parameters
package selection
part of their tool-kit
partitioning
part-time task
people involved in projects
perfective maintenance
performance
performance constraint
performance requirements
persuade
phases
physical constraints
physically separated from the client
plain English
points
precise
pressure of work
priorities
proactive IS dept
Problem Domain Descriptions
problem frame
problems
procedures
process
producing documents
product
product expertise
Product Manager
product manager's document
product platforms
product qualities
product releases
product support
product versions
productise existing system
project
project approach
project leader
project management requirements
project organisation
project plan includes requirements
project size
project team
proof
proposal
Proposed Solutions
protection
protocol solutions
Prototypes
prototyping
providing a service
push back the boundaries
QA
QA person
quality
quality of service
quality requirements
queries re requirements
query processing
Query Processing
questionnaire sheets
reaction to change
read the specification
readership
reading the document
real needs
reasoning

reasons for being involved
Reasons For Changes
receiver of the requirements
reference section
references
refining the requirements
regulations
reported problems
Reports
representation
Request For Proposals
request(s)
requirements as a deliverable
requirements definition (document)
requirements fit design
requirements that need special attention
resolving issues
responsibility
responsive
re-use
reviews
revisions
risk
risk analysis
role of the systems analyst
roles in project organisation
room for manoeuvre
Rules
safe document
SAP implementation
satisfied
scheduler
scope for extending functionality
scope of the system
sections
selling a solution
shared understanding
shared vision
short document
signoff
Signoff Dates
skill of writing
skills
small projects
software architect
software development manager
software engineer
software requirements
solution
Solution Investigation Document
source document
sources of difficulty
specialist teams
specific
specification
stability (of req)
stages of testing
stakeholders
Standard Features
standard package
standard template
standards
standards - documents
standards -telecomms
starting from scratch
state diagrams
state variable
States
steering groups
strategic plan
strategic project
stress testing
structure reflects business functions
structured methods
success factors
supplier
support analyst
supporting procedure
System Architecture
system interfaces
system requirements
system testing
systematic
systematise and classify the products
systems analyst
table
Table Of Contents
tailorable package
tailoring standards
talking with users
task definition
Tasks
team of consultants
team work
technical people
technical problem
technically difficult
technology push
telecomms market
telecomms switches
telecomms versus software
tells users what they are getting
Terminology
terms of reference
test cases
test harness
test specification
tester
testing
text/diagrams
the contract is based on this document
the source of the requirement
the way we do things now
the way we work
There may not be a requirements document
things came to light
think they need
third party
this document supersedes his document
time available
time available for analysis
time constraints

time pressure
timespan mismatch
time-to-market
to evaluate proposals
to get agreement
ToBeDecided
too elaborate
too technical
tools for analysis
top level
traceability
traceability matrix
traffic handling
training using prototype
transaction processing
transformation
Transformation Requirements
Transitions
trust
trying to tighten it down
Types Of Behavioural Requirements
Types Of Effects
Types Of Features
Types Of Information Requirements
types of table
Types Of Transactions
uncertainty about knowledgeability of customer
unclear
uncovering errors later
understanding
unfilled requirements
unforeseeable requirements
Unique Features
unresolved issues
unwanted requirements
Update
updates
URS
usability
use of diagrams
use of language is missed
use what fits the situation
use what's useful
usefulness of the tabular approach
User Actions
user interface design
user interface requirements
user involvement
user representatives
user requirements
User Requirements Specification
user testing
user thinks differently
user-level requirements
user-perceived states
users
user's point of view
uses of diagrams
uses of methods
uses of models
uses of requirements
uses of tools
vague
validation
various drafts
verbal requirements
verify
versions
Versions and Drafts
Very little documentation
vested interest
visible requirements
Volumes
waffly documents
well-written
what actually happens
what is possible
what the customer had asked us for
what they are going to get
who does testing
whole company
whose job is it anyway?
wood from the trees
workflow
working as a group
working documents
workshop
writing
writing things down
written down
written requirements

Appendix G

The Ontology of Concerns



The Ontology of Concerns

Appendix H

The Taxonomy

Different Types of Contents	Problem Domain Descriptions	Business Processes	
		Operations	
		User actions	
		Tasks	
		Events	
		Assumptions	
		Rules	
		Terminology	Glossary
			Acronyms
		Volumes	Historical rates, volumes
			Estimates
		Company Background	
		Interfaces to other systems	
	Types of effects	Types of features	Standard features
			Unique features
		Types of transactions	Update
			Insert
			Delete
			Modify
		Types of information requirements	ad hoc reporting
			data collection
			data flow
			query processing
			reports
		Types of behavioural requirements	corner cases
			error cases
			events
			external events
			states
			Transitions
		Transformation requirements	
	Proposed solutions	Database	Database Contents
			Database Structure
		System Architecture	
		Prototypes	
		Feasibility analysis	
	Goals	Objectives	
		Business case	
		Non-Functional Requirements	
	Constraints	Behavioural Constraints	
		Design Constraints	
	Issues	Closed issues	Changes
			Reasons for changes
		Open issues	
	Document information	Document history	Versions and drafts
			Signoff dates
		Circulation list	Authorization list
		Authorship	
		Table of Contents	
		List of reference documents	

Appendix I

Summary Tables

Areas of Concern					configured solution	house solution	situations
Project	Product requirements are allocated to Projects which are related to Releases Priorities for the next release of the product are influential	The requirements are subject to informal Agreement with the target customer, and subject to formal Approval by the internal Client Organisation either at a higher level, or peer level	There is a formal agreement with the Customer, represented on the Project by a Product Manager There is an Agreed deadline and little scope for selecting requirements based on priority	The Scope of the project must be managed and controlled There is an agreed Deadline and milestones	The Scope of the project is very important Scope creep is a significant risk Speed of implementation is important, so there is an agreed Deadline	The Goals and Objectives of the Project are more important than the Scope, or Deadlines or Priorities There should be a Shared Vision of the Project Objectives Signoff may be used although there is no contract	The Scope of the Project is a central concern, as are the Goals and Objectives Interfaces to other systems may be key Approval of the Project within the Organisation hierarchy is an important concern
Users	Focus groups are often used in the discovery of user requirements	Key users from the customer side may help to define and/or validate requirements But it is difficult to get users involved	The role of user representatives is played by developers, not real users	Individual User Requirements are less influential than the Organisational requirements	Individual User Requirements are much less influential than the Organisational requirements Representative Users take part in workshops	Individual User Requirements are much more influential than in other situations But the level of User participation in Projects varies considerably	Users are always involved to some extent Project Stakeholders often include Users and their managers
Client Organisation	Different levels, for Product Planning, Project planning,	A combination of an external client and internal organisation Often seen as joint development	The client organisation is the external Customer	Negotiating between the Client Organisation and the supplier organisation is significant Client Organisation IS/IT personnel may also be influential	Organisational objectives dominate the requirements, for example, integration of Business Areas Issues often relate to mismatches between the Existing System and the Product functionality	The Client Organisation is that part of the larger organisation that requires and is going to use the system This is often associated with one or more Business Areas	Organisational objectives are extremely important Organisational Change may be a source of Issues to be resolved
Business Area	Not relevant unless it is the Application Domain If so, Functional areas and Business knowledge are important	A source of Common requirements for the Product		The Existing System may be a source of knowledge about the Business Processes and Workflow The Solution is seen in terms of Functions	Business Process engineering of the target Business Area(s) is key Some Business Processes may need to change (BPR) in order to achieve Organisational Objectives	Business requirements are a major concern The Existing System, changes to it, and Organisational Change are all important sources of Issues to be dealt with	The relevant Business Area(s) are important sources, as are Business Processes, Functional areas and Workflow in the Existing System
Application Domain	Knowledge of Regulations or codes of practice in the target market is an important source of Common requirements	Understanding of the application domain e g relevant Standards, etc The Product is state of the art in that application domain	Domains other than business areas A deep understanding of the application domain is an important source of requirements	Non-standard requirements may be significant Application expertise on the part of the supplier may be useful	Knowledge of the Application Domain is embodied in the Product in terms of data types and available functions	Non-standard requirements, particularly Information Requirements are often key	Application Problems and Domain Knowledge may be important sources of requirements
Product	Each release contains selected features, so the scope of the project is much less than the scope of the product	Features of the Product may interact with each other, requiring a high level of product expertise	Product functionality is understood in terms of its behaviour, how it responds to events, and its behavioural constraints		Product Expertise on the part of the supplier is an important source of Solutions The Solution is seen as Functions provided by the Product		Not relevant unless a Product-based Solution is being assumed
Customer	There are many customers, but not on the scale of 'mass-market' situations	Few customers, compared to vertical market situation Some of them may have considerable application expertise Customer representatives validate requirements	The external Customer may in turn have many customers, typically consumers	The Customer is the Client Organisation	The Customer is the Client Organisation	There may be a Project Sponsor who takes on the role of Customer	This is the only situation in which the Customer is the creator of the requirements document
Market Opportunity	Products evolve in Versions in response to market opportunities Bundles of features are developed to address different market	Additional market requirements help to add value to the product Time to market is critical Additional platforms may be another source of system	The project is not usually concerned with other opportunities, except as a source of new contracts	This is not a concern in solution-oriented situations			This is not a concern in problem-oriented situations

		Customer Market		Solution		In-house Solution	Document
Problem domain descriptions	Descriptions of the problem domain tend not to occur in the requirements document	The problem domain is complex It is not typically described in the requirements document Some aspects may be already described in standards documents or glossaries of acronyms	These descriptions frequently appear in the introductory chapter, rather than throughout the document The problem domain often refers to the equipment in which the software is embedded, as well as to the situation of use	Descriptions of business operations and functions form the framework used to organise other requirements statements	Detailed descriptions are made of the relevant Business Processes based on an industry reference model of standard Business Process descriptions but do not appear in the requirements document	These represent business processes and functions, user tasks and business events, and form the bulk of the requirements text	Used mainly to outline the scope of the problem domain The company background, and volumes of transactions, etc
Statements of the required effects	These statements are often organised around features, with priority numbers, interdependencies, and other attributes	These are often organised around features, either standard features or features that are unique to the product or company Interactions between features are also stated	These denote features, or behavioural requirements, referring to the interaction/behaviour of the software in its environment	Statements describing transactions and information requirements form the bulk of the requirements text	Transactions and information requirements written in terms of the chosen product, and often closely tied to the proposed solution	Transactions and information requirements are stated, often in the context of representations of the problem domain	May or may not be very detailed, but may have transactions, information requirements and non-standard requirements
Proposed solutions	These do not commonly occur in the requirements documentation of vertical market products	May include the proposed architecture of the product	A proposed or actual system architecture may be defined in the requirements document	Often includes the database structure and contents or description of a prototype solution, such as the user interface	Solutions are selected from the range of standard functionality and database formats which come with the chosen product	Database contents and/or database structure and references to prototypes are often included	These are not really applicable in a problem-oriented document
Recorded issues and changes	Issues may concern the priority of features, and lead to changes	Issues relate to specific features or lower-level requirements, and might be concerned with adherence to standards	These are extremely important because it is a contract, and because changes cannot be made when the equipment is in production	Issues are extremely important because it is an organisational situation Changes are central because there is a contract, and these are documented after a particular version of the document, with reasons for the change	Issues are often related to required effects which may not be directly supported by the product These and the changes that ensue are recorded separately	Issues arising from organisational changes and from users' concerns about changes to the existing system may be noted	Not applicable in the final document, unless there are still important open issues to be resolved by the solution provider
Defined goals and objectives	Neither organisational goals nor system goals, such as non-functional requirements seem to occur in the requirements document in this type of situation	System goals such as non-functional requirements are important in this type of document	System goals such as non-functional requirements are important in this type of situation	Organisational goals are more significant than system goals and may be expressed in terms of a business	The business case for the system expresses organisational goals These are typically associated with improved business practices supported by the functionality of the system	The business case for the system is important Organisational goals may be linked to high-level requirements System goals are not considered appropriate components of the document	The business objectives of the required solution and the high-level non-functional requirements are important sections
Specified constraints	Neither solution constraints nor application constraints tend to appear in the requirements document	Application constraints, such as behavioural constraints, are significant	Both behavioural constraints and solution constraints are specified	Application constraints may be represented in the database design, or separately, as business rules Solution constraints such as interfaces to other systems may be defined	Defining which of the variety of application constraints supported by the product are needed is an important task, but these are recorded separately, with the Business Processes	Solution constraints are unusual Application constraints may be modelled in the problem domain descriptions, or as business rules	Solution constraints are typically an extremely important section of the document
Recorded agreement	Agreement is in the form of Approval by the relevant levels of the organisation The Approval given applies to the list of features to be included in the project, and their attributes	Agreement takes place at two levels Informal Agreement with the target customer(s) and formal internal Approval	The agreement is a formal contract between the customer and the supplier	The agreement is a formal contract between the customer (client organisation) and the supplier (solution provider) The document is a record of the agreement reached between them on the required functionality and scope of the project	The agreement is a formal contract between the client organisation and the solution provider, a record of the agreement on the scope of the project It is a contract to provide the specified functionality within the constraints of the chosen product	The agreement an informal agreement reached between the developers and their clients The document may or may not be a record of it	The final document is approved by the people in the organisation who have the authority to spend the funds allocated to the project