An Empirical Investigation into Management and Control of Software Prototyping

Liguang Chen
(BSc Mathematics and MSc Information Technology)

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

June 1997

Bournemouth University

ABSTRACT

In response to the so-called 'software crisis', software prototyping has been widely used as a technique in various stage of systems development since the late 70's, and, with the growing sophistication of 4GLs tools and environments, it has becoming a popular alternative to conventional development approaches. A study of the literature revealed that, unlike tools and environments, the management and control of software prototyping practice has been widely reported as being problematic. The study also suggested that there were very few reported studies of prototyping projects in practice. In order to contribute to the understanding of the management and control of prototyping, it was therefore decided to conduct an empirical study.

The empirical investigation comprises three interrelated stages: preliminary survey, field modelling and semi structured interviews. The findings of each stage provided inputs and formed a base for the following stage. From the survey to practitioners it became apparent that the concerns of the literature, regarding the management and control of prototyping projects, were justified. The next stage involved a detailed study using process modelling techniques of ten prototyping projects at eight software development organisations. This was then followed up by semi structured interviews of managers and prototypers at five organisations. In addition a number of documents, minutes and standards were also analysed, and personality tests conducted.

The main lessons learnt include the 'process diversity', the inadequate methods and standards, and lack of quality control, particularly in regard to future maintainability and extensibility. Recommendations are given for each key management and control area identified, including team selection, initial requirement gathering, prototypes building, change requests and quality controls.

Finally the thesis concludes that further work should be extended to areas such as developing 'lean methods' and an easy to use toolset for better management and control of the process.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CHARTS AND DIAGRAMS

ACKNOWLEDGEMENTS

My thanks first go to all the organisations and their staff, especially to those who had extended their collaboration throughout various stages of the investigations, obviously, without their contributions and assistance this research would have been impossible.

I would also like to acknowledge the support of Bournemouth University in the form of a bursary (1993-1997).

I have been most fortunate to be supervised by Professor Martin Shepperd, to whom I feel immensely grateful. Martin has helped me from providing adequate research training at the beginning and timely advice during the investigation, to giving numerous comments and corrections throughout writing up of this thesis. I have also greatly benefited from Dr Pam Mayhew, my second supervisor, with her insightful suggestions during the work and many useful comments on the draft thesis.

My gratitude also extended to all my family members, a number of friends and colleagues, without their encouragement and help the work would not have reached completion.

Chapter 1 Introduction

SYNOPSIS Management and control issues in software prototyping have been recognised as problematic for more than a decade, yet little empirical work has been done in this area. The objective of the investigation is to identify the key factors which contribute to the effective managing and control of such a process and provide possible solutions. This chapter covers the research aims, the scope and limitations, general terms and definitions as well as chapter previews.

1.1 Introduction

The cry of 'software crisis' has been heard since the 1970s and is still far from fading away. The problems have been that the software product is late, over budget, incorrect, incomplete and does not meet users' requirements, or at the worst, is never delivered. The main reasons behind this are (Carey 1990):

- users seldom have a clear, concise understanding of their informational needs until the system is built;
- the traditional function of specification is a narrative description of an information system that is technical and time consuming to read;
- large development teams cause ineffective communications;
- even if systems developed in the traditional manner function correctly, they may be difficult to learn and use;
- conventional approaches emphasise documentation, which is time consuming and difficult to change.

Software prototyping is one technique that attempts to address these problems and provide possible solutions. With the advent of 4GLs, the production of code can be performed much more quickly and cheaply, allowing an early version of the system to be put together and tried out on

the user before the target system is built.

Software prototyping has since been widely practised either as a technique of, or more recently an alternative approach to systems development. Its applications, as a technique, have been seen in almost all stages of systems development cycle from feasibility, requirement gathering, analysis and design to implementation and maintenance. In recent years, it seems that it has increasingly been used as a new alternative systems development approach — popularly termed RAD (Rapid Application Development).

Research in software prototyping is thus far mostly in the area of prototyping languages, tools and environments (Mittermeir 1982; Hollinde 1984; Kruchten and Schonberg 1984; Leibrandt and Schnupp 1984; Venken and Bruynooghe 1984; Luqi 1992). One key issue is to find effective ways to automatically generate prototypes from partial high level specifications for early evaluation, another is to develop domain specific models for automatic prototype production. However, much of the evaluation of such languages and tools seems limited to laboratories.

Since the late 1980's more and more attention has been drawn to the management and organisational issues of prototyping practice (Chapin 1983; Janson and Smith 1985; Pliskin and Shoval 1989; Pliskin and Shoval 1989; Mayhew 1990; Mayhew and Dearnley 1990; Mayhew and Dearnley 1990; Lim and Long 1992; Pliskin, Romm et al. 1993; Baskerville and Smithson 1995; Hardgrave 1995), which suggests that the problem with software development is not just about the effective tools and environments but the lack of good management practice. The main concerns include CR (Change Requests) control, effective methods, flexible project management, and organisational commitment. However, few empirical studies have been carried out in the field, especially to look into the management and control aspects of prototyping process.

Finally, a recent report has shown the significant imbalance between theory and practice in software engineering (Baskerville and Smithson 1995). The fact, as Potts (Potts 1993) pointed out, is that "most of the research done so far is failing to influence industrial practice and the quality of the resulting software".

1.2 Motivation and Objectives for Study

1.2.1 Motivation

As indicated above, software prototyping is a widely practised technique and/or an alternative approach for systems development, and there are many tools or environments available for the production of prototypes. Yet software prototyping, especially in respect of process management and control, is far from mature. This study is therefore undertaken from the following considerations: firstly, it has been reported that the management and control of software prototyping practice are problematic. The main problem is the lack of effective methods and standards, although some work has been done in this area (Shoval and Pliskin 1988; Pliskin and Shoval 1989; Pliskin and Shoval 1989; Mayhew and Dearnley 1990; Smith 1991; Gutierrez 1993), and many issues such as process iteration, CR controls and participant responsibility sharing were raised, yet no in-depth empirical study has been carried out in the field.

1.2.2 Objectives

The overall objectives of the research are:

1) to have a better understanding of the management and control of software prototyping process in terms of:

- what are the practical concerns of practitioners;
- how software prototyping is carried out in practice, e.g. who is involved, what they do, and how they interact;

2) based on 1) to provide practical, useful guidelines for effectively managing and controlling the process. The guidelines will include guidance on what are the problem areas and their associated key factors. They will also include the recommendations on do's and don'ts for better management and control practice of the process.

Therefore, it is hoped that the results of this study will provide, not only practical and useful knowledge for prototyping practitioners, especially

the managers improving their management and control practice, but also a base for further study.

1.3 Proposed Research Methodology

Case study is one of the most widely used methods for empirical study. It is a good vehicle for investigating a contemporary phenomenon within its real-life context, and it allows a researcher to learn and reason about a real-life process without interfering. It relies on multiple sources of evidence, it benefits from the prior development of theoretical propositions to guide data collection and analysis, and it also provides a ground for further theoretical propositions.

Given the motivation and objectives of the study, case study is most suitable and therefore chosen as the overall research strategy for the empirical study. This is reflected in the proposed three-staged methodology which consists of: Preliminary study (questionnaire survey), Field Modelling, and Further Investigations (semi-structured interviews).

The preliminary survey involves designing, sending, collecting and analysing a simple and easy to answer questionnaire. The purposes are: (a) to get a general opinion and find out the practical concerns from industrial practitioners, and (b) to build contacts for following up.

The field modelling involves using Role Activity Diagrams (Ould and Roberts 1986) to model real software prototyping processes, which aims to develop further understanding of both the problems that surfaced from the survey and the process as a whole.

Based on the field modelling, the further investigation — the third and final stage of investigation — goes one step further. It involves collecting and analysing more focused and detailed information by extensive interviews with the process participants, such as project managers and developers. Some other sources of information like project documents and company standards are also used, together with personality testing.

1.4 General Terms and Definitions

Terms used in this thesis are mostly those either widely used and having a 'consensus definition' within the software engineering community, such as quality control and quality assurance, or those commonly seen terms that have obvious meaning such as process participants and role responsibilities. In either case, they are used without further explanation. However, most new terms will be defined in relevant chapters and some terms which are essential to the study are defined as follows.

Software Prototyping Here software prototyping is considered to be an iterative process which involves quickly constructing one or more working models (prototypes) of the whole, or parts of the final system. Software prototyping is mainly used to serve at least one of the following purposes: (a) exploring user requirements; (b) clarifying user requirements; (c) refining user requirements; (d) experimenting with design alternatives and system requirements; and (e) an alternative systems development approach.

RAD It stands here for Rapid Application Development or Rapid Adaptive Development. It should be noted that, in the author's opinion, RAD is the use of software prototyping as a system development approach.

RADs This term stands for Role Activity Diagrams. It is a set of notations for process modelling [see chapter 5; Appendix C], which serves as the process modelling tool in the study.

Process Models The term 'process models' in the thesis means 'software process models', which is defined as a purely descriptive representation of parts of the software development process. This software process is the collection of related activities, seen as a coherent process subject to reasoning, in the production of a software system. Generally, there are two different types of process models: one is derived from, and a representation of, the existing software development, and the other is prescriptive in that the model is derived from theoretical or abstract considerations and imposed on the software development process.

If the term 'process model' appears on its own it always refers to the type of model that describes the existing software process. 'Reference model' is normally used to denote the type of model that is derived from abstract consideration.

RADs models The RADs models represent the software processes that are modelled using the RADs. As all the processes investigated were modelled by RADs, the Process Models and the RADs models are mostly the same in the context of the discussion in the thesis.

Field Modelling It refers here to modelling (using the RADs) an existing software process by observing a process and interviewing process participants.

1.5 Research Scope and Limitations

As with any other study, there are limitations to this study. Here is a brief account of the research scope and its limitations.

1.5.1 Focus of Study

As discussed in section 1.2, the study is focused on the management and control of software prototyping at the organisational level, that is to learn and reason about the process in terms of 'who', 'what', 'when' and 'why'. The technical aspects of the process such as prototyping languages, tools and environments are not included in this study.

1.5.2 Type of Software Prototyping

The study aims to include both software prototyping as a technique for one or more stages of structured systems development, and an alternative approach to systems development. The emphasis is given on the latter use for it is more problematic as discussed in chapter 2, the literature review.

1.5.3 Application Domain

The study is centred on information systems because this is the area where software prototyping has been largely used and mostly reported as being problematic in terms of process management and control (chapters 2 and 4).

1.5.4 Participants and participating companies

There was little choice of which company to include and the number of the people to participate. The significance is discussed in chapters 5, 6, and 7.

The participants of the investigation were mainly project managers and prototypers. Others involved were design managers, IT managers, sales managers, and developers. Customers and users could not be included in this study owing to the limited control over the investigation and man power available.

1.5.5 Project Size

The projects being studied in depth are mostly small to medium sized. On average they are about six to nine months with teams of two to five people. They are either stand-alone or part of larger projects, or extensions to existing systems.

1.5.6 Research Methods

Case study is chosen as the overall methodology for the work (chapter 3). The limitation is that it is hard to have enough control over the investigation in terms of accessing company information and staff. One result of this is that different data are gathered from different companies, so it makes harder for data analysis.

1.6 Thesis Organisation and Overview

An overview of the thesis organisation is given to assist the reader in finding relevant material, as well as understanding the relationships between the chapters.

1.6.1 Research Background and Foundation

This part of the thesis comprises the first three chapters that deal with the fundamental background issues to the research.

Chapter two — Literature Review— starts with a historical account of software prototyping in relation to problems in software development. The crucial point here is its recent use as an alternative approach to conventional systems development. The state of the art of the technology is brought in to view by looking at the definitions, the merits, the applications, the tools and the environments. Current research in the field is broadly discussed and attention is given to the related management and control issues. This chapter finishes with a discussion of the possible research opportunities in the area of management and control of software prototyping, which sets out the research direction for the empirical investigation.

Chapter three deals with the research methodology issues of the investigation. It argues about why case study is chosen as our overall research strategy and lays the framework to the investigation. First, the three most commonly used strategies for empirical studies are closely examined. The result is that case study is the most appropriate strategy to adopt, as the research is about seeking 'how and why' answers to the problems of concern. Multiple cases are chosen over single case study considering the trade off between the degree of generalisation and the level of details that each method offers. Secondly, this chapter gives a flavour of the overall structure of the research by describing and explaining the particular methods used at each stage of the investigation, the relationships among the methods and their rationale.

1.6.2 Empirical Investigation

The second part of the thesis contains chapters 4, 5, and 6, and each of which deals with one stage of the investigation.

The Preliminary Survey (chapter 4) is the first step of the investigation. This survey aims at obtaining a broad view from industrial practitioners about their software prototyping practice in general and revealing their real concerns about the management and control of the process in particular. It also functions as a readjustment and reassurance of the initial objectives arrived from the literature review as well as a bridge to the next stage. The chapter describes these aims, the questionnaire design, and the results. The main finding of the survey is that management and control of the process is the overwhelming concern, which largely confirms the results of the literature review.

Based on the findings and resulting industrial contacts, the next stage of the investigation — Field Modelling — has been carried out, which is the subject of chapter 5. It discusses the rationale behind the field modelling — to learn more about the process and to focus on the problem areas on one hand, and to provide the framework for further detailed study on the other. It also explains why RADs (Role Activity Diagrams) are chosen for the field modelling. The main reason is that the RADs suit the purpose of the study well: the diagrams clearly show an overall picture of process in terms of 'who', 'what' and 'when'. The chapter ends with a summary of the findings of the field modelling, of which the most striking finding is the diversity of the resulting process models.

Chapter 6 — Further Investigation and Data Analysis — covers the final stage of the investigation. The chapter first describes the aims and methods. The main aim is to seek answers for the questions generated from the field modelling, i.e., to find and reason about why things happened or didn't happen in their particular way; the main method used is by interviewing the key members of their development team (i.e. the managers and developers). Many insights were gained as the result of the data analysis, which were summarised in the form of recommendations in Part Three of the thesis.

1.6.3 Conclusions

Conclusions — chapter seven — forms the third and final part of the thesis, which consists of review, recommendations, further work and final conclusions.

The review section gives a summary of the main findings at each stage and an overall evaluation of the adopted methodology: its strengths and its shortcomings.

The recommendations are given in two parts: recommendations on key management and control areas and recommendations on process modelling. The first part presents the main results of the investigation, it identifies the four key control areas: project initiation, initial requirement gathering, CR (Change Requests) control and user participation. The second part of the recommendations are about process modelling, which are intended to share some useful lessons learnt from modelling software prototyping projects using Role Activity Diagrams.

In the further work section, six areas are identified as a direct result of the study. They include investigating customer and user perspectives, developing and evaluating a measurement programme based on the study, developing simple and flexible methods and standards for software prototyping according to appropriate process classification, and an easy-to-use tool providing guidance to prototyping project managers for managing and controlling the process.

In the Final Conclusion, main contributions and shortcomings of the study are recapitulated.

1.7 Summary

As a response to the software crisis, software prototyping has been popularly used as either a technique, or as an alternative systems development approach for more than a decade. Over the years, more and

more sophisticated 4GLs and CASE tools and environments have become available for fast production of working prototypes. However, the 'unknown' management and control practice of software prototyping has in recent years caused increasing concerns. Moreover, there is little in-depth empirical study being carried out in this area (see chapter 2).

In such circumstances, this empirical study aims to gain a better understanding and reasoning about the management and control process of software prototyping. To achieve the overall aim, a case study approach is employed as the research strategy. The methodology is designed as a three-stage investigation, which comprises the questionnaire survey, the field modelling and the semi-structured interviews. It is worth noting that the field modelling is a novel approach to problem of this kind, in the sense that there are few published empirical studies of prototyping and none that use process modelling techniques.

The main focus of the study is centred on the management and control process of evolutionary prototyping in small and medium information systems development and the main limitation is the lack of customer and user perspective.

Chapter 2 Literature Review

SYNOPSIS This chapter is a literature review of current software prototyping practice and research. It starts by looking into the rationale behind and the commonly used definitions and classifications for software prototyping. A working definition is then given for purpose of this study. There follows an overview of software prototyping languages, tools and environments. More importantly, it draws attention to the increasingly seen problematic area — management and control of prototyping, the focus of the discussion is on issues such as exploring the various ways in which this may affect those people associated with software prototyping, managing and controlling changes to the software during prototyping process, controlling the iterations and developing effective prototyping methods. Finally, it concludes that an empirical study is needed in order to better understand and, therefore, improve the management and control of software prototyping.

2.1 Introduction

Prototyping has long been a standard technique in conventional engineering fields. However, this has not been feasible in software engineering until the 1980s because of the large amount of coding effort for producing a software prototype.

With the advent of 4GLs, the production of code can be performed much more quickly and cheaply, allowing an early version of the system to be put together and tried out by the user before the target system is built. Moreover, a great deal of demand for the use of software prototyping has emerged due to the increasing numbers of systems being built that fail to satisfy actual customer needs (Berosff and Davis 1991; Smith 1991; Spence and Carey 1991; Davis 1992; Kautz 1994; Brooks 1995). Since then software prototyping has been a popular and widely used technique or an

approach to systems development. However, it is still a relatively new area of software engineering, and the understanding of the subject is still far from mature.

To date the term 'software prototyping' has been defined and used variously. Some see prototyping as a series of approximations towards a final system from a evolutionary point of view; some view it as a way to address customer and/or user participation; others address the learning process between the users and developers (as Mayhew explains 'the crucial component of this definition is encompassed in the word 'learn', if nothing has been learned during the prototyping process it has been a waste of time and effort.' (Mayhew and Dearnley 1990)). Apparently, they all have a different focus that depends on the way in which prototyping is used, which will be further examined in section 2.2 below.

The chapter is arranged in this way: section 2.2 looks at current software prototyping practice including: some definitions seen in the literature and a working definition for the study; the reasons behind software prototyping, the types and applications, the tools and environments. Section 2.3 concentrates on software prototyping research in general and management and control of the process in particular. In section 2.4, conclusions are drawn based on the findings, and the chapter ends with a summary about current software prototyping practice and the direction for the research.

2.2 Current Practice

This section examines the current practice of software prototyping in terms of concepts, application, tools and environments.

2.2.1 Definitions Of Software Prototyping

As mentioned in section 2.1 above, software prototyping is a relatively new technology and the concepts are still evolving. Thus far there seems no consensus about what software prototyping is, where and how it should be used. However, a working definition is needed for the purpose of this study. Before giving the working definition, let us first have a look

at some of the definitions in use:

"The construction and analysis of an executable model that approximates a proposed system" (Luqi 1992).

"The idea behind prototyping is to include users in the development cycle" (Stahl 1986)

"Prototyping is the process of quickly building a model of the final software system, which is used primarily as a communication tool to assess and meet the information needs of the user" (Carey 1990).

More recently, Reinhard Budde et al. (Budde and Zullighoven 1992) defined software prototyping as: *"an approach based on an evolutionary view of software development process as a whole. Prototyping involves producing early working versions ("prototypes") of the future application system and experimenting with them".*

Other similar definitions can be found elsewhere (Kammersgaard 1984; Monckmeyer 1984; Mayhew 1989; Bischofberger and Pomberger 1992; Lim and Long 1992; Gutierrez 1993). All in all, it appears that these definitions emphasise different aspects of software prototyping: either a way for quick construction of an executable model or a communication tool between user and developer or an evolution process. Moreover, from these definitions it is not clear whether software prototyping is one of many new techniques for conventional development or an alternative system development methodology which should have defined development methods as well as associated management and control mechanisms.

Although the latter use of software prototyping appears to be more popular, the former use has had a much longer history in practice. Both views are included in the study in order to have a broad understanding of the subject, a working definition therefore is defined as follows:

Software prototyping is an iterative process of building a working (executable) version of a partial or whole system, which is either used as a technique in one or more stages of conventional software development or

as an alternative approach to software development.

Based on this view of software prototyping, the following discussion provides a systematic look at current practice and research of software prototyping.

2.2.2 Reasons for Prototyping

Why do we use prototyping in software engineering? Is there any need for it? The answer may be best illustrated by Boar's (Boar 1984) justification:

'Most currently recommended methods for defining business system requirements are designed to establish a final, complete, consistent, and correct set of requirements before the system is designed, constructed, seen or experienced by the user. Common and recurring industry experience indicates that despite the use of rigorous techniques, in many cases users still reject applications as neither correct nor complete upon completion.'

Although the above quotation represents an extreme view, its fundamental argument is sound. Software prototyping is a response to problems which are encountered by software developers, as Boar illustrated above. These problems can be further summarised as follows (Carey 1990):

- users seldom have clear, concise understanding of their informational needs. Therefore, they cannot pre-specify the requirements. Once they begin to use a system, however, it is clearer to them what they want.

- the traditional specification is a narrative description of an information system that is technical and time consuming to read. Static graphic techniques (such as dataflow diagrams, and data dictionary entries found in the structured approach) once thought to be the solution to communication, cannot demonstrate the workings of a live dynamic system.

- the larger the development team, including user representatives, the more difficult communication becomes. Semantic barriers and lack of physical proximity and time inhibit the ability of all members of the team to have a common understanding of the system being developed.

- even if systems developed in the traditional manner function correctly, they may be difficult to learn and use.

- both traditional and structured approaches emphasise documentation, which is time consuming and difficult to change.

All of these problems suggest that some alternative technique is needed. Prototyping is one technique that attempts to address these problems and provides possible solutions (Boehm 1988; Berosff and Davis 1991; Bischofberger and Pomberger 1992).

2.2.3 Advantages and disadvantages of software prototyping

2.2.3.1 Advantages

Software prototyping has being used in industry with varying degrees of success. The advantages claimed by its advocates are (Carey 1990; McDermid 1991):

- systems can be developed much faster;
- development costs are reduced;
- user requirements are easier to determine;
- user and developer communication is enhanced;
- systems are easier to learn and use;
- programming and analysis effort is much less;
- development backlogs can be decreased;
- end-user involvement is increased;
- the resultant system is the 'right' system and needs little changing;
- it is a good vehicle for training;
- can be used as a medium to evaluate design alternatives;
- can be used as a test oracle.

All of these positive attributes make prototyping an attractive approach for system development. Many organisations have adopted some use of prototyping within their development life cycle (Livesey 1984; Nosek 1984; Janson and Smith 1985; Budde 1992; Davis 1992; Lichter, Schneider-Hufschmidt et al. 1993). However, not all these benefits are proven nor are the inevitable result of prototyping, and few report is on how these benefits can be achieved, which will be discussed in section 2.3.

2.2.3.2 Disadvantages

There are also disadvantages associated with software prototyping (Morrison 1988; Graham 1989; Carey 1990; Hilal and Soltan 1992). The main problems are seen as follows:

- **Undue user expectations.** The ability of the systems group to develop a prototype so quickly may raise undue expectations on the part of the user.

- **Lack of attention to good human factors.** Many application generators have rather inflexible screen and menu formats, which often inhibit the use of good human-factors techniques, so the use of application generators as prototyping tools does not ensure the resultant systems will adhere to human-factors guide-lines (Kellner and Hansen 1989; Long 1996).

- **Inattention to analysis.** Prototyping tools are relatively easy to use and produce quick results, which may lead to insufficient analysis. This may result in incomplete functionality, poor design.

- **Over-relaxed quality control.** Prototypes are often produced by compromising on the quality controls such as quality plans, various formal reviews, and quality checks. This may result in unknown quality.

- **Lack of attention to adequate documentation.** Because of the emphasis on providing fast software solutions for customer and/or

users, attention is often given to produce a working prototype rather than documentation. However, lack of adequate documentation may seriously reduce systems maintainability.

All these advantages and disadvantages of prototyping are further explained along with prototyping tools and techniques in section 2.2.5.

2.2.4 Types of Prototyping

The following examines some commonly seen classifications with relation to their management and control models.

Contrary to popular belief, there are a number of different ways of categorising prototyping. In this section, two most common categories, i.e., throwaway prototyping and evolutionary prototyping are described. Furthermore, comparisons are made on the differences with traditional approaches. In addition, another type — operational prototyping (Davis 1992), which is derived from the two base types — is elaborated. Some other different classification schemes or terminology can be found elsewhere (Nosek 1984; Rzevski and George 1984; Mayhew and Dearnley 1987; Tate 1990; Martin 1991; McDermid 1991; Budde 1992; Mock and Hodge 1992).

2.2.4.1 Throwaway Prototyping

Throwaway prototyping can be seen as an iterative process of requirements clarification by means of a throwaway prototype. A throwaway prototype is built as quickly as possible for the purpose of learning and clarifying poorly understood system and/or user requirements, which is subsequently discarded after the desired information is learned.

The requirements specification that incorporates what was learned should be the direct results of such a exercise. A full-scale system based on that specification could then be built. This can be illustrated as follows (Dearnley and Mayhew 1983):

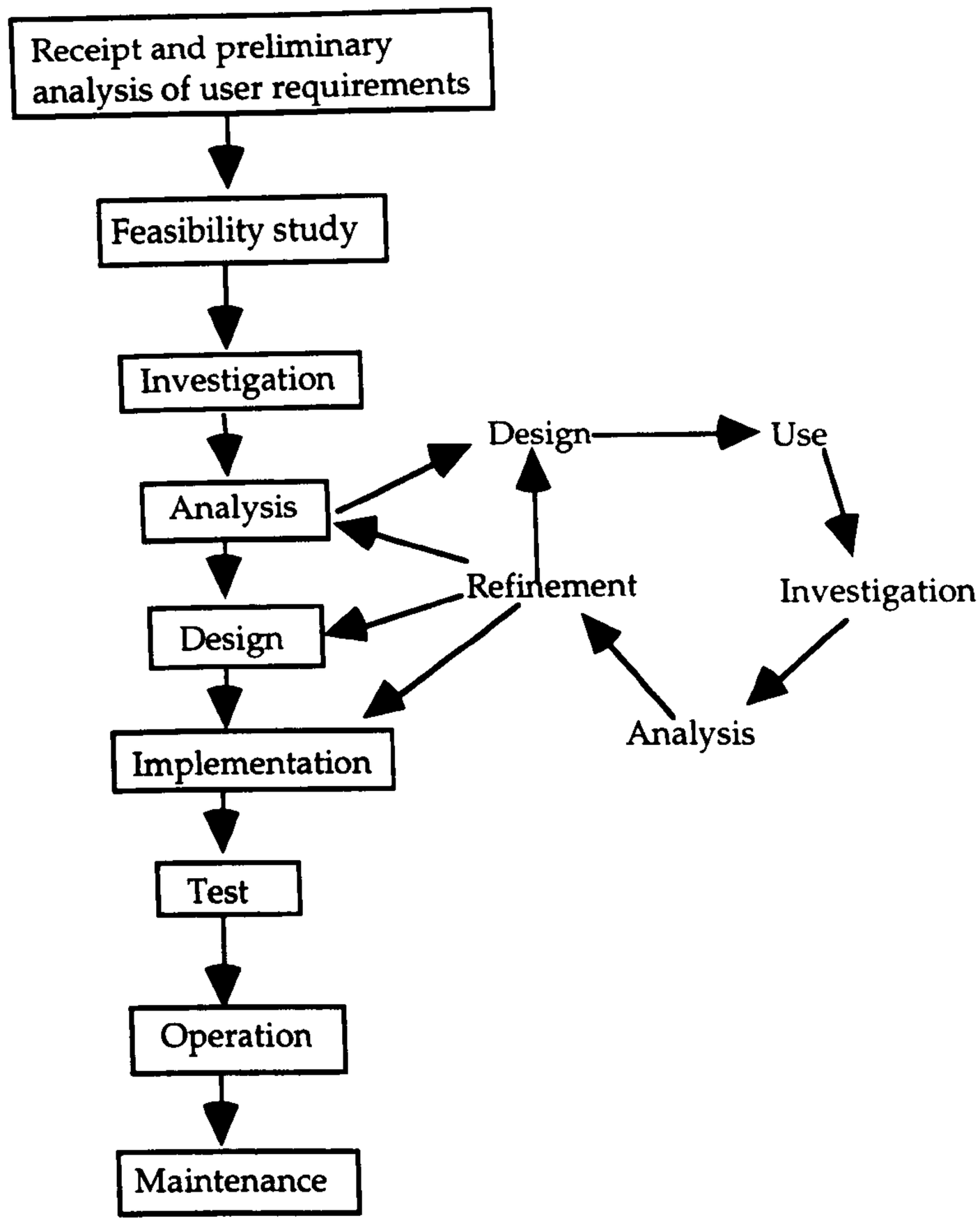


Figure 2.1 Model Of Throwaway Prototyping

This shows how throw-away prototyping can be incorporated into conventional phase-oriented software development. As figure 2.1 indicates, once the prototyping process starts, it goes iteratively through design, use, investigation, analysis and refinement of the prototypes until users' requirements are met. It then may exit to three different phases:

- to analysis if it is used only for part of requirements clarification;
- to design if all requirements are incorporated in the final prototype;
- to implementation if the design of the final prototype is highly refined.

It is worth noting that this model only shows the typical use of throwaway prototyping — the requirements analysis. In fact, it could be utilised in almost all stages of the waterfall model (Budde 1984; Budgen

1984; Mistrik and Ivan 1984; Kraushaar and Shirland 1985; Blum 1986; Gomaa 1986; Harker 1988; Shoval and Pliskin 1988; Alexander 1990; Bischofberger and Pomberger 1992; Budde and Zullighoven 1992; Hilal and Soltan 1992; Hardgrave 1995; Buti 1996; Cerpa and Verner 1996).

As pointed out in section 2.2.3.2, one of the most commonly associated shortcomings of the throwaway prototype is its potential inconsistencies with the final system, i.e., what the user sees may not be what the user gets.

2.2.4.2 Evolutionary Prototyping

Evolutionary prototyping is the process of evolving an initial prototype towards a final product. It has arisen in response to the impact of change on large software project, it is an attempt to ensure that an executable version of a system is available throughout a project.

Evolutionary prototyping is not merely used as a technique in the context of a single development project; it is a continuous process for adapting an application system to rapidly changing organisational requirements. This means that software development is no longer seen as a self-contained project, but as a process continuously accompanying the application. One consequence of this is that the role of the developer changes. They are no longer the "protagonists" of a self-contained project. Instead, they become technical consultants working continuously in close co-operation with the users to improve the application system.

In contrast to a throwaway prototype, an evolutionary prototype is built as an evolutionary basis. When the prototype is complete, the developer modifies the software-requirements specification to incorporate what was learned. The system is redesigned, recoded, and retested. This process is repeated until all parties involved are satisfied.

A model for evolutionary prototyping by Brice and Connell (Brice and Connel 1989) is as follows:

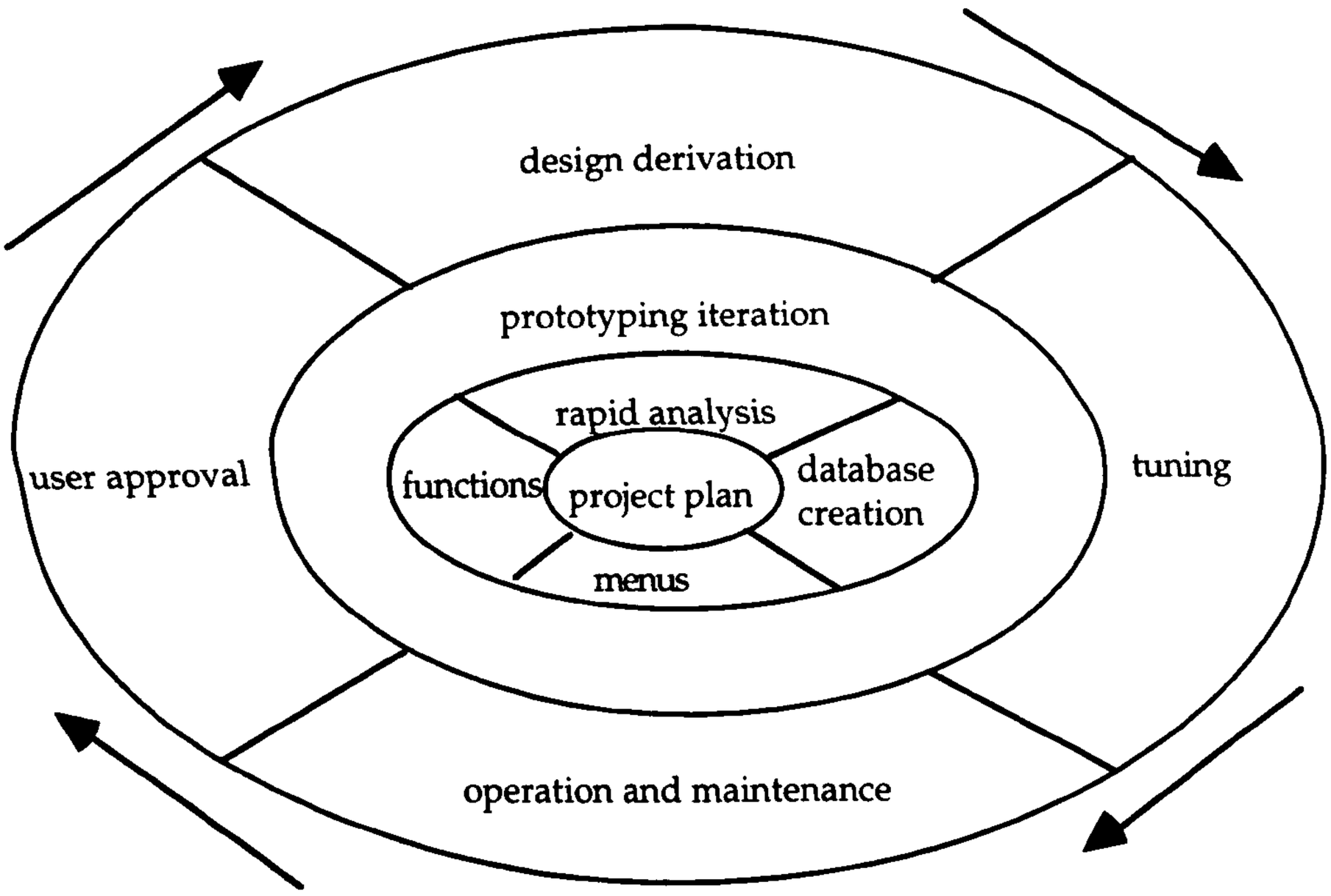


Figure 2.2 Model Of Evolutionary Prototyping

Apart from the apparent paradigm shift from the stage-wise to iterative process as indicated in figure 2.2 above, evolutionary prototyping differs from conventional software development in a number of ways (McDermid 1991) :

- often the duration of the prototyping project is much shorter, because of the use of 4GLs and CASE technology;

- hence, the milestones in an evolutionary prototyping project tend to be closer together. This is because the boundaries between analysis, design and implementation are overlapped instead of being distinct activities as in a conventional approach. Therefore, some of the tasks between phases will disappear, or at least be minimised;

- there is an additional milestone for preliminary requirements. Other milestones in the conventional software project will be replaced by milestones which are preceded by different activities (e.g. requirement specifications and system design will be replaced by rapid analysis and building prototypes);

- detailed functional requirements are not finalised until a working prototype is signed off by the customer;

- in the conventional software project, testing of performance is often intertwined with functional testing. In prototyping, they are distinct processes separated in time;

- tuning replaces the conventional implementation phase.

The primary difference is that conventional development attempts to build the entire set of requirements; whereas, evolutionary prototyping acknowledges that we do not understand all the requirements, it builds only those that are well understood first, and incorporates more complete requirements as the systems evolves.

The major problems associated are:

- **end-user computing.** While end-user involvement in the system development is positive, end-user computing may have some negative ramifications for system integration and database integrity (Kraushaar and Shirland 1985; Pliskin and Shoval 1989; Lichter, Schneider-Hufschmidt et al. 1993);

- **final system inefficiencies.** 4GLs have a reputation for generating less than optimum code in terms of efficiency and throughput. Care must be taken to predetermine whether the new system should be written with an application generator/prototyping tool or prototyped in a 4GL and then coded in a 3GL for maximum efficiency (Grant 1985; Martin 1986; Martin 1991; McDermid 1991).

However, the latter is less a problem with the increasing power of both hardware systems and software development environments (refer to section 2.2.5).

2.2.4.3 Comparison Of Throwaway and Evolutionary Prototyping

As discussed before, a throwaway prototype implements only poorly understood requirements and hopefully incorporates them into the well-understood class. An evolutionary prototype implements a well understood requirement, expecting users and developers to uncover previously unknown requirements.

Throwaway prototypes work well in isolation to verify relatively small parts of complex problems. Evolutionary prototypes work well when most of the critical functions are well understood.

2.2.4.4 Operational Prototyping

For some complex systems, developers may use throwaway prototyping for some parts and evolutionary prototyping for others, that is neither throwaway nor evolutionary prototyping alone is suitable.

In attempting to tackle the inadequacy of either approaches, Davis (Davis 1992) introduces operational prototyping, which offers a balance between the two by building throwaway prototypes selectively on top of evolutionary prototypes. The figure 2.3 below shows how operational prototyping works:

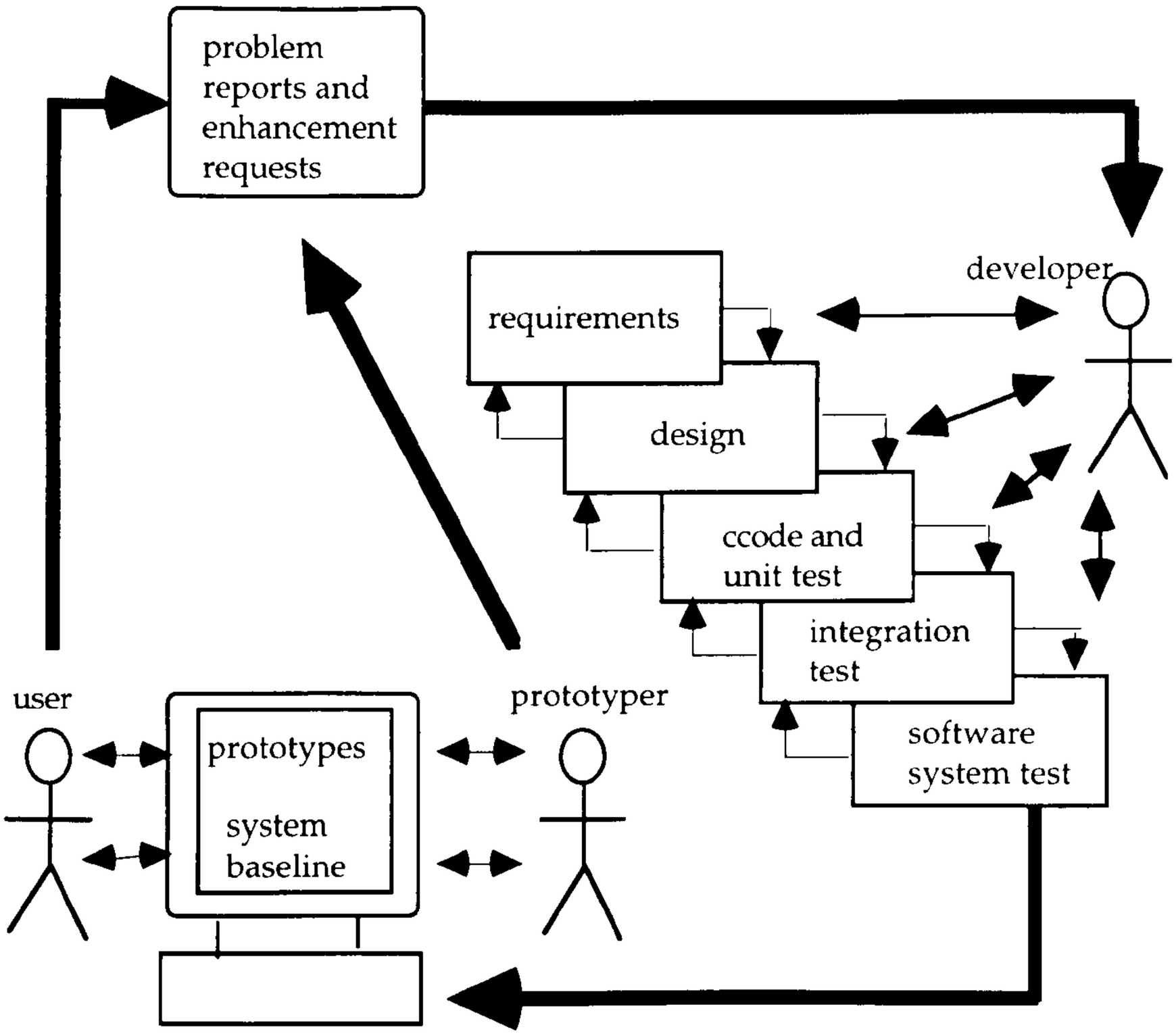


Figure 2.3 Operational Prototyping[1]

Figure 2.3 demonstrates the relationships between user, prototyper and developer. First, a set of prototyping sessions are conducted between user and prototyper, which are followed by recording requests for changes and/or enhancements. Changes are made via developer(s) based on the requests and then incorporated in the system baseline. This process will continue until all users' and system requirements are met.

This approach is consolidated by incorporating configuration management and quality assurance guidelines (Davis 1992), which attempt to combine effectively the advantages of both throwaway and evolutionary prototyping without reducing quality.

---

[1] adapted from (Davis 1992).

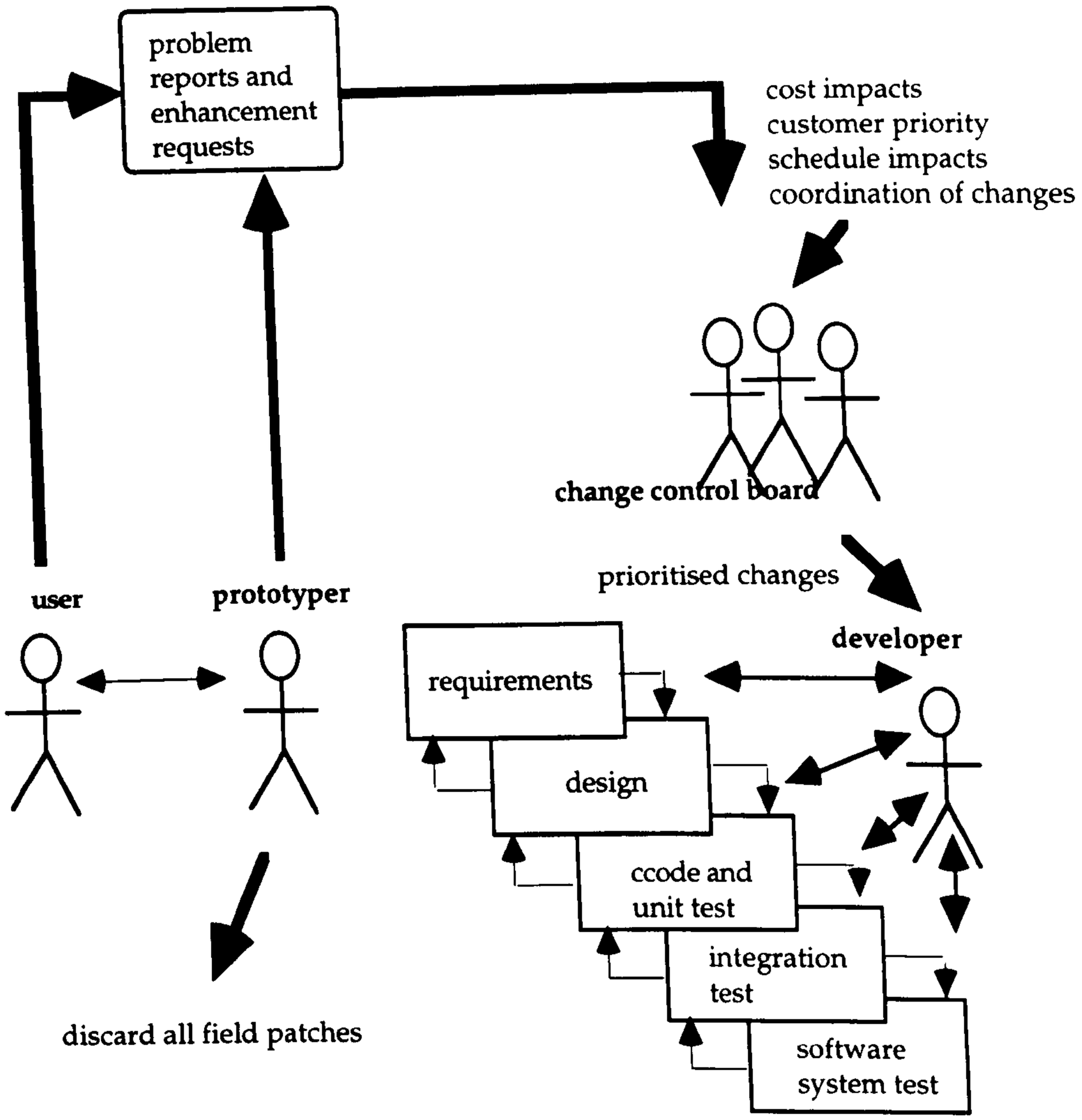


Figure 2.4 Configuration Management For Operational Prototyping[2]

Clearly the intention here is that product quality can be guarded via the 'change control board', therefore the 'throwaways' will not become part of the system being built.

2.2.5 Tools and Techniques for Prototyping

There are many tools and techniques available to conduct software prototyping. This section discusses the general characteristics of prototyping tools and techniques, as well as the principles to employ them for prototyping.

---

[2] adapted from (Davis 1992).

2.2.5.1 Fourth Generation Techniques

Fourth Generation Techniques (4GTs) encompass a broad array of data base query languages, program and application generators, and other very high-level, non-procedural languages (Pressman 1992). Because 4GTs enable the software engineer to generate executable code quickly, they are ideal for rapid prototyping.

1) Fourth-generation languages 4GLs are probably the best known medium which can be used for prototyping, certainly for the prototyping of commercial data processing applications. The reason that 4GLs are such a useful medium for prototyping is that they offer high-level facilities such as the very powerful facilities for retrieving data based on complex predicates. Finally, in terms of the ability to produce screens, 4GLs again offer very sophisticated facilities for defining the form of a screen in terms of colour, data entry fields, sophisticated line graphics, and the ability to characterise a field in terms of its contents, its mode and its relationship with other fields (Martin 1986; Martin 1991; McDermid 1991).

2) Very high-level languages (VHLL) A VHLL is one in which it is possible to express complicated operations using a small amount of written code. The disadvantage of such languages is that they tend to be inefficient in terms of run time or memory usage. In general, VHLLs are interpretive and interactive, offer a rich set of data objects and corresponding operations, contain a language notation which is short, concise and very expressive, and are normally supported by very powerful programming support environments and debugging facilities. It is the expressive power, together with the ease with which modification can be carried out, that make such languages ideal for prototyping. Some of them are APL, Lisp, Prolog (Venken and Bruynooghe 1984) and SETL. SETL is, a little known, but particularly effective for evolutionary prototyping as it allows the programmer to declare procedures and functions which operate in a simple, succinct way, but which can be refined to make them more and more efficient (McDermid 1991).

No VHLL is suitable for prototyping all application areas. For example, in prototyping an artificial intelligence application the natural choice would be Lisp, or perhaps Prolog. In prototyping a numerical application, for

example a large simulation problem, the natural choice would be APL.

There are programming languages which are oriented towards one particular application area, for example, stock control or process monitoring. The majority of application-oriented VHLLs are application generators. These are interpretative systems which interactively execute an application description.

Functional programming languages are another type of VHLL. Unlike conventional programming languages, such as Pascal, FORTRAN, and C/C++, they are mathematical in nature, and are an attempt to overcome the bottleneck problems that occur with conventional languages on multiprocessor computers (Hudak 1989; Hughes 1989). There are many functional languages available such as Miranda, FP, Scheme, and ML (Turner 1986; Hudak 1989; Hughes 1989). Of these ML is probably the best, as it contains useful facilities for modularization, data encapsulation and exception handling. Unfortunately, the vast majority of functional languages are currently only implemented on conventional, von-Neumann computers; consequently, they suffer from speed problems. Nevertheless, this does not disbar them from being an excellent medium for throwaway prototyping (Henderson 1986; McDermid 1991; Harrison 1995).

3) Object-oriented languages The earliest language with object-oriented properties was Simula 67. However, the most well-known object-oriented languages are Object Pascal, Smalltalk (Goldberg and Robson 1983), Eiffel (Meyer 1992) and C++ (Stroustrup 1988; Stroustrup 1991).

The importance of object-oriented languages for prototyping lies in a property known as inheritance. This allows the programmer to define a base object, say a plane, and operations on that object, and then with little effort to define new objects based on that object, for example, fighter planes and transport planes.

Object-oriented languages can be used by the developer to build up a library of objects and operations which are required in the application area and then, for each application to be prototyped, to develop the objects in the applications by using the inheritance facilities of the

language (Rumbaugh 1991; Shlaer and Mellor 1992). In essence, prototyping is achieved by very sophisticated software re-use technology. This leads on to the next approach to rapid prototyping.

2.2.5.2 Reuse Software Components/Product

Prototyping by reuse means to assemble, rather than build, the prototype by using a set of existing software components or products. A software component may be a data structure (or database) or a software architectural component (i.e., a program) or a procedural component (i.e., a module). An existing software product can also be used as a prototype for a "new, improved" competitive product. In a way this is a form of reusability for software prototyping (Redwine 1988; Basili and Rombach 1991; Lewis, Henry et al. 1991).

2.2.5.3 Formal Specification and Prototyping Environments

Over the past two decades a number of formal specification languages have been developed to replace natural language specification techniques (Ruiz, Harmelen et al. 1994). These formal languages are in the process of developing interactive environments that (1) enable an analyst to interactively create a language-based specification of a system or software, (2) invoke automated tools that translate the language-based specification into executable code, and (3) enable the customer to use the prototype executable code to refine formal requirements. Specification languages such as OBJ, VDM and Z along with many others have been tried to achieve an automated software engineering paradigm (Henderson 1986; Hekmatpour and Ince 1988; Borba and Meira 1997). Although such environments in theory may offer substantial hope for improved prototyping and software development productivity (Rombach 1988; Bidoit, Kreowski et al. 1991; Austin and Parkin 1992; Blazy and Facon 1995; Borba and Meira 1997), their practical use in software development is and will be still arguably limited by (a) the number of applications where requirements can be precisely described, and (b) the they are much more difficult to use than natural language descriptions.

Apart from the tools and techniques discussed above, there are some others approaches. Tool-set approach is one of them, which uses a software system to provide a series of processors and tools which can be linked together easily and are table-driven. For example, the utilities such as YACC and LEX of the Unix operating system are capable of processing tables of definitions and producing software very quickly (McDermid 1991).

Another medium which can be used for prototyping is associated with expert system technology. Apart from numerous free/shareware expert shells, many commercial products are available for expert systems development, such as Crystal (Intelligent Environments Europe Ltd.), PowerSmarts (Cognition Technology Corporation), Rete++ and VBXpert of The Haley Enterprise Inc., and most of the products can be ported on to various platforms. All these shells appear to have good interactive development facilities and hence are generally suitable for prototyping. However, their use has been criticised as limited by the number of rules that can be incorporated in the expert systems (McDermid 1991). More recent reports (Liebowitz 1994; Eom 1996) indicates that it is becoming less of a problem with the advance of the technology.

2.2.5.4 Choosing a Medium for Prototyping

As described above, there is quite a wide choice for carrying out prototyping. However, in historical terms, techniques and tools for prototyping functionality became available much earlier, and are therefore much more sophisticated than those for prototyping the HCI.

Although there are many uses of software prototyping (Budde 1984; Floyd 1984; Riddle 1984; Mayhew and Dearnley 1987; Tate 1990; Luqi 1992), they can be roughly divided into two areas of application:

1) Function Prototyping

For a commercial data processing application, the natural choice would be a 4GL, because there are many good implementations for a wide range of computers. Moreover, such languages are, on the whole, easy to learn. For prototyping in other application areas, one could use Unix as a tool

set, or use functional languages. For most developers the former would be the choice. While functional programming languages or logic programming languages are a good alternative, they do suffer from a skill shortage, particularly in functional languages, where a good understanding of mathematics is required (Henderson 1986; Hekmatpour and Ince 1988).

There are also many CASE tools becoming available for functional prototyping, among some well recognised and widely utilised (e.g. Teamwork, Composer, Principia/SSADM, Select OMT Professional[3]). In addition, a growing number of rapid application development environments have become available that are good prototyping mediums both for database and other more general application. This is further discussed in section 2.2.5.5.

2) HCI Prototyping

4GLs will still be the choice if a commercial data processing system is concerned. For other application areas, if programming language facilities still requires a large degree of work to produce interfaces, the choice would be some graphical hypermedia or multimedia systems which contain a programming language and allow the prototyper to control and sequence a dialogue with the customer (Chistensen 1984; Linton, Vlissides et al. 1989; Luqi and Y.Lee 1989). With the growing number of GUI tools available on all kinds of platforms, one is often spoilt for choice and care must taken for appropriate selection (Myers 1995; Myers 1997). In fact, many of the tools are not just capable of HCI prototyping, they are fully integrated environments for rapid application development.

2.2.5.5 Software Prototyping Methods and Standards

Well defined methods and standards for hardware prototyping exist for a wide range of industry, which is, however, not the case for software prototyping. As a result of the literature search and enquiries made to BSI (where most international standards are held from various standards organisations), apart from general quality standards such as ISO 9001 and TickIT (DTI 1992), no software prototyping or RAD methods and

[3]All these CASE tools along with many others have been formally evaluated by OVUM which can be fine at: http://www.ovum.com/.

standards were found that are accredited by any international standards body.

However, there has been some effort made to develop such standards at a more local level, for example, the "prototyping in an SSADM environment" (CCTA 1993) and the "Dynamic Systems Development Method" (DSDM) (Millington and Stapleton 1995). While the CCTA guidelines are confined only to SSADM, the DSDM guidelines are aimed at the UK RAD community. DSDM is further examined in the following section.

2.2.5.6 Rapid Application Development (RAD)

Broadly speaking, RAD is a kind of evolutionary or incremental prototyping which is employed as an alternative approach rather than just as techniques for systems development. However, since 1990s the term RAD has been popularly used by most of the business community and software vendors, and many new development toolkit and/or environments have been developed and are widely available in commercial market. Although RAD's popularity may well be partially a result of marketing, it does reflect, to a larger degree, the increasing demand from business world for flexible, fast 'time-to-market' solutions (Boyer 1995; Card 1995; Reilly and Carmel 1995; Editorial 1996; Robinson 1996).

RAD tools and environments are available across various platforms, but mostly are MS windows-based, and have been particularly aimed for client/server applications (Boyer 1995; Card 1995; Yap 1995; Lemieux 1996). Among the hundreds of RAD products, many are domain or application specific. There is little evidence or evaluation of their use in public domain. However, there are some popularly used environments such as PowerBuilder, Oracle Tools, Microsoft's Visual Basic and Visual C++, Boland's Delphi and C++ as well as Java tools (Lemieux 1996; Grehan 1997).

The apparent main advantages of these environments are their strong GUI and DBMS capabilities, and generally better performance by largely employing 3GLs over 4GLs for core functionality. Although they are all

general-purpose tools their use still should be judged depending on application areas, developers, technical strength and so on. The main problem with these RAD tools is that they all, to a greater or lesser extent, suffer from the lack of sound management support for RAD approaches[4].

To respond to this problem, some efforts have been made to develop standards and methods for RAD. It appears that many vendors and consultants come up with their own interpretation and approach to RAD. In an attempt to reduce the confusion caused by various interpretations, a UK based consortium has been set up in early 1994. Their overall objective is to develop and to promote a world-wide industrial standard, which is named DSDM (Dynamic Systems Development Methods). The first version of DSDM manual was published early 1995, it consists a set of principles and a development process framework with corresponding management standards (Millington and Stapleton 1995). These principles on which DSDM based are:

- user involvement is imperative,
- DSDM teams must be empowered to make decisions,
- the focus is on frequent delivery of products,
- fitness for business purpose is the essential criterion for acceptance of deliverables,
- iterative and incremental development is necessary to converge on an accurate business solution,
- all changes during development are reversible,
- requirements are baselined at a high level (only high level requirements are required as a configuration management item),
- testing is integrated throughout the life-cycle, and
- a collaborative and co-operative approach between all stakeholders is essential.

The DSDM development process is divided into five phases: feasibility, business study, functional model iteration, design-and-build iteration, and implementation. Furthermore, different management and control aspects — from project managing, personnel to quality assurance and software procurement — have been developed and is still developing into the

[4]more details can be seen at: http://www.ovum.com/evaluate/.

DSDM.

The overall approach of DSDM can be summarised as: (a) targeting on wide industrial or commercial RAD communities; (b) great emphasis on end-user involvement; (c) highly iterative development process; (d) product-based focus and time-boxed managerial style. However, one fundamental question to be asked, and yet to be answered is: does this methodology apply to a wide spectrum of RAD applications where organisational infrastructure, culture, business areas, project sizes differ? If so, how? How much practical guidance? Otherwise what are the suggestions for improvement?

Thus far, apart from the reported trials within the consortium companies, there is little published on its use and results in wider RAD community.

2.3 Research Areas

Section 2.2 has given an extensive discussion about the characteristics of different types of prototyping as well as the tools and techniques associated with them. This section turns to the current research issues related to software prototyping. The research in the area of software prototyping can be roughly divided into two areas: one is related to supporting tools and environments for prototyping; the other is management and control of the process. The former has been largely described in the previous section, here it is included and briefly discussed from a research viewpoint, the centre of the discussion in this section is management and control related issues.

2.3.1 Prototyping Languages and Automatic Tool Support

There are a number of research institutes that have been working in this area over recent years, so a number of such languages and support environments have been developed (Mittermeir 1982; Hollinde 1984; Kruchten and Schonberg 1984; Leibrandt and Schnupp 1984; Van Hoeve and Engmenn 1984; Venken and Bruynooghe 1984; Vonk 1989; Martin

1991; Bischofberger and Pomberger 1992; Luqi 1992). More recent attempts have been made, with some successes, in automatically transforming specifications into executable prototypes (Heping and Zedan 1996; Borba and Meira 1997). The main challenge in designing a prototyping language is how to execute partial descriptions. This support can be provided by reusable code, transformation templates, and systems of default assumptions. However, to be useful the default assumptions must correspond to reasonable designs most of the time (Luqi 1992).

Some work has also been made in the areas of:

- **reuse and program generation.** Making reuse work requires systematic and long-range planning, investment, and co-ordinating by different organisations developing the same kinds of applications (Boehm 1988; Basili and Rombach 1991; Henhapl, Kaes et al. 1991; Lewis, Henry et al. 1991; Fernstrom 1992). So it seems that progress in this area depends more on organisational and social functions than that of technology (Redwine 1988; Basili and Rombach 1991).

- **domain models.** The main challenges in evolving domain models are how to automatically recognise two concepts that are variations on the same theme and how to generalise previous designs and code so that they cover all cases (Budde, Reinhard et al. 1984; Tracz, Coglianese et al. 1993). The objective is to use generic domain models and their associated generic designs and generic programs to instantiate new prototypes. However, prototyping is needed most in domains that are not well-understood, which contradicts the basis of domain models (McDermid 1991).

Apart from those languages and tools specially designed and developed for prototyping, many other general development tools and environments are already or are becoming available (Nelson and David 1984; Tavolato and Vincena 1984; Van Hoeve and Engmenn 1984; Endres and Weber 1991; Henhapl, Kaes et al. 1991; Pocock 1991; Steinbauer 1991; Sugiyama and Horowitz 1991; Fernstrom 1992; Lott 1993). Lott (Lott 1993) in his "Process and measurement support in SEEs" (SEEs — Software Engineering Environments) did a comparative study of current SEEs with

respect to process control and measurement-based management support. The results shows that these tools, more or less, provide measurement-based process support in terms of systematic process data collection. However, their ability to support management and control of the prototyping process in practice has yet to be substantiated.

2.3.2 Management and Control Related Issues

In recent years, there has been a wide recognition of the need for, and various ways to, improvement software process in general (Basili 1985; Blum 1985; Callender 1985; Dowson 1985; Chroust 1986; Dowson 1986; Goldberg, Green et al. 1986; Tully 1986; Wileden and Dowson 1986; Osterweil 1987; Humphrey 1988; Kellner and Hansen 1988; Deiters, Gruhn et al. 1989; Humphrey 1989; Humphrey and Kellner 1989; Kellner 1989; Kellner and Hansen 1989; Kitson and Humphrey 1989; Kellner 1990; Frailey, Bate et al. 1991; Kellner 1991; Finkelstein 1992; Lott 1994). Meanwhile problems with management and control of software prototyping practice are receiving particular attention (Shoval and Pliskin 1988; Mayhew 1989; Mayhew, Worsley et al. 1989; Pliskin and Shoval 1989; Mayhew 1990; Mayhew and Dearnley 1990; Smith 1991; Gutierrez 1993; Kautz 1993). The main concerns here are:

(1) exploring the various ways in which this may affect those people associated with systems development. e.g. developers, users and managers alike.

Lichter (Lichter, Schneider-Hufschmidt et al. 1993) concluded from an empirical study that application domain analysis is a necessary prerequisite for successful prototyping, and it is imperative that users as application domain experts participate throughout a prototyping project.

Pliskin examined the 'Responsibility sharing between sophisticated users and professionals in structured prototyping' (Pliskin and Shoval 1989). The case study constituted an intermediate scenario in terms of both system complexity and user sophistication and demonstrated the iterative nature of the approach and underlined the feedback loop. The collaboration reviewed can be presented in terms of two nested loops: a

loop of prototyping activities nested within a loop of analysis and design. In each loop, intensive collaboration can be expected, whereby the division of responsibilities is consistent with the separation between the external and internal architecture. Further research and real-world evidence on small and large systems is needed to verify the benefits and assess the limitations of the methodology. Are conflicts between users and professionals eliminated, or are they replaced by new conflicts? Are there development situations whereby the benefit-cost ratio is more favourable than in other situations? Furthermore, an investigation of the managerial implications of the approach is needed. For example, how can mixed teams of professionals and sophisticated users be promoted and encouraged? Pliskin suggested that such future research would have to be interdisciplinary.

(2) managing and controlling changes to the software during the prototyping process.

Some attempt has been made to address the configuration management issues that have emerged from prototyping (Hollinde 1984; Mayhew, Worsley et al. 1989). Mayhew's Change Classification Method (Mayhew, Worsley et al. 1989) was particularly proposed as a framework for controlling the typical changes that arise through prototyping, which seems to have been successfully used during the development of a commercial estimating system for the Royal Dockyards, UK. However, Mayhew felt that a) more research is needed to ascertain whether this approach is widely applicable; b) one situation that would certainly require some amendment to the framework is when the prototype system is actually given to the users to assess in their own working environment; c) a further problem that must be addressed to control prototyping fully is the need for impact analysis of propagated changes.

(3) controlling the iterations. e.g., when to cease the prototyping process.

An obvious criteria, as Mayhew suggested (Mayhew, Worsley et al. 1989), would be a metric based on the number and type of changes suggested during some set period, e.g. cease prototyping when less than two cosmetic changes have been suggested during the last half hour. She also

suggested that a 'statistics' database would be beneficial and make it easier to estimate the number of prototype iterations likely to be required for a particular project.

However, Tozer (Tozer 1987) pointed out that the fear of never-ending prototyping did not occur in real life. The reason for this, as the writer claimed, is that the commercial constrains will take over rapidly and " ... it is apparent that to both users and system builders that enough has been learnt from the first prototype to make more interactions unnecessary and economically unjustifiable". While this might be the case in reality, the criteria for 'enough has been learnt' was left undetermined.

(4) developing effective prototyping methods and standards.

There have been a number of proposed prototyping methods either general-purpose or domain specific (Aaram and Jarle 1984; Budde 1984; Budgen 1984; Kammersgaard 1984; Kraushaar and Shirland 1985; Lantz 1986; Luqi and M.Ketabchi 1988; Brice and Connel 1989; Chen and Wang 1989; Bischofberger and Pomberger 1992; Nelson and Bymes 1992; Buti 1996). More recently development in this area is the DSDM which is discussed in section 2.2.5.5 above. However, little evidence can be found about their industrial application. There is also an apparent lack of adequate methods and standards as well as guidelines for prototyping (Gordon and J 1991; Boyer 1995; Card 1995; Millington and Stapleton 1995; Robinson 1996).

(5) effectiveness of software prototyping.

While there are many claims of advantages (Boar 1984; Blum 1986; Bourke 1986; Gomaa 1986; Iggulden 1986; Lipp 1986; Harker 1988; Winek and Sriraman 1995; Cerpa and Verner 1996; Friel and Budgen 1997) on benefits of prototyping approaches, little work has appeared on quantitative measures and comparative studies of the effectiveness, such as reduced development effort, reduced testing time, more effective solutions, fewer maintenance requests, greater user satisfaction (Cavano and McCall 1978; DeMarco 1982; Fenton and Melton 1990; Fenton and Kitchenham 1991).

Some work has been done in evaluating the effectiveness of software prototyping (Leibrandt and Schnupp 1984; Stephens and Bate 1990; Gutierrez 1993; Friel and Budgen 1997). Boehm, in particular, did a controlled experiment in comparing prototyping and specifying approaches (Boehm 1984). In this experiment, seven software teams developed a small-size (2000-4000 SLOC) software product. Four teams used the specifying and three teams used the prototyping approach. The main results were:

- prototyping yielded products with roughly equivalent performance, but with about 40 percent less code and 45 percent less effort;

- the prototyped products rated somewhat lower on functionality and robustness, but higher on ease of use and ease of learning;

- specifying produced more coherent designs and software that was easier to integrate;

The main limitations were that the experiment was sensitive to exceptional individuals' performance, experimental boundary conditions and the size and nature of the application.

(6) organisational issues.

The effects of organisation issues, such as infrastructure, culture, managerial attitudes, customer and end-user education etc., upon software prototyping have been widely reported (Basili and Rombach 1987; Mayhew and Dearnley 1990; Baskerville and Smithson 1995; Hardgrave 1995; Yap 1995). For example, Gutierrez (Gutierrez 1993) highlighted the importance of organisational commitment and practice of self-evaluation.

(7) cost estimation and quality assessment of prototyping project.

While the common belief is that the results of prototyping lead to a quality product and are more cost-effective in terms of users satisfaction and time saving, some (Hardgrave 1995) argued that, for some complex and large systems, things could be the reverse. For example, cost may increase when the prototypes are constantly being throw away (Boar 1984), or reduced product quality because of being failing to follow the quality assurance standard used in conventional projects (McDermid 1991).

Lastly, Mathiassen (Mathiassen, Stage et al. 1992) made a good point in his paper entitled 'the principle of limited reduction. The point he argued was the fact that complexity and uncertainty cannot be reduced in an unlimited way; only that we can achieve some sort of equilibrium. If we see specifying as reducing complexity and prototyping as reducing uncertainty, then the rule certainly applies. The implication is that given limited resource one has to reach some balance in reducing both complexity and uncertainty: one reduction would cause an increase in another.

2.4 Conclusions

Based on the discussions 2.1 to 2.3 above, the main conclusions are drawn as follows:

(1) lack of consensus about software prototyping.

As discussed in sections 2.1 and 2.2 above, there exists wide differences on what software prototyping is, and where and how it should be applied. A working definition is given for this study, which includes two different approaches: both a techniques for structured development and an alternative systems development paradigm.

(2) there are many tools and environments for software prototyping.

Section 2.2 above clearly demonstrates that there are a large number of languages, tools, environments that are for or can be used for all kinds of

prototyping.

(3) lack of empirical work in management and control aspects of software prototyping practice.

The discussion above (section2.3) has demonstrated that the problematic of management and control issues are often rooted from the lack of understanding of the process (also refer to (Curtis, Krasner et al. 1988; Curtis 1990; Fenton 1993; Potts 1993; Fenton and Pfleeger 1994)). Research work along this line appears to be specially needed in providing better understanding of management and control of prototyping.

Chapter 3 Methodology

SYNOPSIS This chapter explains and describes the methods for work. Firstly, it introduces and then argues about the suitability of case study as the overall research strategy for this investigation (3.1 and 3.2). The three most commonly used strategies for empirical studies are closely examined, and case study is chosen as the most appropriate research strategy for the study because it is a particularly good vehicle for seeking 'why' answers to the problems of concern; furthermore, multiple cases is adopted over single case study considering the trade off between generalisability and level of detail that each method offers. Secondly, this chapter gives a flavour of our overall structure of the research by describing and explaining the particular methods to use at each stage of the investigation, the relationships among the methods and their rationale.

3.1 Introduction

Having discussed the current research and practice in software prototyping, conclusion has been made that the management and control aspects of software prototyping are more problematic and less researched than prototyping tools and environments. The overall objective of this investigation is therefore to gain a better understanding of the management and control of the process.

To achieve such an overall objective effectively, a well defined methodology for work is needed. As the research is basically about seeking 'how and why' answers of the problems of concern in the field by empirical study, the natural choice for the research strategy is case studies (Whyte 1984; Yin 1984; Walker 1985; Humphrey 1988; Walsham 1995), at same time, survey is also used as an additional strategy at the early stage of the investigation (Rossi, Wright et al. 1983; Hakim 1987; Miller 1991).

The following section (3.2) is an overview about case study: its use, a working definition, its strength and weakness, a comparison between case study, survey and experiment, and its use as the primary research strategy. Section 3.3 expounds the framework of the research, the methods to use and the rationale. Finally, the key points are summarised in section 3.4.

3.2 Case Study as Research Strategy

3.2.1 What is Case Study?

3.2.1.1 Use and Working Definition

Case study has long been used as one of several ways of doing social science research. However, in recent year, it has been gradually gaining ground in IS (information Systems) research (Potts 1993; Walsham 1995), such examples are evident (Pliskin and Shoval 1989; Carey 1990; Tate and Verner 1990; Martin 1991; Lichter, Schneider-Hufschmidt et al. 1993; Pliskin, Romm et al. 1993).

Although it has been variously defined according to its particular use (Whyte 1984; Miller 1991; Hammersley 1992), the following (Yin 1984) is used as our working definition.

A case study is an empirical inquiry that:

- investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident;
- copes with the technically distinctive situation in which there will be many more variables of interest than data points
- relies on multiple sources of evidence, with data needing to converge in a triangulating fashion;
- benefits from the prior development of theoretical propositions to guide data collection and analysis.

3.2.1.2 Strengths and Weaknesses

While the strengths, generally speaking, are contained in the above definition, the major weakness has been its allowance of equivocal evidence or biased views to influence the direction of the findings and conclusions, and it results in massive and often unreadable documents. Probably, the most common concern is its generalisation, however, the answer to this is that "case studies, like experiments, are generalisable to theoretical propositions and not to populations or universes." (Yin 1984).

Its advantages and disadvantages can be demonstrated by comparing the case study with survey and experiment. The discussion here is based on Martyn Hammersley's view in his "what's wrong with ethnography" (Hammersley 1992), in which he explains that the distinctiveness of experiment is that the researcher creates the cases to be studied through the manipulation of the research situation, thereby controlling theoretical and at least some relevant extraneous variables; the distinctiveness of survey is that they involve the simultaneous selection for study of a relatively large number of naturally occurring cases; and case study combines some features of these other two strategies which involves the investigation of a relatively small number of naturally occurring cases.

The comparison is best illustrated by the following diagrams[5] (Hammersley 1992):

[5] from page 186 and page 193 of (Hammersley 1992).

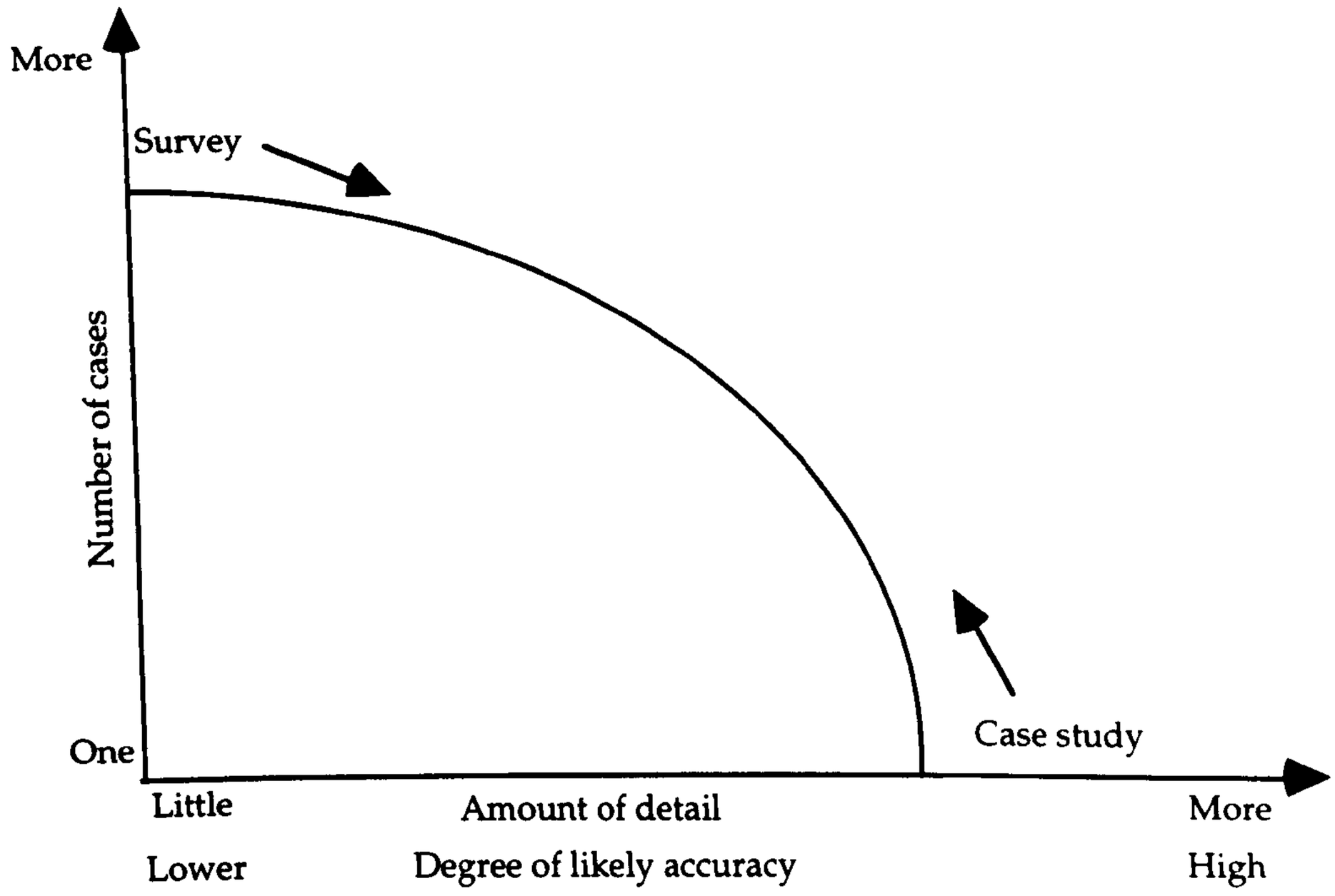


Figure 3.1 Relationship Between Survey and Case Study

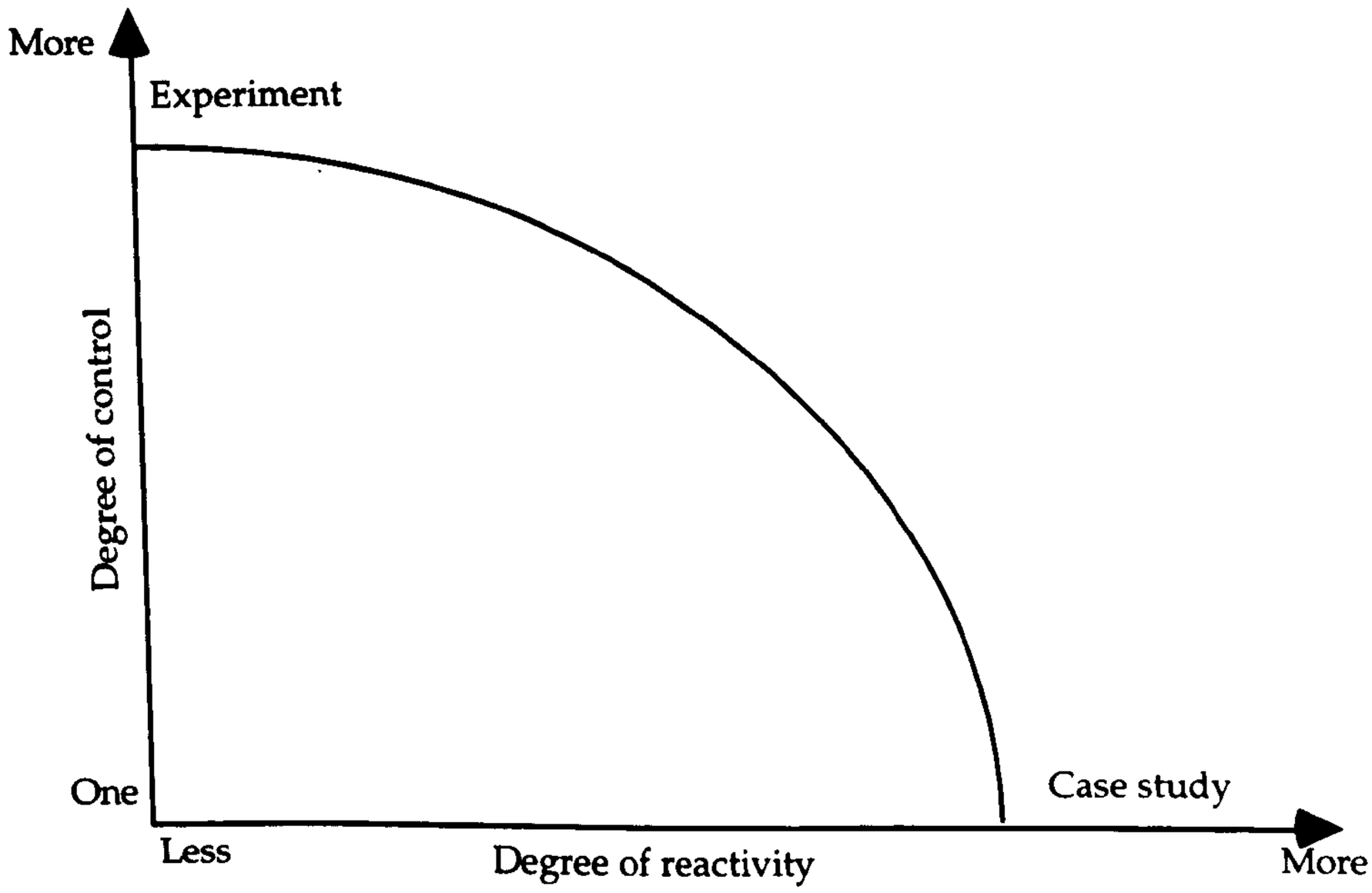


Figure 3.2 Relationship Between Experiment and Case Study

Clearly, figure 3.1 shows that, given certain resources, the distinction between case study and survey involves a trade-off between the likely generalisability of the information obtained and the detail and likely

accuracy of data about particular cases. Figure 3.2 shows that the trade-off is between the level of control of variables and the level of likely reactivity. It also should be clear that, on one hand, experimentation may increase the chances of coming to sound conclusions by creating the cases we need and varying theoretical and extraneous variables fairly easily, which case study lacks; on the other hand, experimentation cannot provide us with what case study offers, that is, the information about the naturally occurring situations in which we are interested.

Although empirical methods can be broadly grouped in two contrasting camps — case study vs. experiment — as discussed above, there exist a number of variants, most notably action research. A detailed comparison of these methods can be seen in the following references (Galliers and Land 1987; Galliers and King 1994). Action research differs from conventional case study in several ways: firstly, it requires the researcher to actively take part in the process being studied; secondly, it does not pre-define the topic of study; and thirdly it intends to interfere with the process by attempting to solve the problem at the site (Cavaye 1996). Therefore, fully or partially adopting an action research strategy would, on one hand, provide some insights to this study from a different perspective, on the other hand, weaken the power to focus on the problem area and to observe the phenomena without interfering. Another problem is one of resources. It is recommended that action research has a minimum duration of one year (Cavaye 1996). Given that we had ten processes to study, this made action research an inappropriate choice for this study.

3.2.2 Why Use Case Study as Our Primary Research Strategy?

Considering the gains and losses in the light of the particular goals and circumstances of the research including the resources available, casy study is selected as the overall research stragey.

Firstly, it is a good means to achieve the overall objective. The proposed study, considering its *interpretive* nature and the practical concern, is well suited for case studies (section 3.2.1). The interpretive nature is meant to learn about the *phenomenon in its context*, or the natural occurring

situations in which we are interested in, and to reason about the relationship between the phenomenon and its context; and the practical concern is that the result of the study is to be used as not only the inputs of theory building but also the *practical* guidance for practitioners.

Secondly, case study enables an adequate level of validity for the study. Referring to the comparison made above in section 3.2.1, it is clear that, while surveys find it difficult to handle 'how and why' kind of information, experimentation requires tight control over the situations we are interested in, which, to this study, is neither intended nor possible. By contrast, case study is best suited for seeking 'how' and 'why' answers and studying naturally occurring situations.

3.2.3 Single Cases vs. Multi-Cases — Which to Choose?

Here there is a choice between a single case or multiple cases. Although Yin (Yin 1984) considers that strategically there is no fundamental difference between the single case study and multiple cases studies, they have distinct advantages and disadvantages, "... multiple cases are often considered more compelling, and the overall study is therefore regarded as being more robust.", whereas the single case study is better for "unusual or rare case, the critical case, and the revelatory case" (Yin 1984). Similarly, as discussed in section 3.2.1.2 above, the difference may be seen as the trade-off between the generalisability and the level of details and accuracy of information a under given resource (Hammersley 1992). Having considered these points in relation to the overall objective as well as the resource limitations, a decision is made to conduct multiple case studies with a small number of cases depending on the number of companies available.

At the same time, a survey is also used during the early stage of the investigation to get wider information about the current practice. For, as discussed in section 3.2.1, each strategy has its own strengths and weaknesses and there is no clear cut between boundaries, and they may employed under circumstances where most appropriate.

The above is only a brief account that aims to form a overall picture of our

research strategy and how it is defined and implemented will be the topics of the whole thesis. The following is a brief description and explanation of the framework which is aimed at carrying out such a strategy effectively.

3.3 Framework Defined

Clearly having a framework defined at the outset is important for any research work. This is particularly true for the proposed study because a multi-case study will involve a large amount descriptive information as well as multiple sources of evidence. Figure 3.3 depicts the framework of the empirical study: the stages it consists of, and the relationships among them.

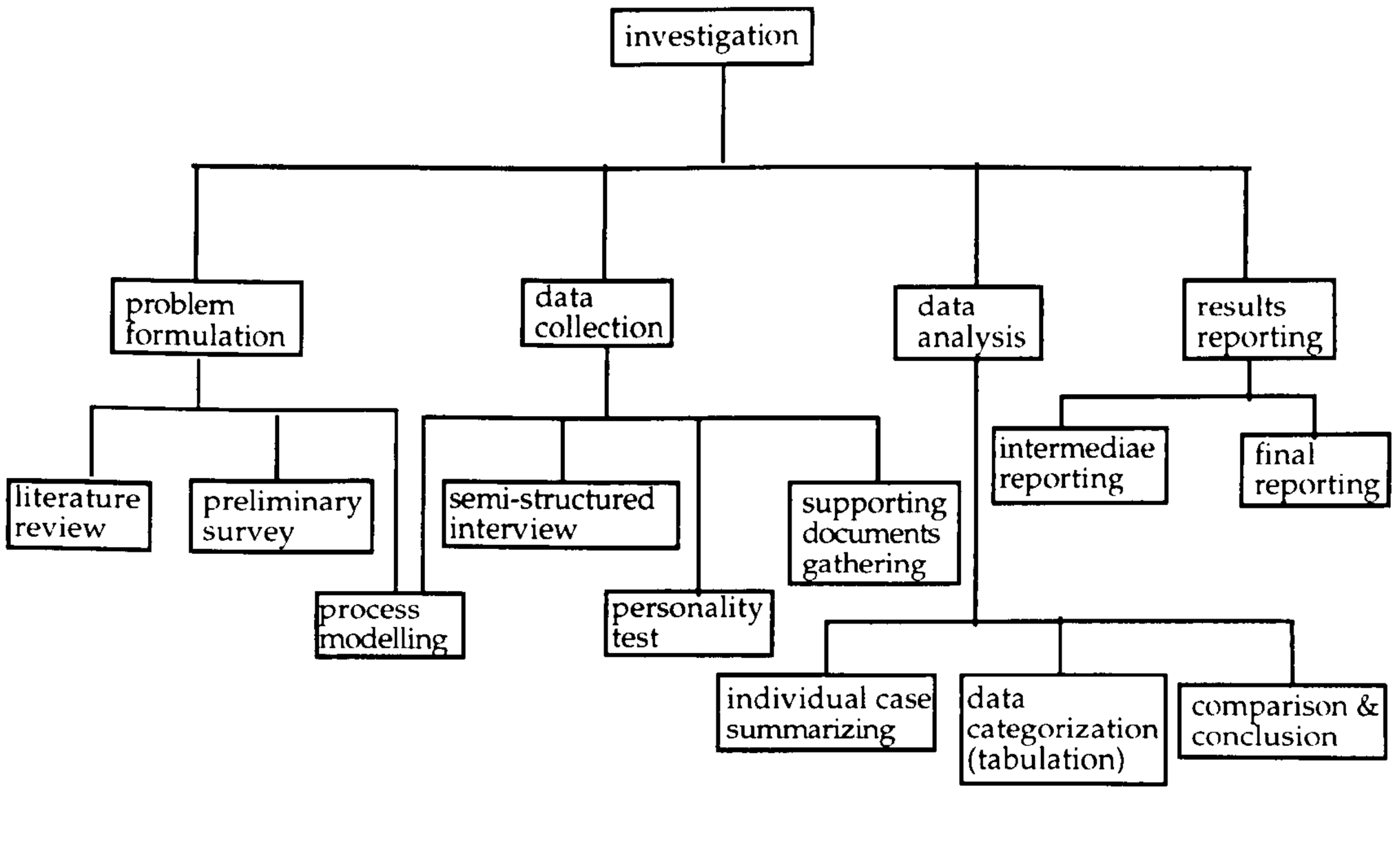


Figure 3.3 The Framework and Methods for the Investigation

As the above figure 3.3 indicates, the investigation is divided into four basic stages: problem formulation, data collection, data analysis, and results reporting. They, together with the activities and/or methods involved in each stage, are explained in the following sections.

3.3.1 Problem Formulation

The problem formulation involves the literature review, the preliminary survey and the field modelling. Each later step can be seen as both a independent source of informing problems, and a further clarifying and focusing of the problems identified from earlier steps, which is indicated in following figure (figure 3.4).

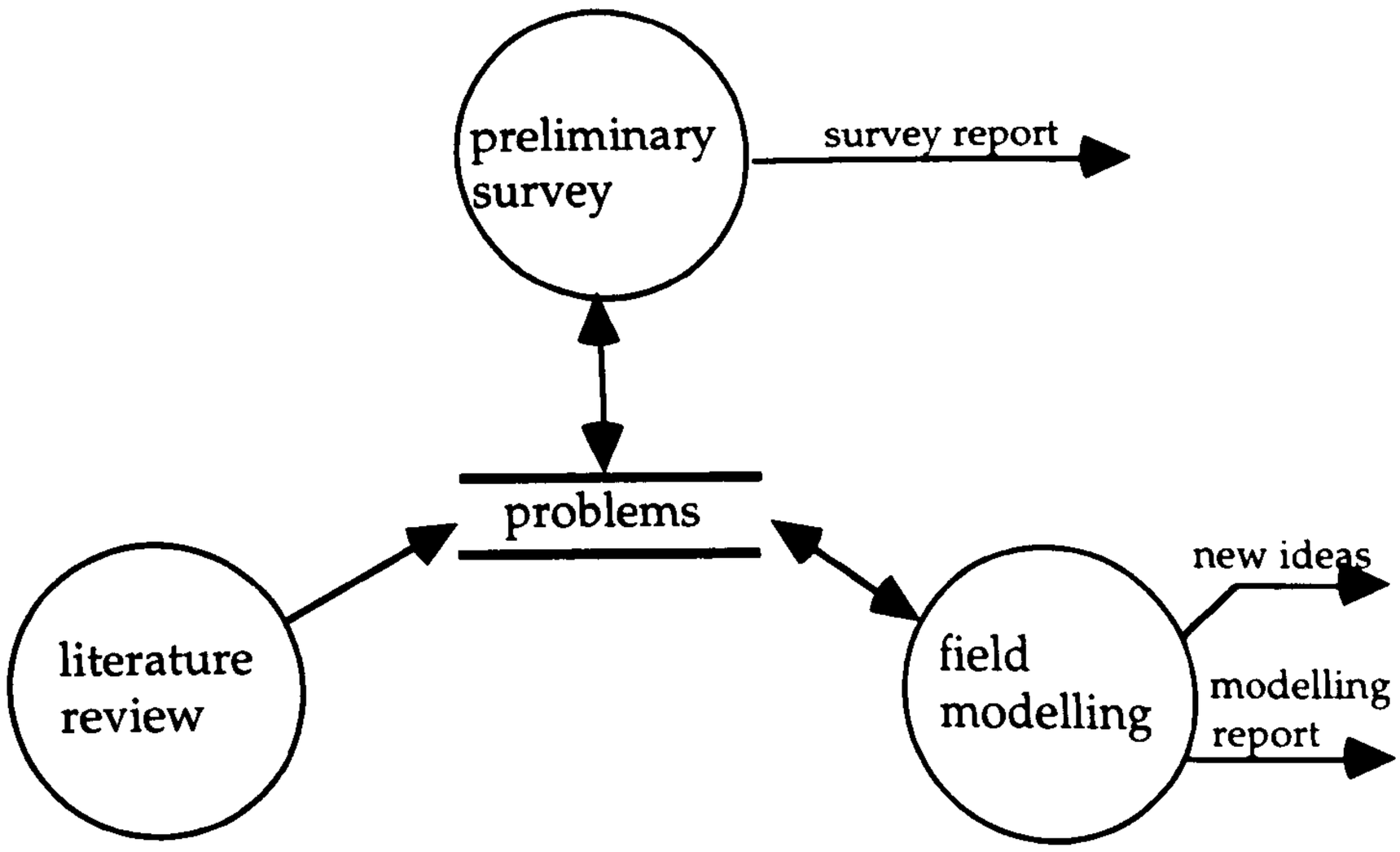


Figure 3.4 The Three Steps of Problem Formulation

Here, 'problems' is meant to be the sum of problems reported from the literature review, and problems updated and/or found by the survey or the modelling; 'survey report' is the report to be sent to survey participants; 'modelling report' is the report to be sent to modelling participants; and 'new ideas' is meant to be the new ideas generated for further investigation. The following further explains the reasons behind the structure for the problem formulation.

3.3.1.1 Literature Review

The literature review (chapter 2) is the first step of the problem formulation, which is to identify the area of study as well as its associated problems. It should be clear that our research strategy and framework is

defined after and indeed based on the findings of the literature review as well as the initiative of emphasising the practical usefulness of the research.

3.3.1.2 Preliminary Survey

The preliminary survey is the next step after the literature review. The purpose of such a survey is three-fold: a) to gather some general opinion and particular concerns from practitioners in the field; b) to assure and/or adjust the objectives sought; c) to build industrial links for further collaboration. (For the survey and the results analysis, see chapter 4).

3.3.1.3 Field Modelling (I) — for a Better Understanding of the Process

As shown in Figure 3.3 (the framework of the study), the field modelling is an activity that cross the first and the second stage — problem formulation and data collection — of this empirical study. It serves two purposes for this study: (a) for a better understanding of software prototyping process in practice; (b) as a framework for data collection .

Process modelling is a powerful technology that has been used for modelling business processes since 1960's, and it has stimulated a great deal of interest in evolving into software process modelling in recent years (Ould and Roberts 1986; Hansen and Kellner 1988; Ould and Roberts 1988; Kellner and Hansen 1989; Kellner 1990; Soong and Osterweil 1991; Yeh 1991; Abeysinghe and Phalp 1997; Penedo and Shu 1997), which will be further discussed in chapter 5 — Field Modelling. Although it has been used for various purposes such as process improvement, process re-engineering, process automation, it is used for this study to serve two purposes: first, to better understand of the process, especially, from the human perspective; and second, to use the actual process models as a framework for data collection and data analysis (see 3.3.2.1).

To satisfy these purposes, the RADs (Role Activity Diagrams) is chosen as a modelling tool (Holt, Ramsey et al. 1983; Ould and Roberts 1986; Abeysinghe and Phalp 1997) (more about RADs see Chapter 5 and Appendix C). The main advantage is that they give a straight-forward

view of a process in terms of process roles, role activities, and role interactions. The benefits of using such a notations are therefore not only to increase the efficiency of communication between the reporter and the interviewee, but also to make the data collection easier by asking questions directly referring to the processes modelled and make the data analysis more compelling by providing more comparative data.

3.3.2 Data Collection

The main purpose of this stage is to collect as much relevant data as possible relating to the more clarified and focused problems from the previous stages. The data collection consists of four sub-processes and each of them involve collecting a different type of data. This is further explained by looking at the methods to use and the rationale behind them.

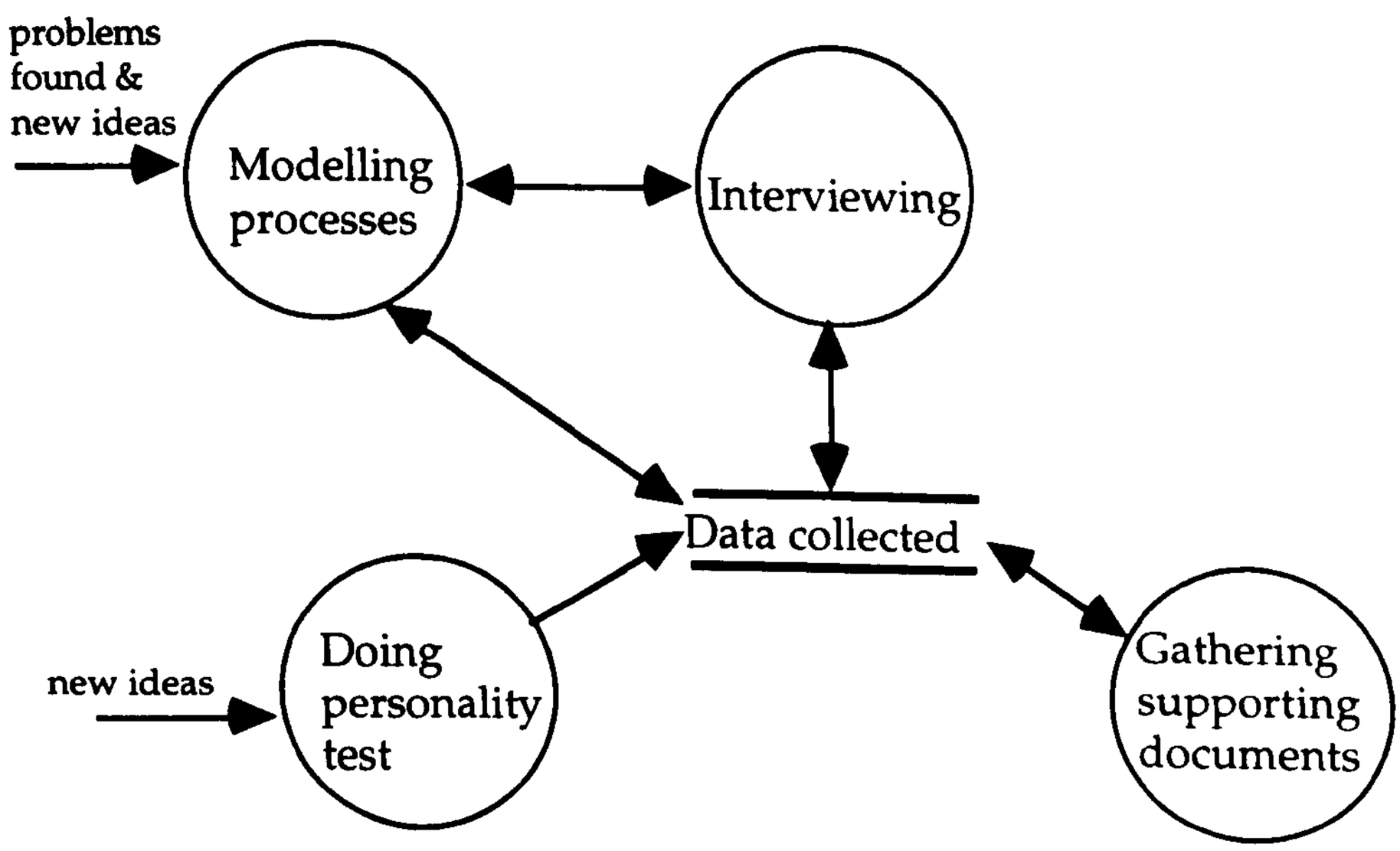


Figure 3.5 Data Collection

3.3.2.1 Field Modelling (II) — as a Framework for Data Collection

Section 3.3.1.3 above has explained one of the two functions for the field modelling — problem formulation, this section describes its other function — as a framework for data collection. This is illustrated in figures 3.3, 3.4

and 3.5 above. The problems found and new ideas from previous stages are the basis for the field modelling, the results of modelling provide inputs for further interviews, and the other data collection activities include doing the personality test and gathering the supporting documents. Clearly, the important function of the field modelling here (Field modelling (I) and (II)) is that it provides the transition from the problem formulation to the data collection.

3.3.2.2 Semi-Structured Interviews

Semi-structured interviews are used because, firstly, the interviews need to be well structured for later meaningful comparison, and secondly to be flexible and adaptable to different organisation or individual accordingly. The importance of the interviews at this stage can not be over emphasised, for they are the crucial part of the case studies and they provide the richest and the most substantial data or evidence for the analysis (The interview form template and interview question guidelines can be seen in the Appendix E(a) and E(c)).

3.3.2.3 Personality Test

As figure 3.5 indicates, the personality test is treated as one source of evidence, which is used to uncover the relationship between participants' personalities and the roles they play, and consequently the possible implications on management and control of the process.

3.3.2.4 Supporting Documents

The supporting documents here could be anything from development methodology to project documents such as proposals, sign-offs, high level designs. The intention here is to provide some evidence for a) updating process models; b) uncovering discrepancies between documentation and actual practice.

3.3.3 Data Analysis

Data analysis is at the heart of the study, the task is to fairly and clearly present the data gathered and to make valid comparison and analysis. The following figure (figure 3.6) shows the methods to use and their inter-relationship, which will be further explained in sections (3.3.3.1 - 3.3.3.3) below.

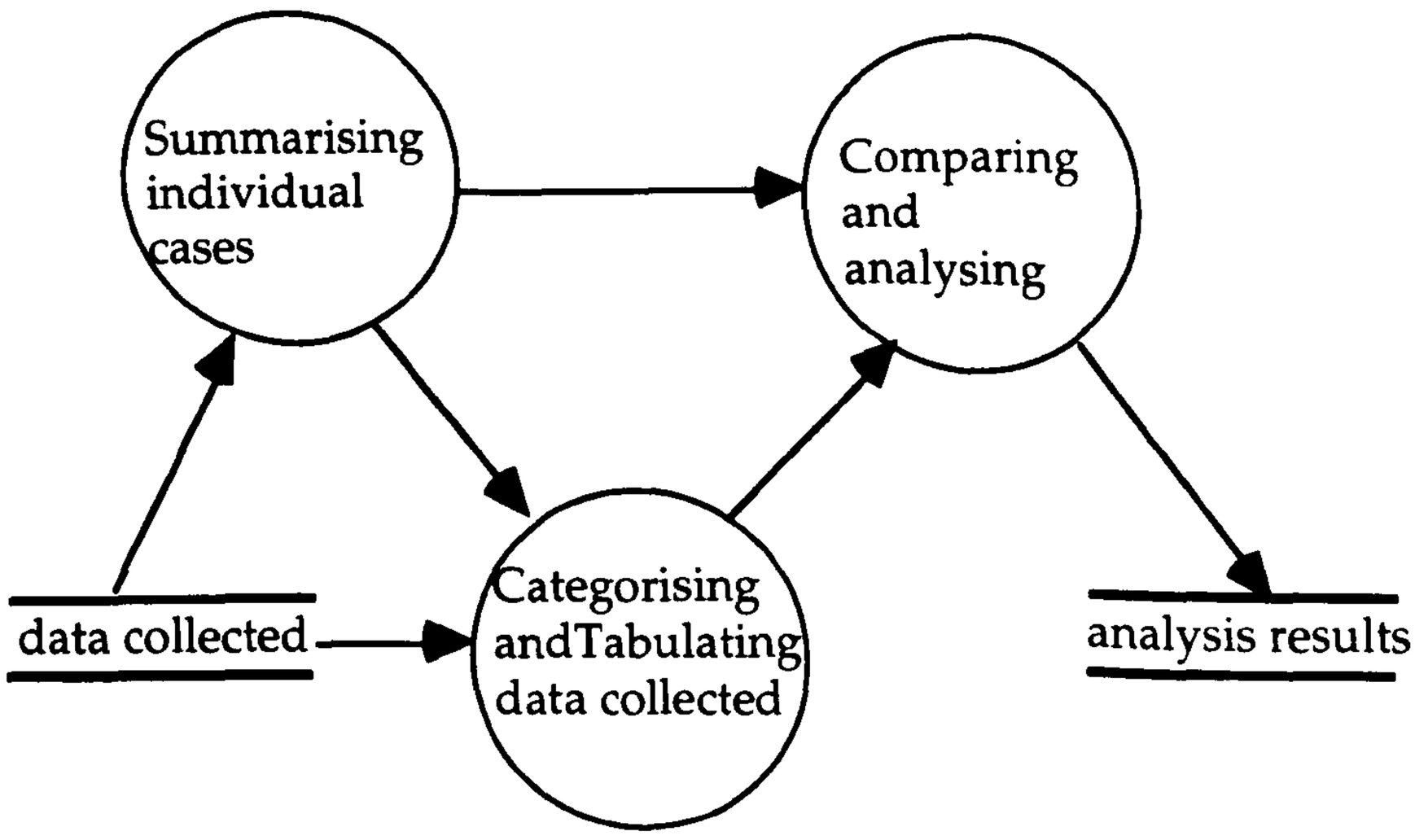


Figure 3.6 Data Analysis

3.3.3.1 Individual Case Summary

Each individual case and/or interview is first summarised. Each summary takes all the relevant data gathered for that case. The purpose is to make provision for categorising and subsequently comparing and analysing the data.

3.3.3.2 Data Categorisation

Data categorisation is a step further towards to final analysis. The task is to tabulate all the comparable data of all sources from each case into different tables.

3.3.3.3 Comparison and Analysis

Comparison and analysis will be performed once the data are gathered and categorised, and conclusions will then be drawn upon. At the same time, other data that could not be categorised will also be accounted for.

3.3.4 Results Reporting

Figure 3.7 illustrates the reporting structure:

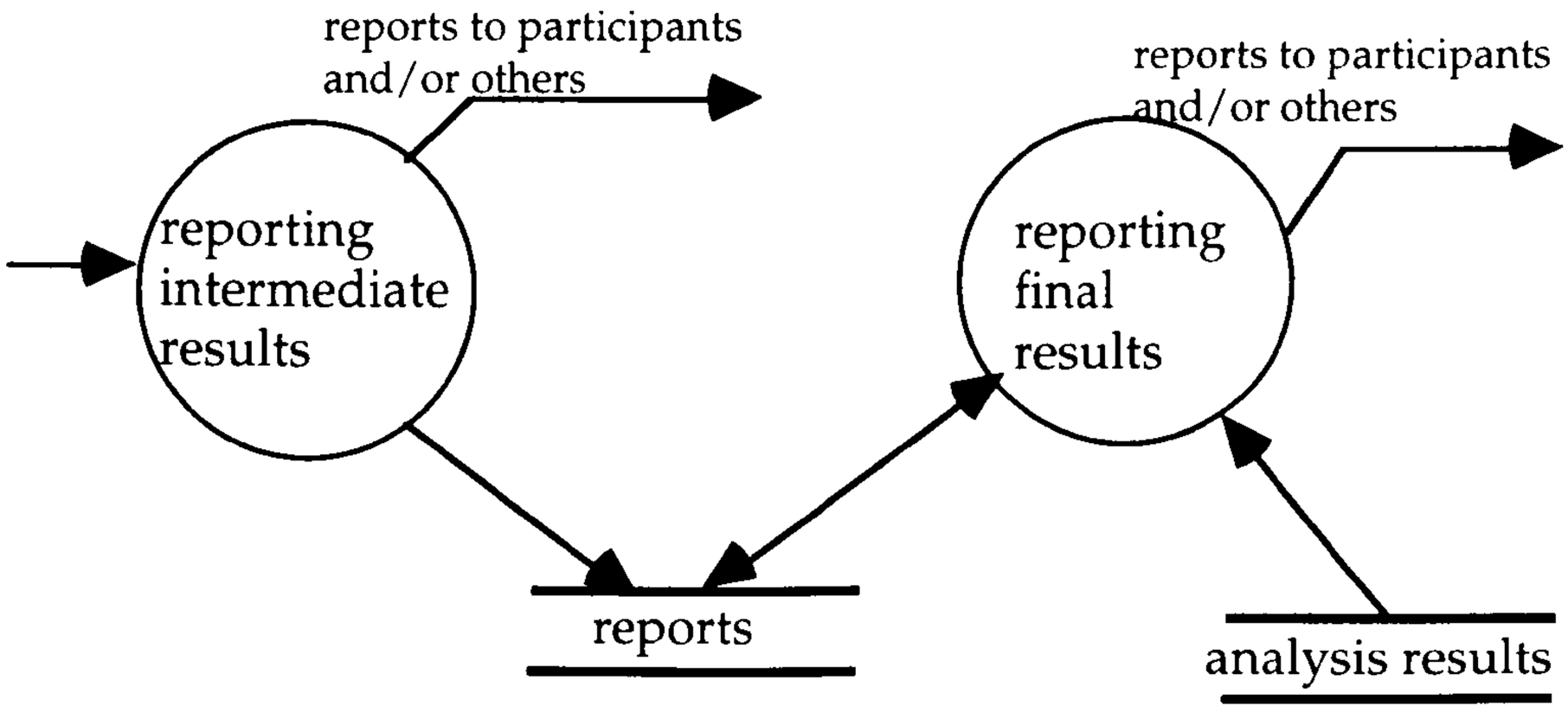


Figure 3.7 Results Reporting

3.3.4.1 Intermediate Reporting

Intermediate reports are intended after the initial and intermediate stages of the investigation, which include the reports on literature review, preliminary industrial survey, field modelling, and personality test. They function as intermediate reviews and are distributed to the industrial participants as well as the colleagues for feedback.

3.4.2 Final Reporting

The final reports are intended at the end of investigation, and they may take form of the thesis chapters, technical reports to industrial participants, conference and/or journal papers.

3.4 Conclusions

Firstly, by contrasting the three most commonly used strategies for empirical studies, case study is chosen as the most appropriate strategy to adopt for the purpose of this study, and also multiple cases is chosen over single case study for acquiring more compelling results, although only a small number of cases are intended considering the trade off between generalisability and level of details. Secondly, a flavour of the overall structure of the research has been given by describing and explaining the methods to use, the relationships among the methods and their rationale. Finally, emphasis is given to those crucial stages or activities to this investigation: one is the two-staged field modelling both as vehicle for process understanding, especially from human perspective, and as the framework for data collection; another is the semi-structured interviews which is to be the major source of evidence.

Chapter 4 Preliminary Survey

SYNOPSIS The preliminary survey is the first step of the investigation. This survey aims at obtaining a broad view of industrial practitioners about their software prototyping practice in general and revealing their real concerns about the management and control of the process in particular. It also functions as a readjustment and reassurance of the initial objectives arrived from the literature review as well as a bridge to next stage. The main finding of the survey, which largely confirms the literature review, is that management and control of the process is the overwhelming concern.

4.1 Introduction

The questionnaire survey was to serve two basic purposes. First, it was to find out more about software prototyping practice from the industrial practitioners' viewpoints about how it was used and what were the pressing problems to be solved. Second, it was to stimulate and explore further opportunities for industrial collaboration. Moreover, by conducting such a survey, not only would the research directions from the literature review be re-assured or adjusted before embarking on further investigations, but one would also ensure that this empirical study deals with important issues of practical concern.

4.2 Method for Work

As discussed in the previous chapter, a questionnaire survey was chosen as the method for work at this stage. The main reasons for this were that it is a relatively easy and quick way to gather broad and general opinions about the process, and it would be more meaningful for comparison and easier for analysis.

To achieve the objectives given above, a questionnaire was designed (see Appendix A) to gather, from practitioners, information on the following issues :

- how long software prototyping has been used within their organisation?
- where had it been used?
- what were the benefits gained?
- what were the problems encountered?
- what were the most pressing issues that needed to be solved?
- what were the suggestions for improvement?

Additionally, in order to establish and maintain participants' interests in the investigation, a question was asked about their willingness for further contact, and the incentive was that they would receive reports addressing their concerns.

To effectively achieve the overall objective with high return rate within a limited time, the questionnaire was designed to be short, concise and easy to answer. Several psychologists were consulted during the questionnaire design, and a pilot version was used among a number of computing staff in the department prior to mailing.

4.3 Survey Results

From the 80 questionnaires sent out, between the end of December 1993 and the beginning of February 1994, 26 replies were received .

The companies, to which the questionnaire were to be distributed, were selected from the departmental (Computing Department of Bournemouth University) industrial contacts database, and based upon the following principles:

- companies which were likely to be practising software prototyping;
- companies which were convenient for visiting;
- companies which were likely to reply.

In addition, time and manpower availability were also factors for the number of companies selected for the survey.

The results were represented in the following charts (charts 4.1 — 4.4). In each chart the horizontal axis represents categories (e.g. domains, benefits and so on) and the vertical axis represents the positive response rate in percentages.

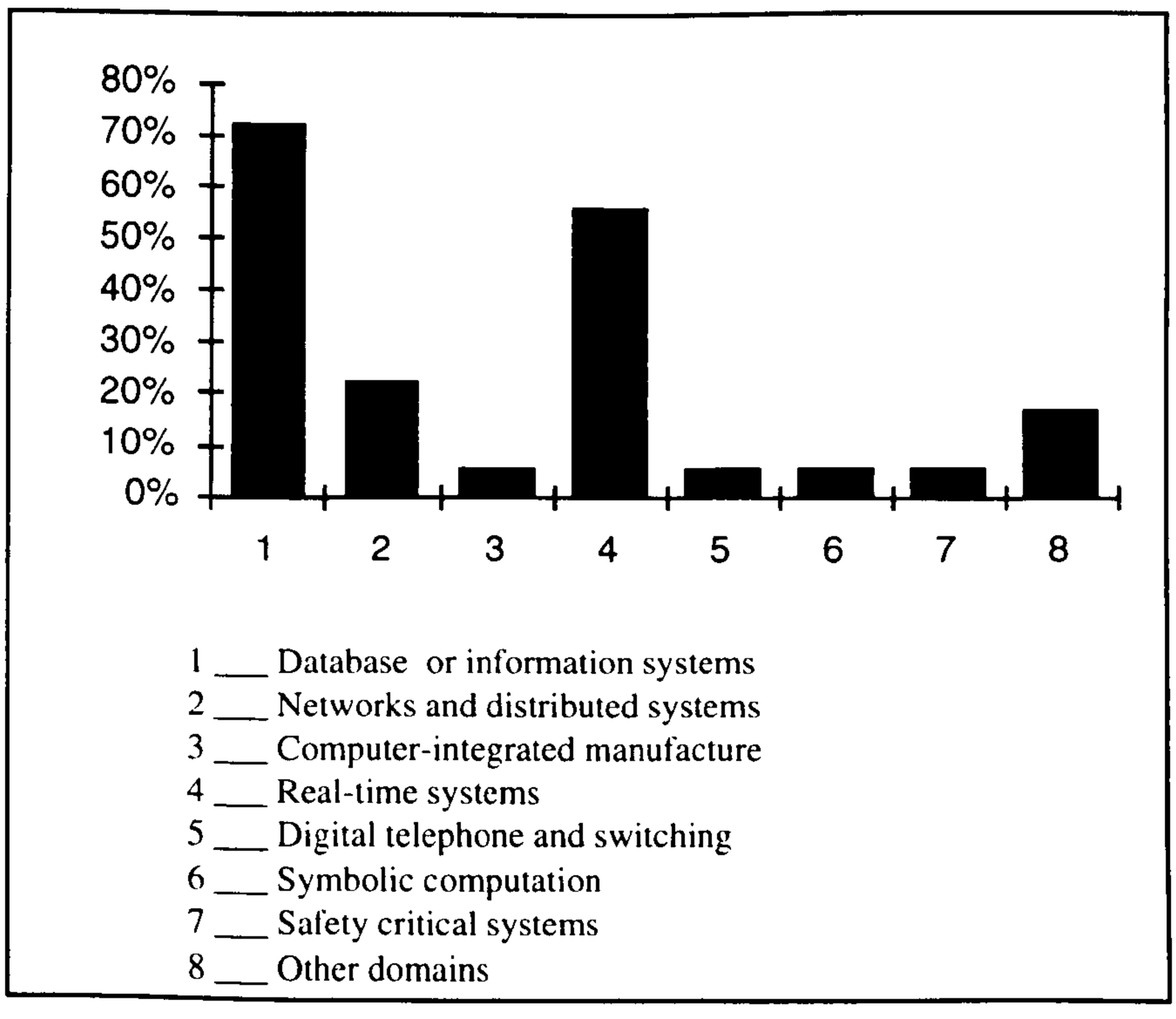


Chart 4.1 Application Domain Distribution

Chart 4.1 shows the distribution of where prototyping has been used. Here domains may overlap, e.g. some companies practise prototyping in

more than one domain. It is obvious that the database or information systems (left most column) is the most significant application area for prototyping. Another outstanding area appears to be real-time systems. A number of additional application areas mentioned on the replies include multimedia tools, graphical user interface and automatic testing equipment.

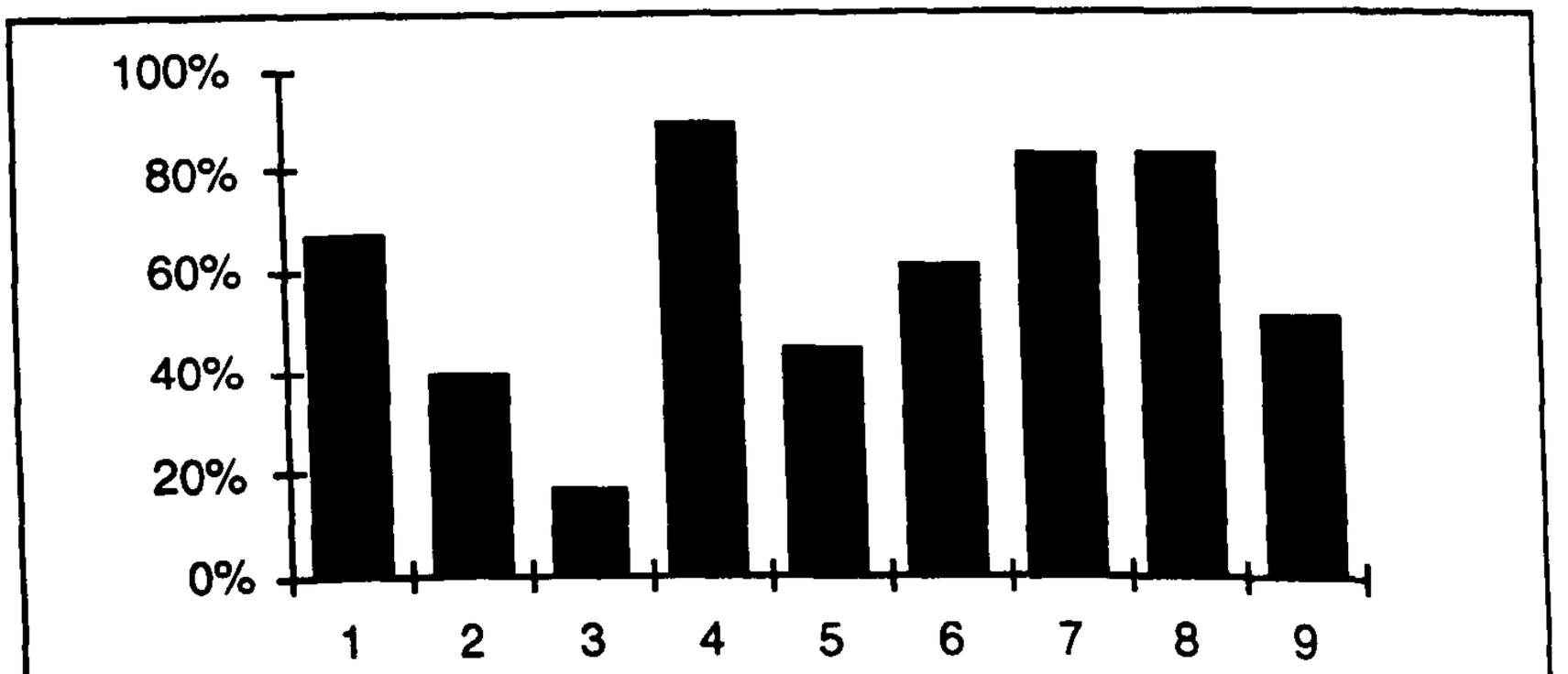
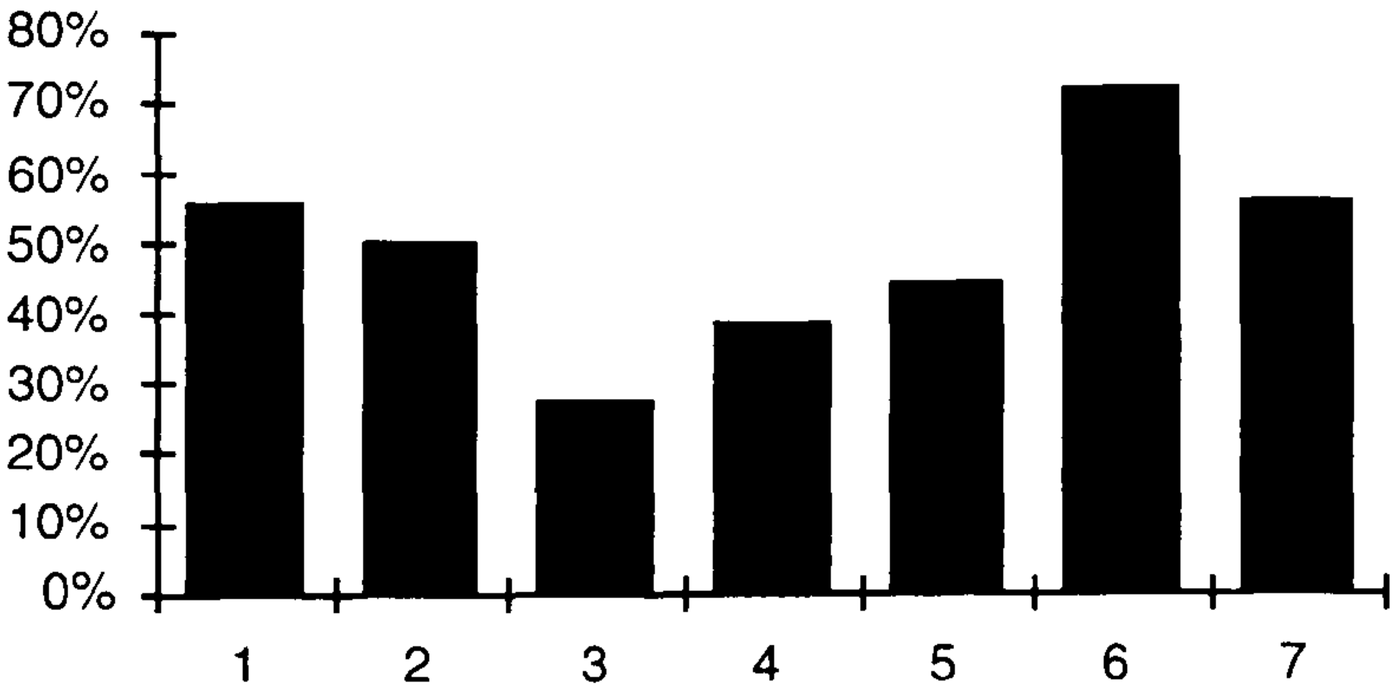


Chart 4.2 Opinions on Benefits Gained

Chart 4.2 shows the distribution of the respondents' opinions on the benefits of software prototyping. Almost all (nearly 100% shown in chart 4.2) agreed on the following gains: greater user satisfaction, better human computer interface and earlier user validation. Few agreed with enhanced maintainability (less than 20%). Other benefits reported are clearer visualisation, better demonstration of functionality and better use as a training medium.



1 ___ Lack of effective methods or guidance
2 ___ Inadequate tools and / or environments
3 ___ Lack of management support
4 ___ Prototyping process is hard to control
5 ___ Problems of communication and cooperation between users, prototypers, developers and managers
6 ___ Controlling product quality
7 ___ Other problems

Chart 4.3 Problems Encountered and Frequency

Chart 4.3 shows the distribution of the respondents' opinions on the problems encountered with using software prototyping. Most responses on problems listed on the questionnaire are fairly evenly distributed, apart from the question on controlling product quality (column 6). Other common problems are: keeping documentation in line with fast changing prototypes, "throwaway" prototypes becoming end-products and cost estimation for prototyping projects.

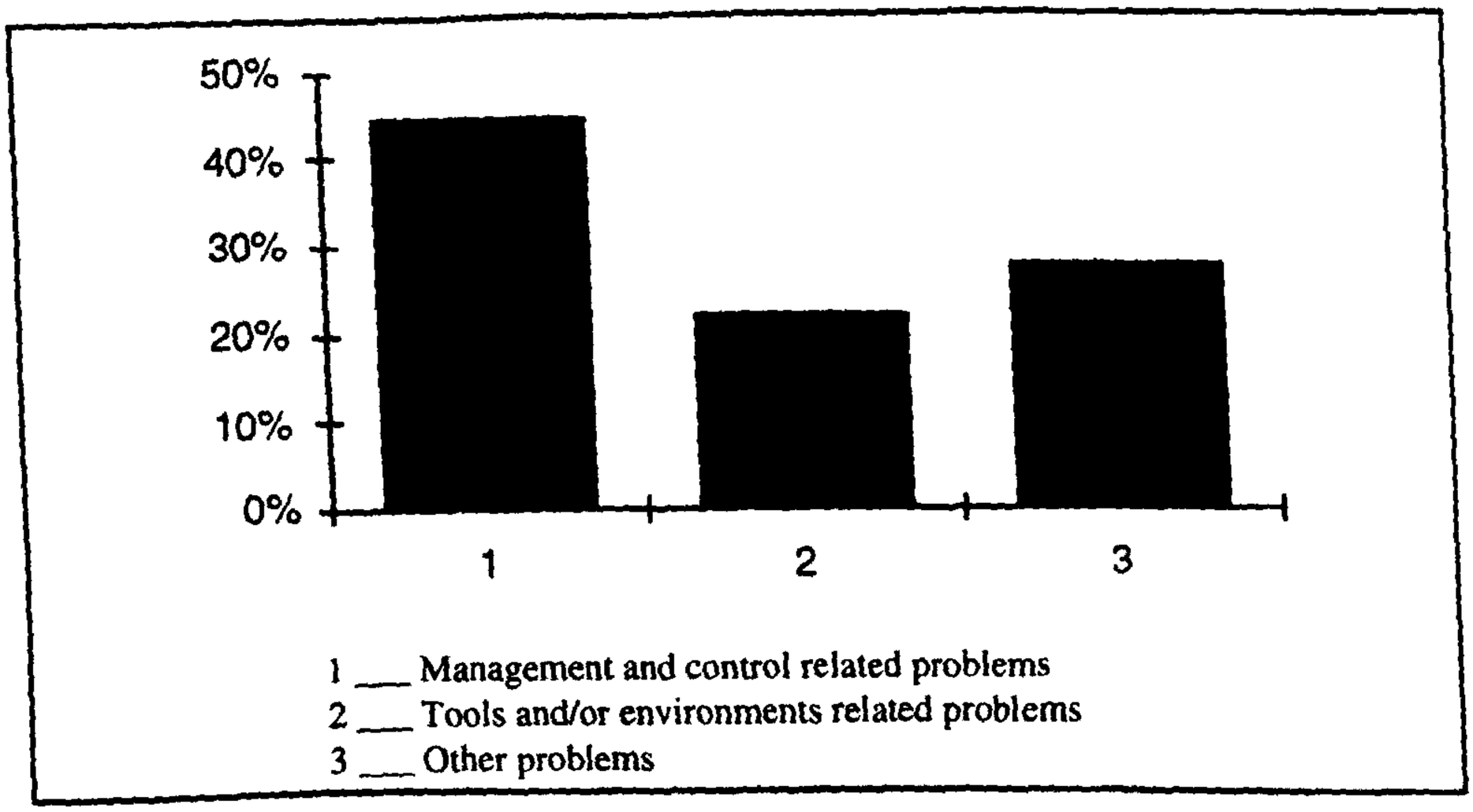


Chart 4.4 Problems Distribution

The distribution of the respondents' opinions on the most pressing problems to be solved are represented in chart 4.4 About half the respondents indicated management and control related issues were their first priority. Tool related problems, cost estimation and users' over-expectations were also of concern.

4.4 Summary

Having compared the questionnaire results received from 26 organisations, it is clear that these results largely confirm the literature findings on software prototyping in terms of its application areas, benefits claimed and problems encountered. The result indicates that the dominant application area is information systems, and the most problematic area is the management and control of the process (particularly, product quality) compared with tools and environments. This reflects the recent concerns of the research community and therefore re-affirms the value of research into this area.

In addition, as a result of this survey, 8 out of the 26 companies expressed their interest in further co-operation. Undoubtedly, this would enable the researcher to further establish close links with industry and therefore substantiate the practical value of this study.

Finally, it should be pointed out that the main limitations of the results of this survey were the relatively small sample size, and the relative arbitrary selection criteria of the target companies. Nevertheless, given the time and resources available, the survey has fulfilled its basic objectives, i.e. it has confirmed that management and control is a particularly problematic area, and it has established a number of industrial links for further investigations.

Chapter 5 Field Modelling

SYNOPSIS Following the preliminary survey, the field modelling was carried out within eight organisations. The output was ten RADs (Role Activity Diagrams) models of prototyping practice. The purpose of the field modelling was to learn more about the process by looking at each individual prototyping practice and also, to provide a framework for further detailed study. An overview of process modelling is given at the beginning of the chapter. RADs are chosen as the modelling technique for the study because it suits the purpose of the study: it enable us to capture an overall picture of process in terms of 'who', 'what', 'where' and 'when'. The findings of the field modelling are summarised at the end of this chapter, of which the most striking one is the diversity of the resulting process models.

5.1 Introduction

Having re-confirmed, by the preliminary survey (chapter 4), that management and control of software prototyping related issues were the major concerns of practitioners, the main objective of this stage was to find out how prototyping was practised by each of the eight organisations. The rationale was that the process must be better understood before it could be measured and improved. As defined in Methodology (chapter 3), RADs — one of the process modelling techniques — was used as the main method for work at this stage.

During the field modelling, 22 people (managers and prototypers) of 8 organisations were involved in modelling their prototyping processes. These organisations range from large (a few hundred software developers) international companies to a small software house (about 20 developers) and include an academic institute (refer to the table in Appendix B). The main result of the field modelling was a set of 10

process models showing their actual practice. For two organisations, there were two different prototyping processes within each and they were modelled separately. Here the process models mean the normal prototyping practice of the visited organisations or departments within the organisations. The following discussion centres around these models. It presents the facts gathered thus far and summarises the important issues. It also gives recommendations on some key issues regarding management and control of prototyping, and the possible areas for further investigation.

Before discussing the field modelling results, it is important to understand the rationale behind process modelling.

5.2 Method for Work

5.2.1 Process Modelling and Software Process

In the context of computing systems, the term 'process modelling' has come to be associated with ideas concerning the dynamic behaviour of organisations, businesses or systems more generally. The basic idea is that such systems can be thought of as operating or behaving as a number of interrelated processes. To study and understand systems, one constructs 'process models' according to particular viewpoints and using particular modelling techniques.

There are basically two types of approach to process modelling which are categorised as 'descriptive modelling' and 'active modelling' (Greenwood, Robertson et al. 1995). Descriptive modelling gives more information about techniques used for process models whose purpose is to describe processes and organisational behaviour. Active modelling gives more information about the idea that process models can be used to provide computer based support for businesses and other systems. Its development has coincided and overlaps with developing ideas in business organisations under the general heading of business process engineering or re-engineering. These seek to understand

businesses in terms of key processes and to offer principles for business organisations that maximise the effectiveness of these key processes and thereby of the business itself. In publications such as Management in the Nineties (Morton 1990) relationships are drawn between business organisations and the opportunities offered by developments in information technology.

Historically, process modelling had its origins in concerns for the software life cycle and the software development process. In the 1970s it was clear that the production of computer based systems, and of software specifically, presented problems which were not usually present in more familiar 'manufactured' products. As computer systems became more complex and more pervasive, this contrast became increasingly apparent. The first conference on the 'Software Process' was held in England in 1984 (Greenwood, Robertson et al. 1995) and has been followed by many others since. The IOPT (Introduction of Process Technology) was set up in 1992 in England aiming to widen industry understanding of Process Technology and its application.

Various models of the software development process have been suggested and a number of modelling techniques developed, frequently associated with some form of computer support to provide assistance for software developers in following such a development processes (Grief 1988; Maresh and Wastell 1990). The Software Engineering Institute at Carnegie Mellon University in Pittsburgh has developed a 'maturity model' by which software development organisations can be assessed according to their adherence to 'good' software development practices (Humphrey 1988). The European Software Institute in Bilbao is also working in this area. The paper by Curtis et al (Curtis, Kellner et al. 1992) gives a reasonable summary of present software process modelling approaches, whilst the book edited by Finkelstein, Kramer and Nuseibeh (Finkelstein, Kramer et al. 1994) provides a more up-to-date description of many developments in Europe concerned with process modelling for software development.

5.2.2 Methods Used and Rationale

As stated in 5.1, the main purpose of this stage of the study is to have more understanding about the prototyping process in terms of management and control. This clearly can be offered by using descriptive modelling, for it mainly concerns describing processes rather than automating process support.

Such models may be formed in a variety of ways, using different techniques. Generally, such techniques will be supported by software tools which enable the modeller to create the models. A number of specific techniques and tools have been developed to support the production of such models. Some tools are particularly concerned with providing a representation, an example is the IDEF family (FIPS 1993). Other tools allow properties of descriptive models to be analysed using techniques such as finite element simulation, difference equations or execution of rules. Examples of these include Systems Dynamics models supported by tools such as iThink and ProcessWise Workbench (ICL).

There are many existing process representation notations that can be used for process modelling. Some of the commonly used are: Flowcharts, Petri Nets (Reisig 1982), ETVX (Radice and Phillips 1988), Information Mapping[6], IDEF (FIPS 1993), MVP-L (Multi-View Process Modelling Language) (Rombach 1991), and RADs (Role Activity Diagrams) (Holt, Ramsey et al. 1983). Having compared some of the modelling tools, RADs are found to be particularly suitable for the purpose of this study. Firstly, RADs are designed to capture a process in terms of process roles, process entities, role activities, and role interactions; secondly RADs are a diagrammatic modelling technique consisting of a number of simple notional primitives, which are intuitive, easy to understand and use, and yet enable us to express very complex process behaviour (for RADs notations refer to Appendix C).

[6]http://www.infomap.com

5.3 Limitations

The limitations of the field modelling were:

1) The short time period for information gathering and clarification. Normally it took about one hour for the first interview, including drafting the RADs, and about the same length of time for refining and clarifying the first draft during the second interview. So, some degree of inaccuracy and incompleteness is inevitable.

2) Inability to include all process roles in the field modelling. The process models were derived mostly by interviewing two main process roles about their processes: the project manager and/or prototyper. Other process roles such as customer and/or user were unable to be included in the interviews due to the resources these companies offered. The RADs models are therefore not 'full' perspectives in term of all the roles involved in the process.

3) Lack of choice of the type and the number of organisations involved due to various practical difficulties such as willingness and resource availability. As a result of the questionnaire survey, only eight out of 80 companies showed an interest in further co-operation.

4) Although RADs (Role Activity Diagrams) offer a straight-forward view of a process in term of the roles, the activities under each role and the role interactions, they may be unsuitable for representing other aspects of a process such as hierarchical structure and data flows.

5.4 Terms Used

There follows a brief description of the modelling terms used.

1) All the processes are referred to as process 1, process 2 and so on, instead of referring to companies by name due to confidentiality. Some general background information for each company can be found in Appendix B.

2) "Process model" defined here as the typical way that prototyping projects are being carried out.

3) Role is defined here as a significant and distinct function in a process. So one role may be performed by one or more people and vice visa.

4) Role activity/private activity is an action(s) performed only by one role.

5) Interaction/roles interaction/public activity is action(s) involving two or more roles.

6) Role switching means the role performer switching his or her role from one to another. e.g. manager to prototyper.

7) Role coupling means the tightness between two or more roles in term of frequency of role switching and interactions.

8) Process activity means either private or public activity or both.

5.5 Process Characteristics

In this section, process characteristics are summarised in terms of process roles, their interactions and activities, which are based on the following ten Role Activity Diagrams derived from different prototyping practice. They are presented here to show an overall picture of these processes. For larger versions of these RADs, refer to Appendix D.

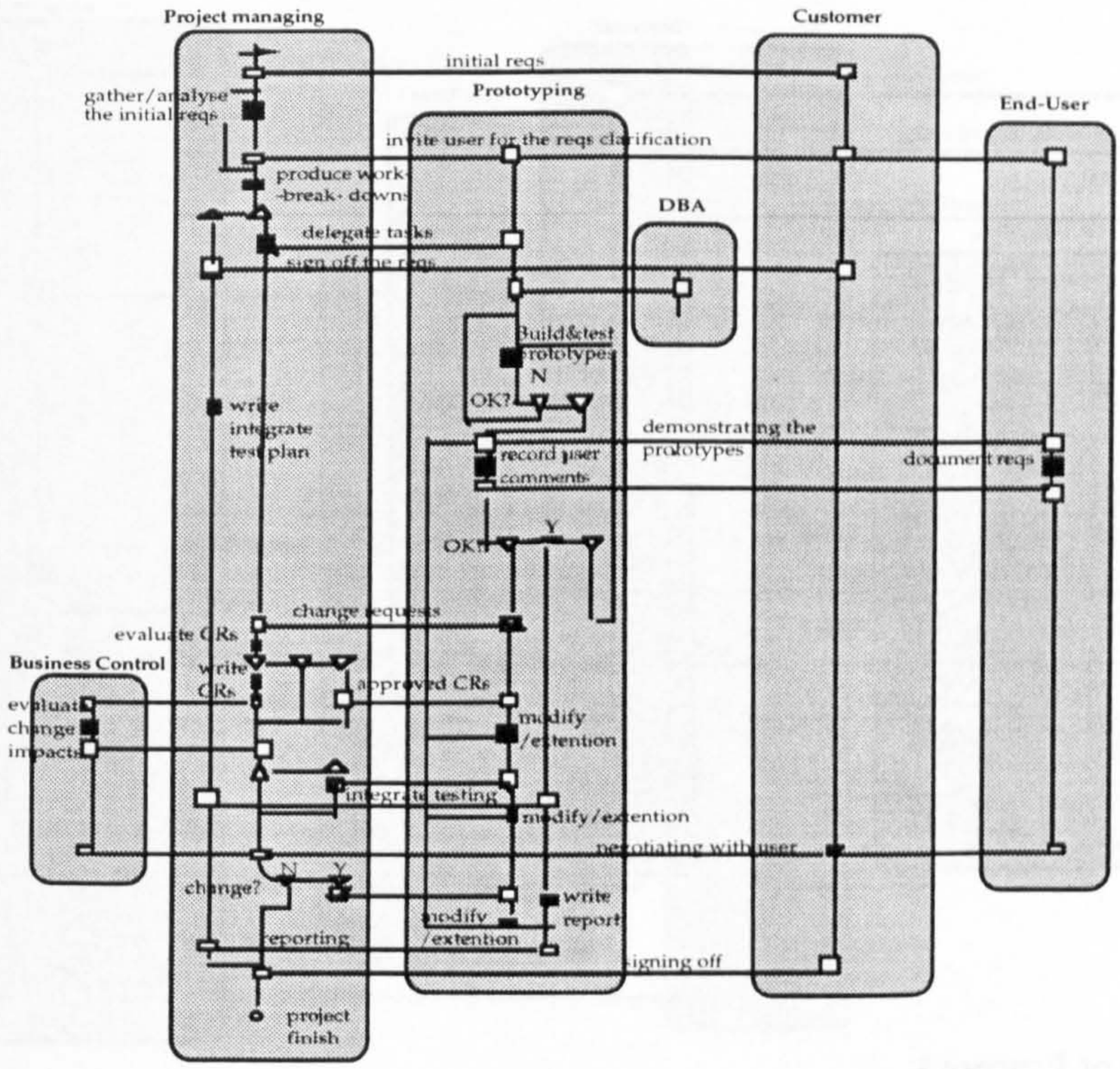


Diagram 5.1 RADs of Process 1

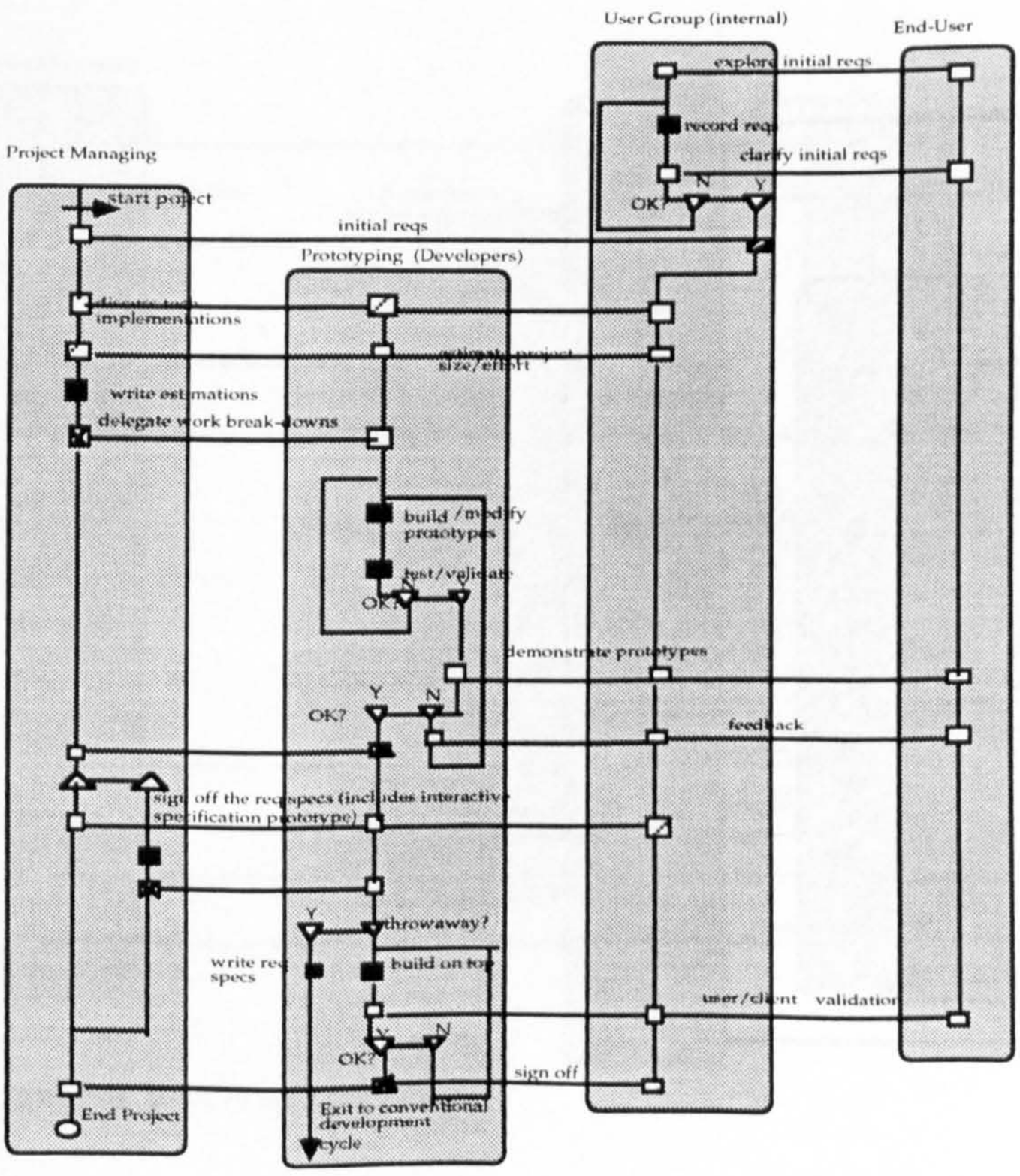


Diagram 5.2 RADs of Process 2

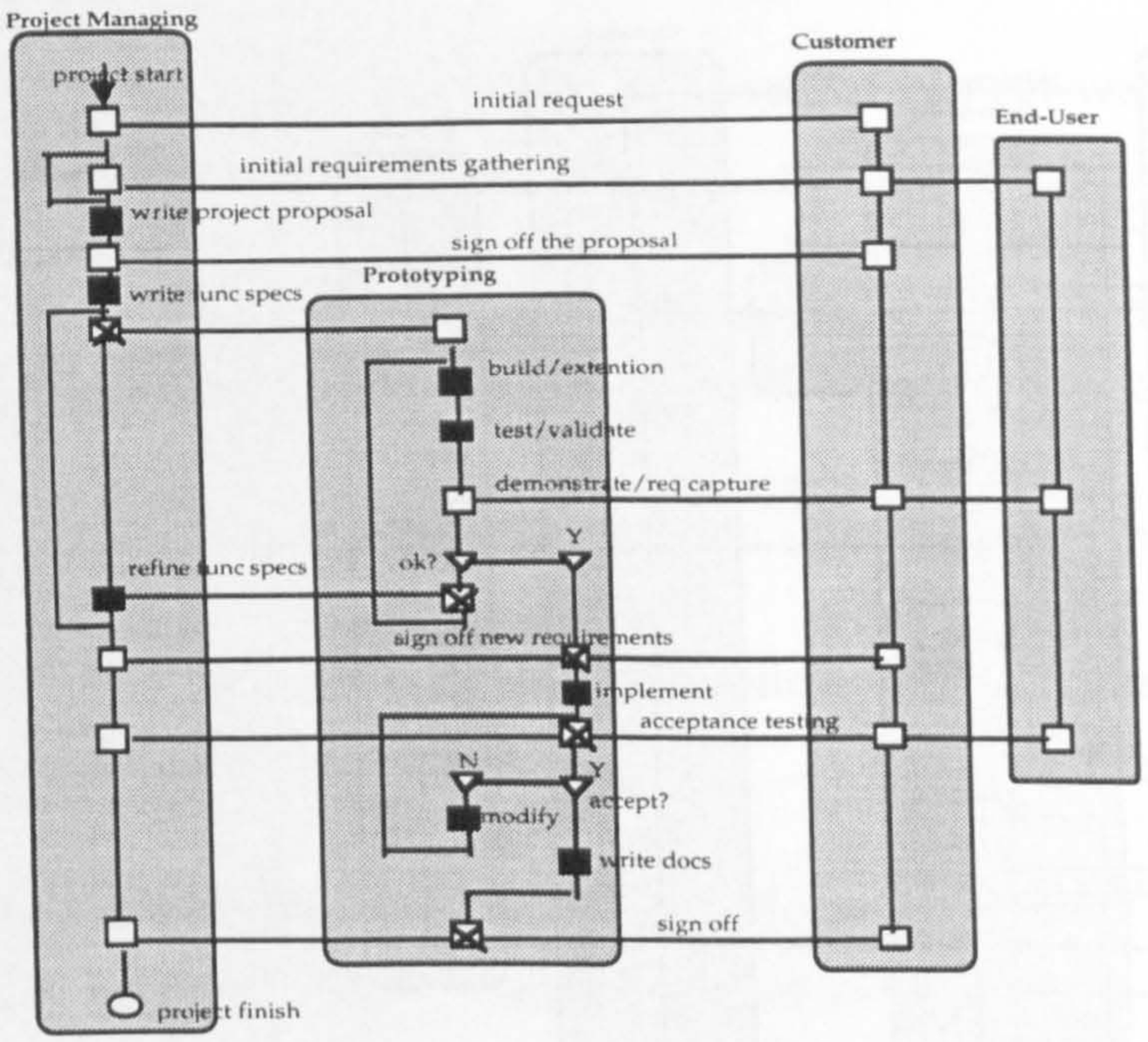


Diagram 5.3 RADs of Process 3

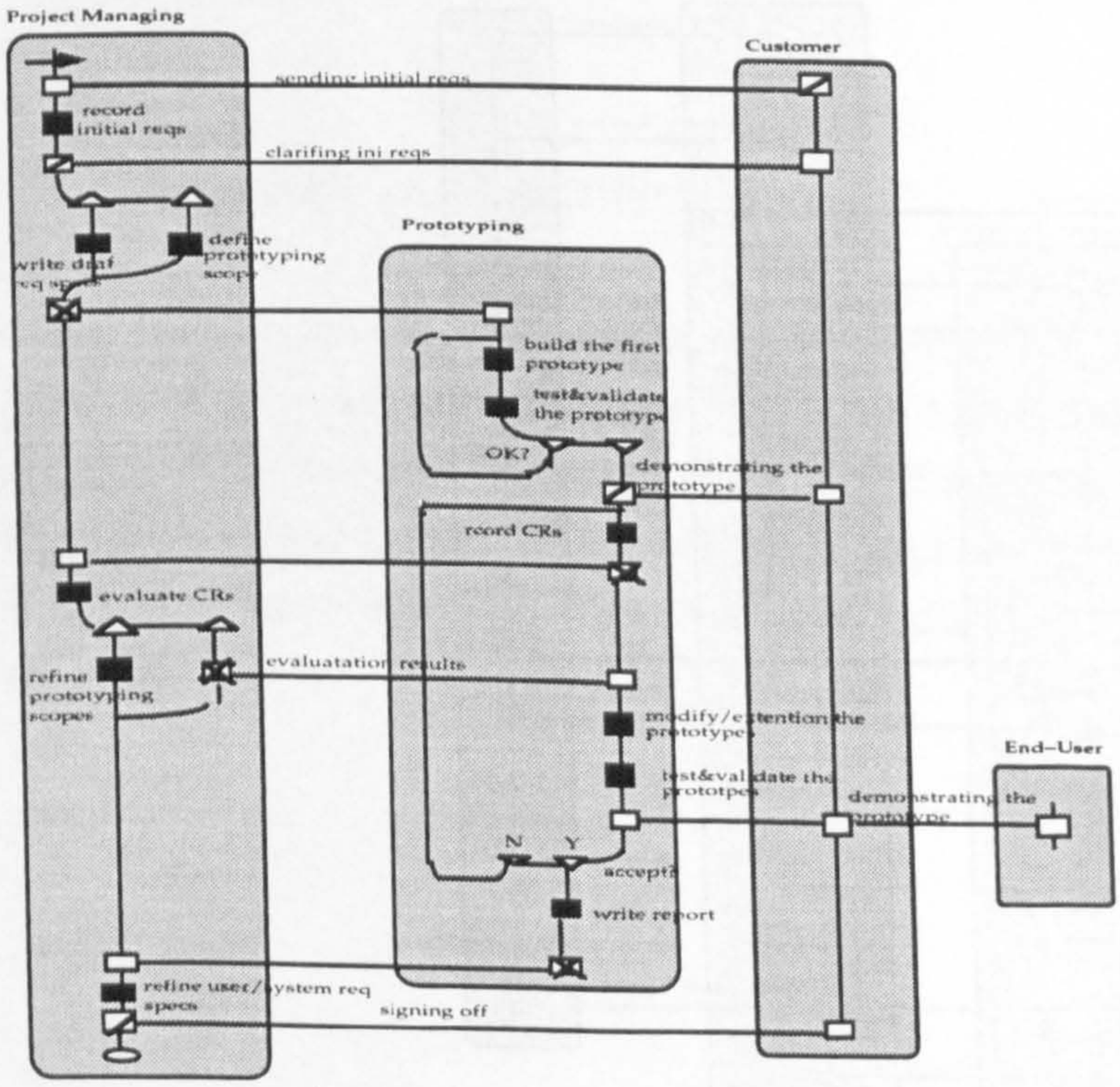


Diagram 5.4 RADs of Process 4

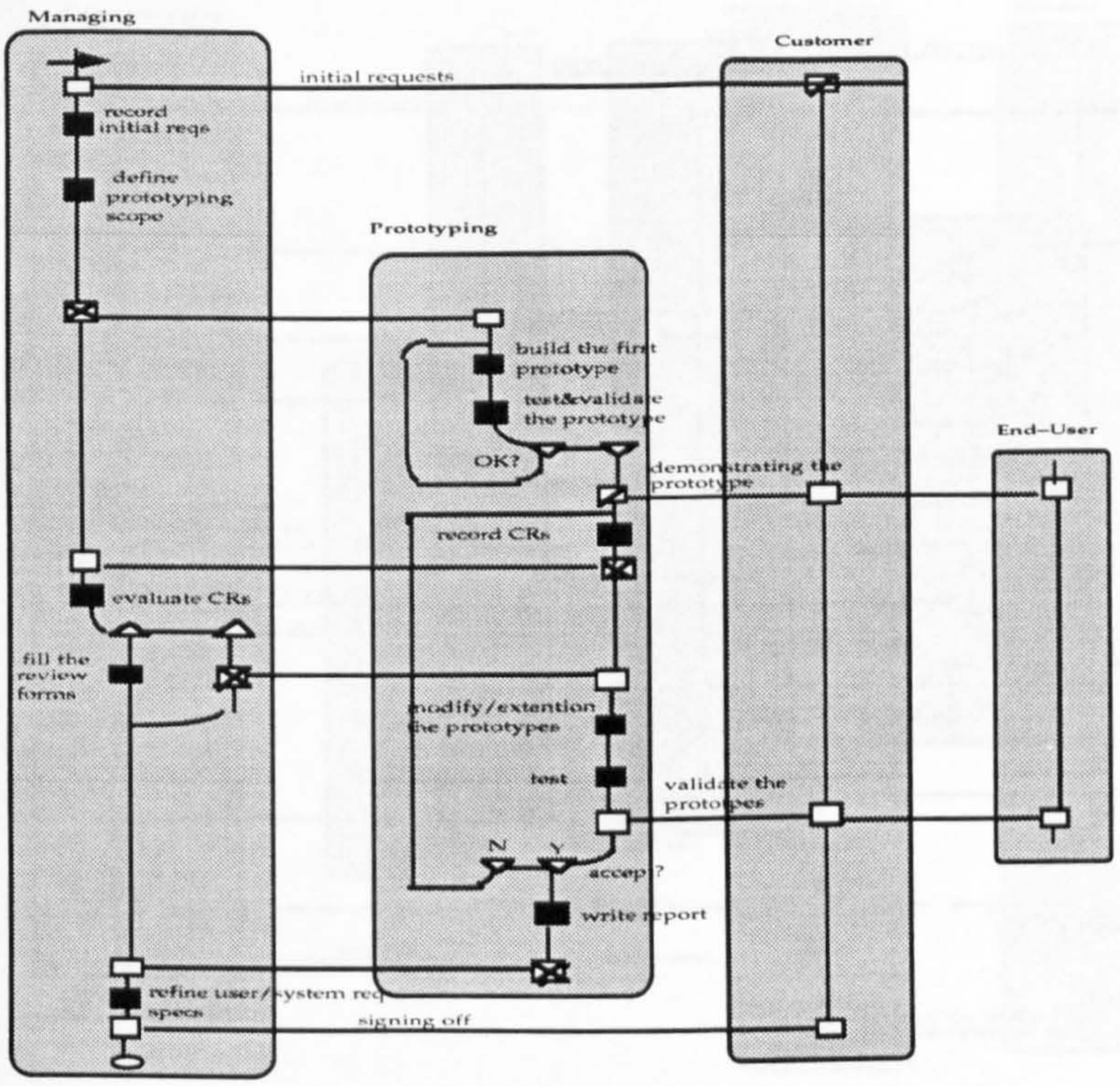


Diagram 5.5 RADs of Process 5

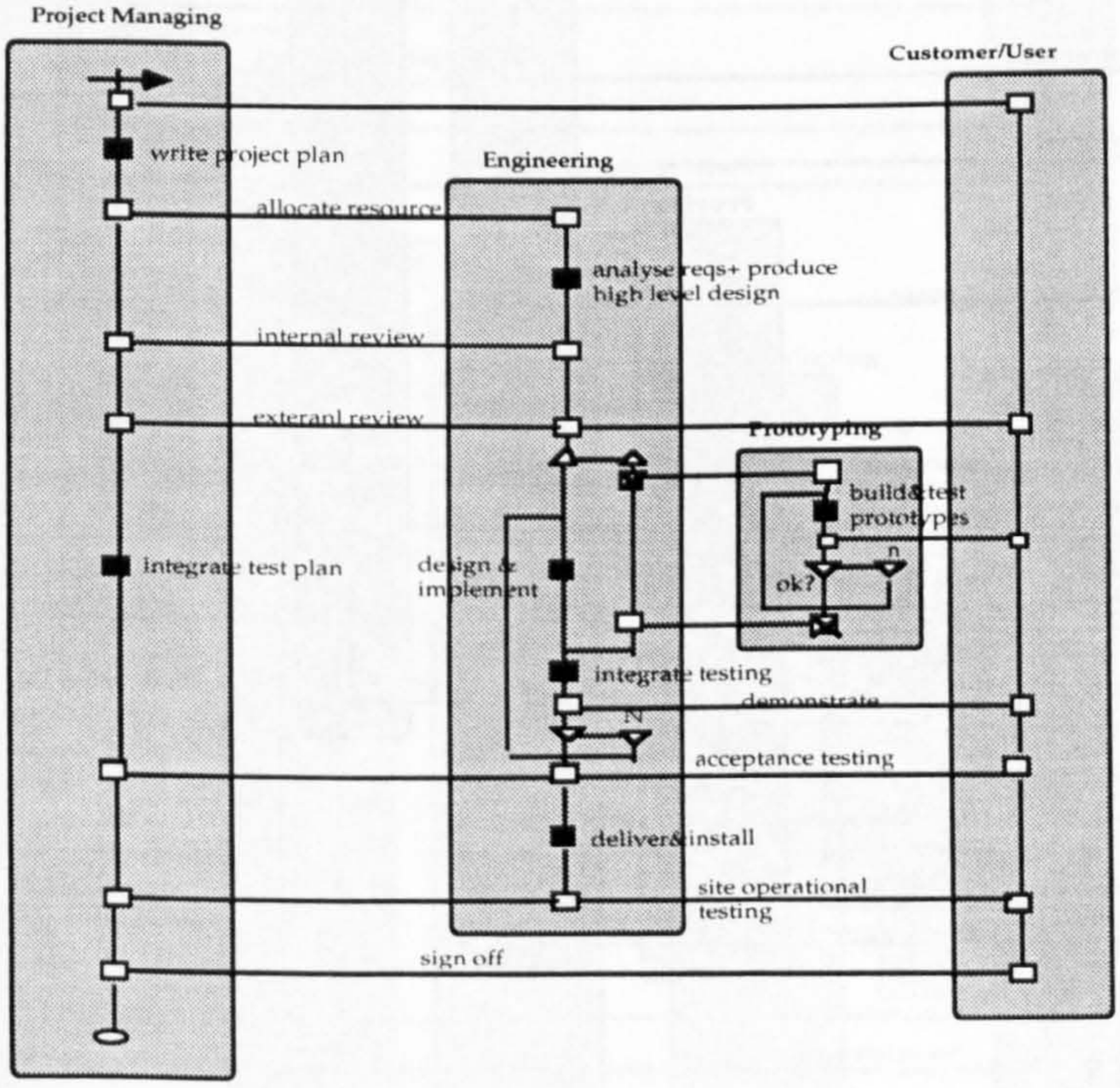


Diagram 5.6 RADs of Process 6

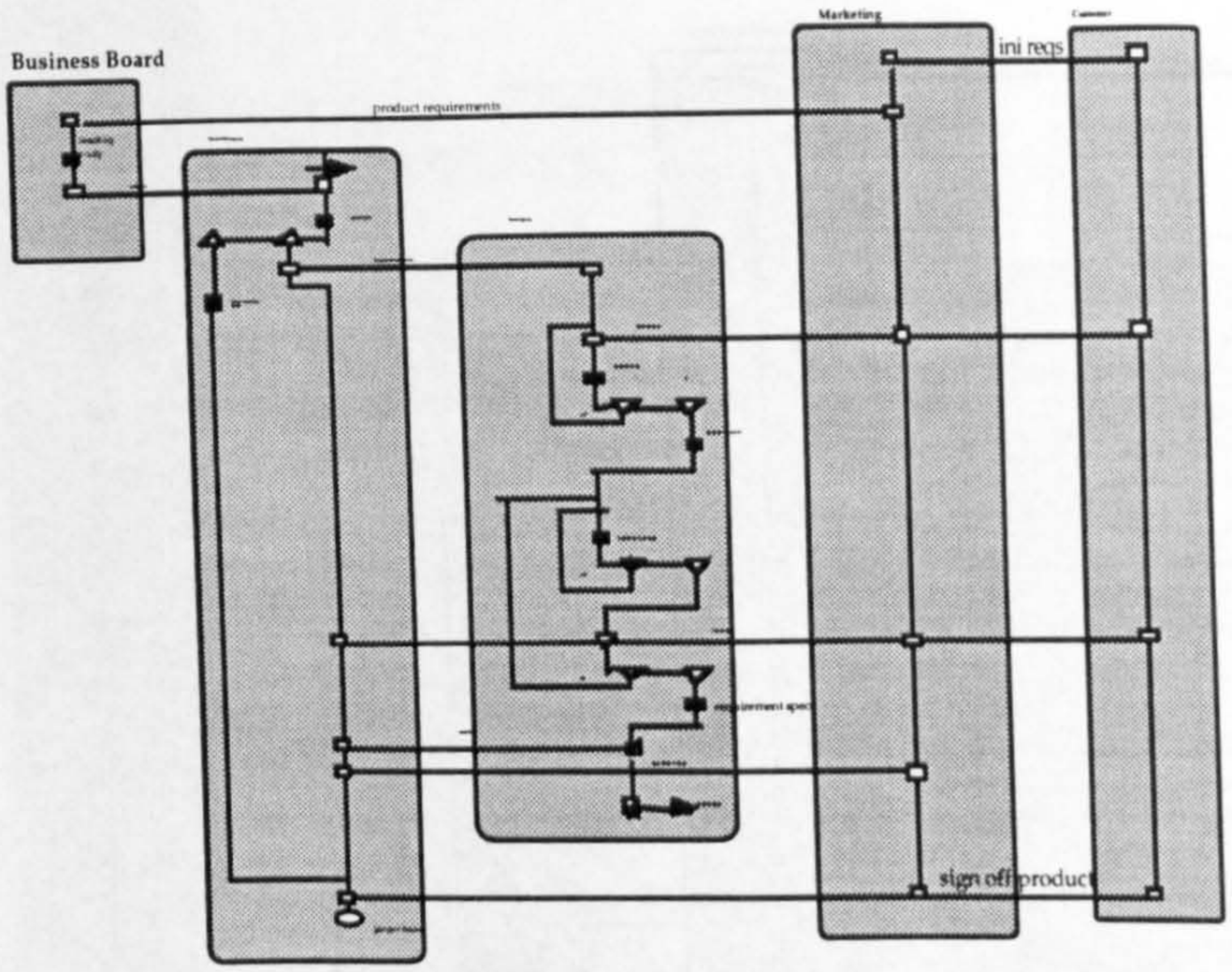


Diagram 5.7 RADs of Process 7

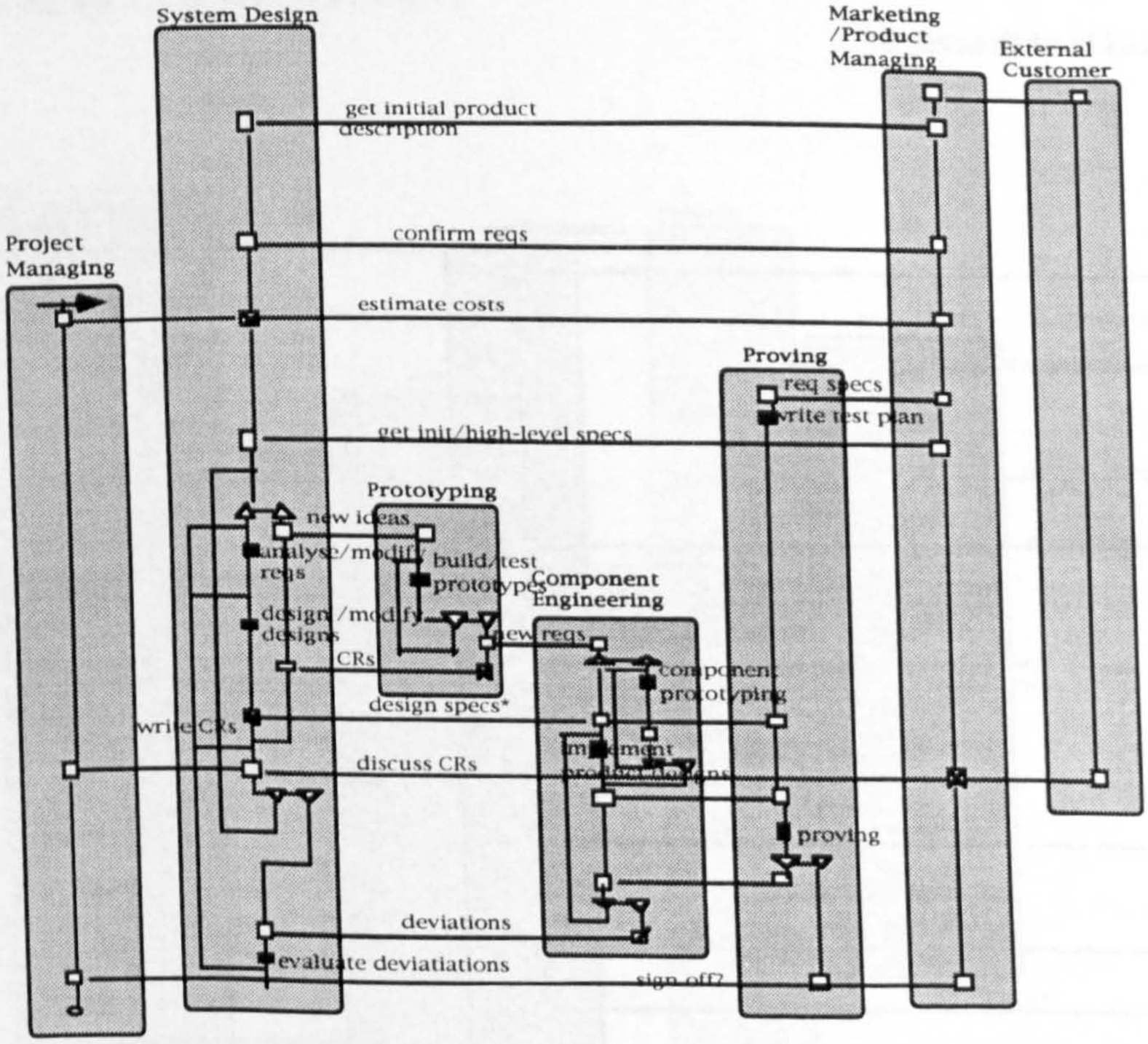


Diagram 5.8 RADs of Process 8

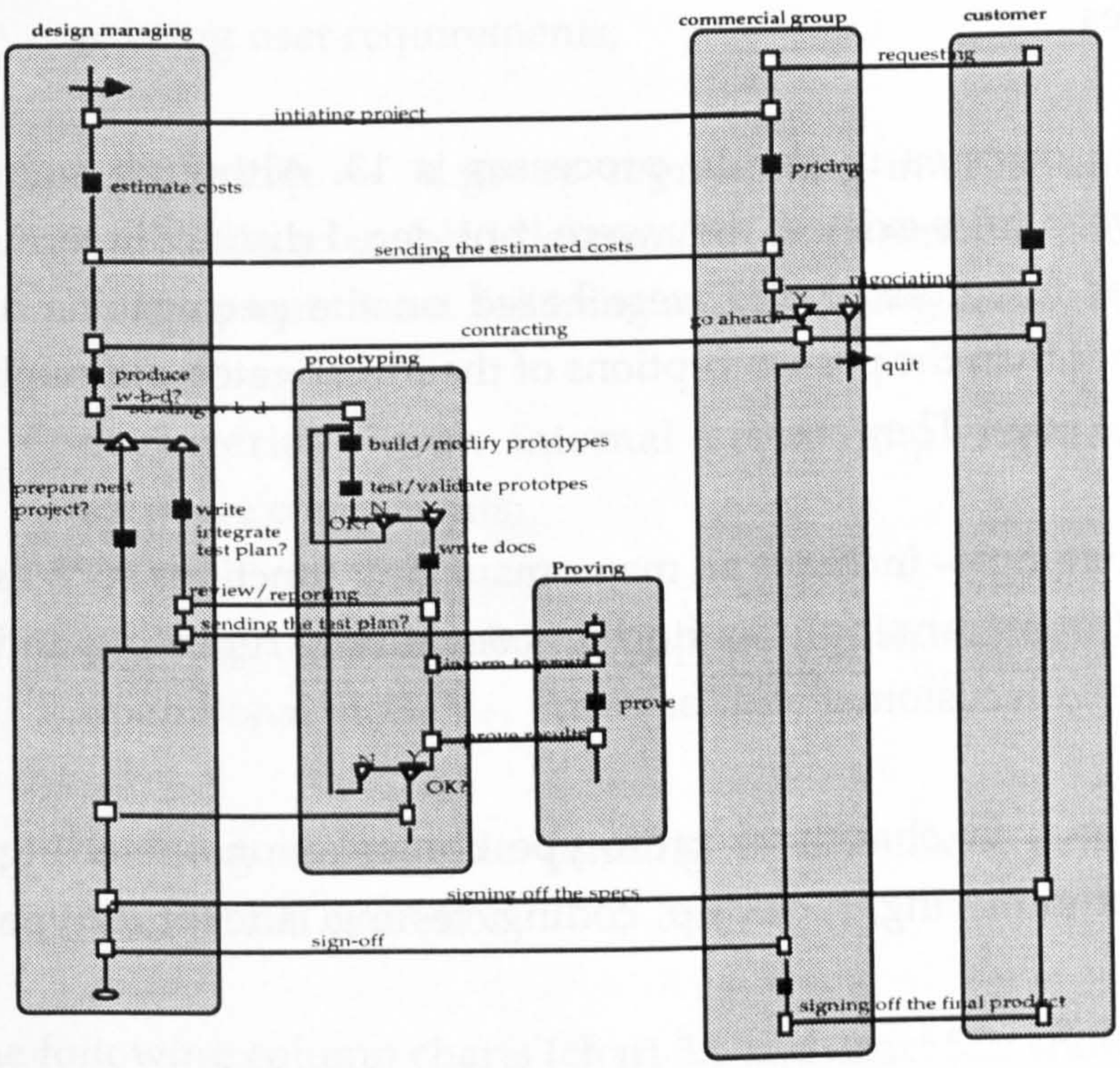


Diagram 5.9 RADs of Process 9

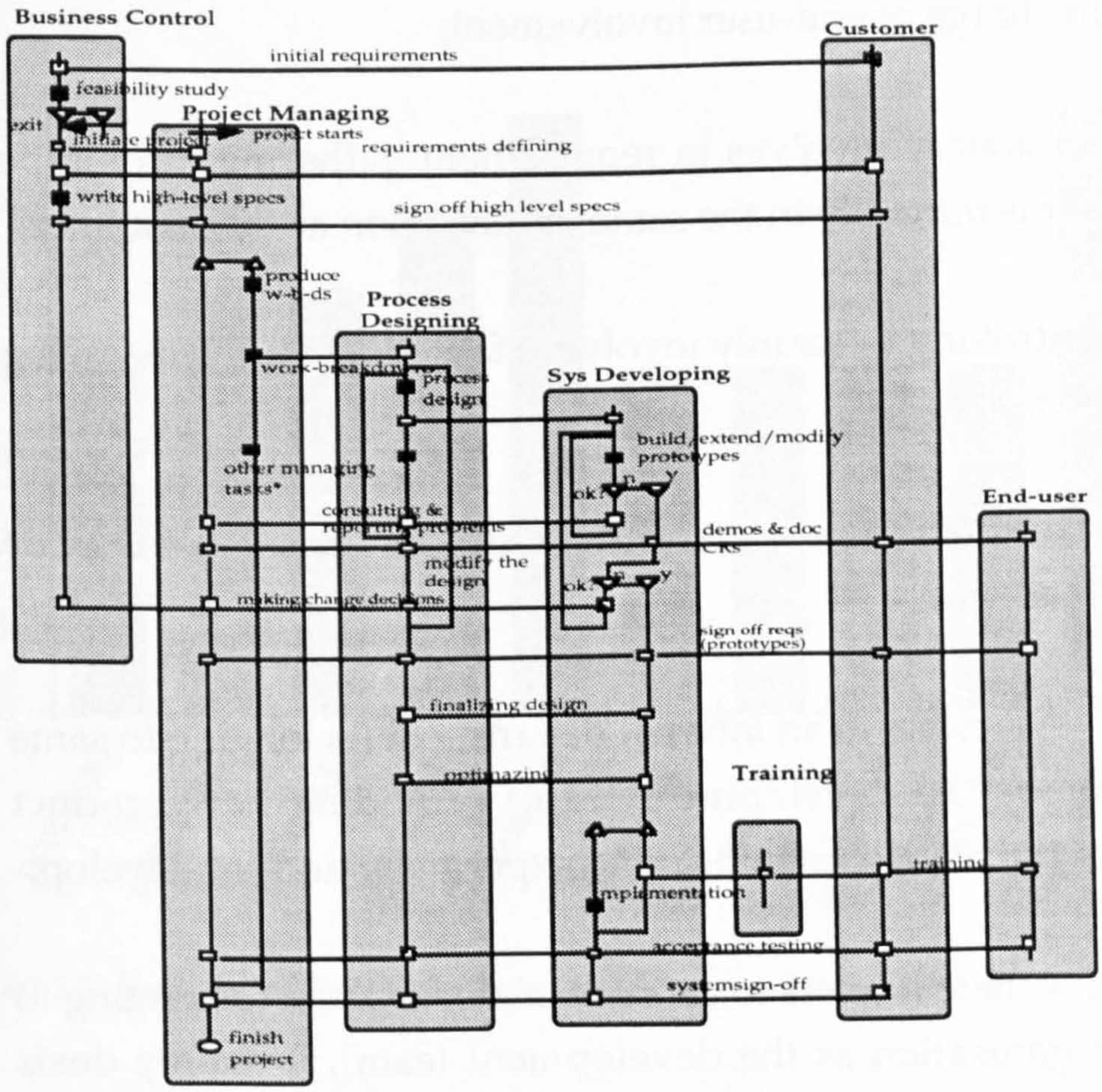


Diagram 5.10 RADs of Process 10

5.5.1 Process Roles

The total number of roles in the 10 processes is 13. Although some overlapping between roles existed, they were considered distinct in terms of the their major functions. Roles were based on the perceptions of organisations rather than the pre-conceptions of the investigator, although synonyms were avoided. They are:

- project managing — includes all major managing functions such as project staffing, planning, monitoring, controlling, reporting and negotiating with customer and/or user;

- prototyper — in charge of prototype construction including requirement gathering, mock up, coding, testing, and prototype validating;

- customer — here it specifically refers to an external client. Its main functions are negotiating user requirements in terms of time and cost, and in charge of end-user involvement;

- end-user — mainly involves in requirement gathering, prototype validation. It is normally in the same organisation as the customer;

- business controlling — mainly involving feasibility study, assessing change impacts;

- database administration — administrating access to central or server databases;

- marketing — it refers to an internal department (belonging to same organisation as the development team) providing new product concepts for the development team to experiment and/or develop;

- commercial — here it refers to an internal department (belonging to the same organisation as the development team), it mainly deals with finance aspects of the project with customer;

- design — produce software designs either from new ideas or

existing user requirements;

- user group — a group representing internal end-users, which functions as a 'middle man' in between the end-user and the development team (eliciting and reporting user requirements);

- engineering — an internal group implementing and testing software components;

- proving — an internal group testing the software developed in operational mode;

- training — provide training to end-users before putting the software in operation.

The following column charts (chart 5.1 and chart 5.2) show the number of roles and their frequency in each process:

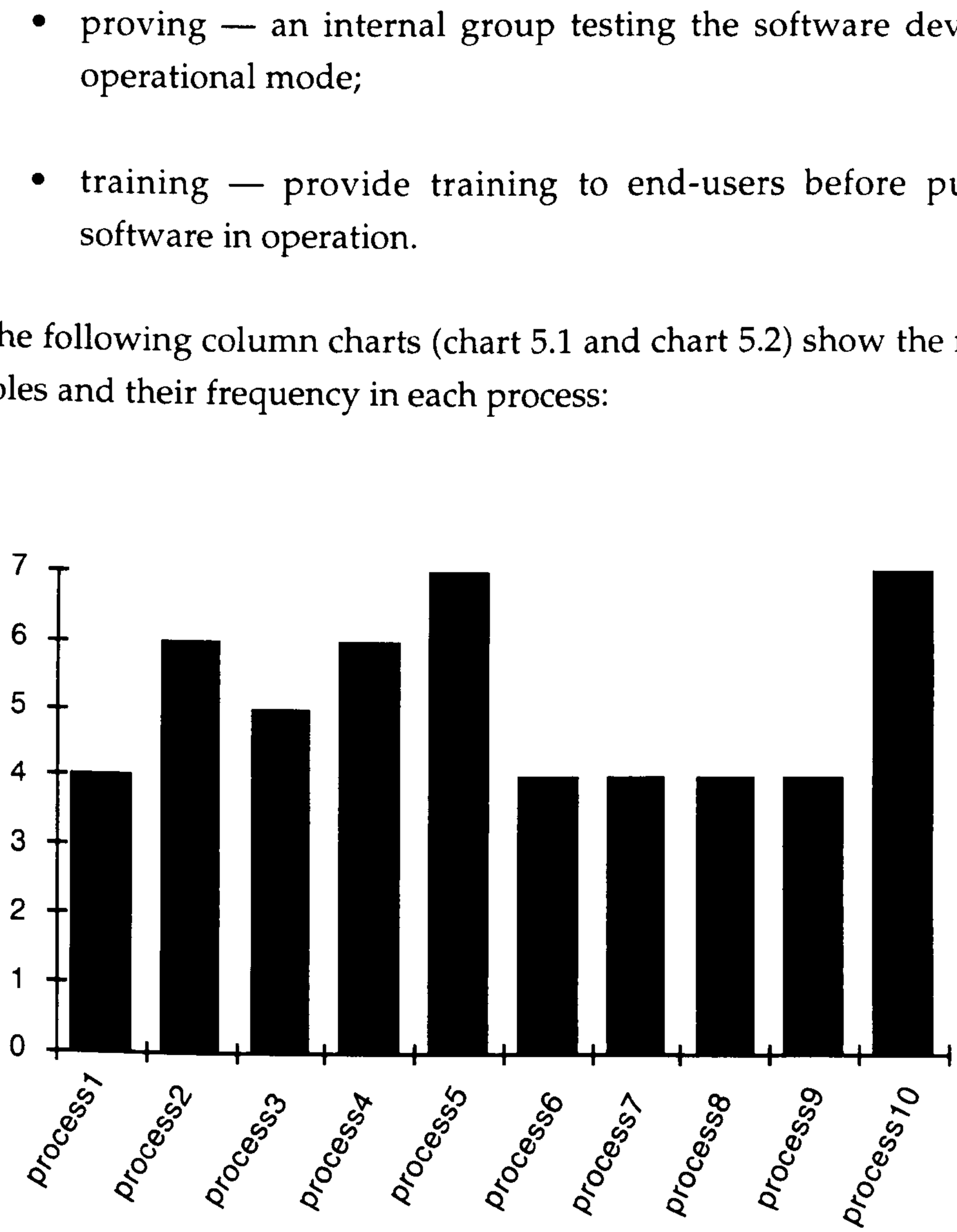


Chart 5.1 Number of Roles per Process

Chart 5.1, shows that the number of roles in the processes varies from a minimum of 4 to a maximum of 7 and, in most cases (5 out of 10) there are four roles in a process.

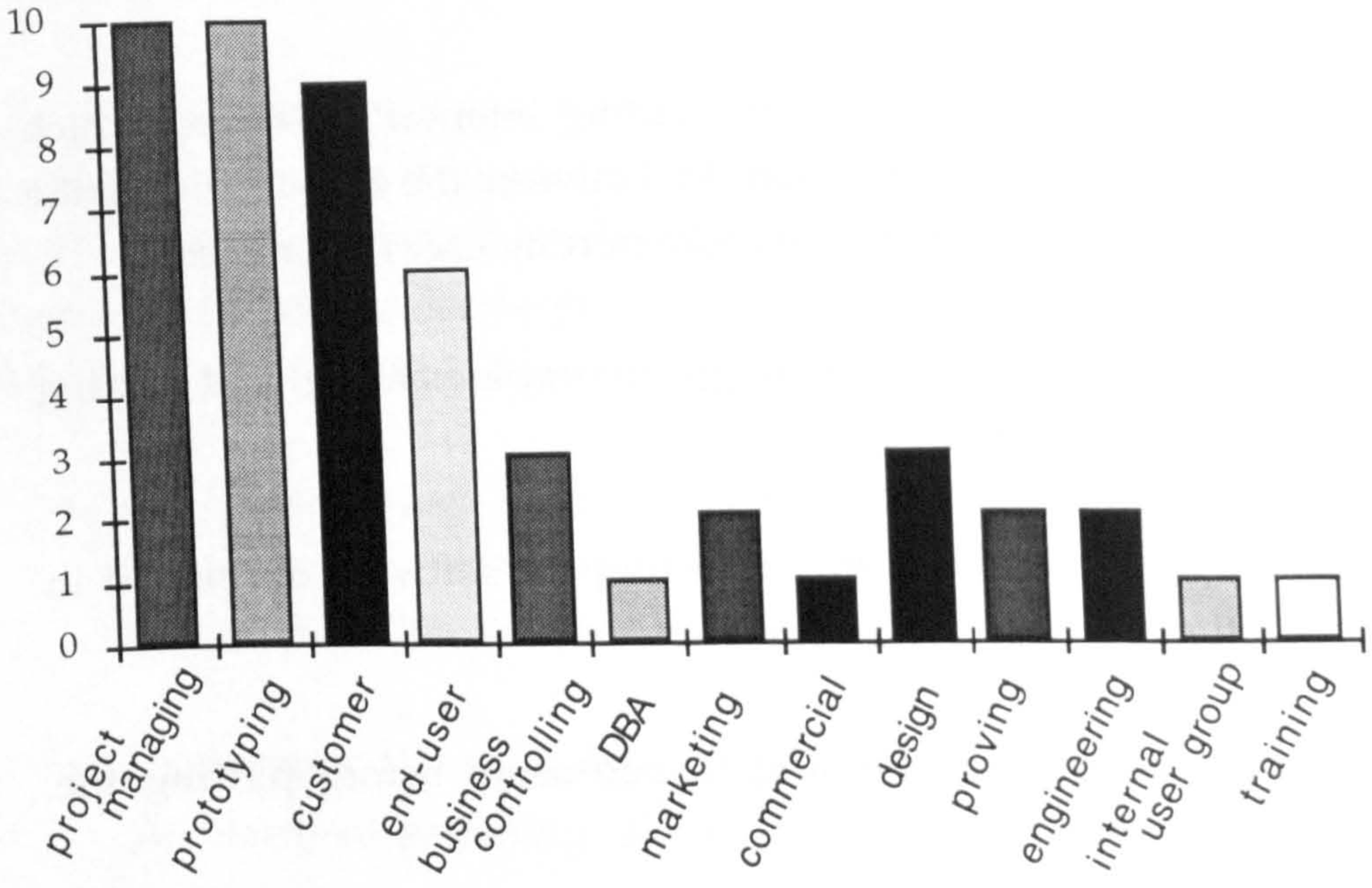


Chart 5.2 Frequency of Each Process Role

Chart 5.2 confirms that project managing, prototyping, customer and end-user are the most common roles, which appear on almost every process. Meanwhile, roles such as marketing (in process 7, 8), commercial (in process 9) and user-group (in process 2) show much similarity although they function slightly differently. The training role seemed only particular to process 10.

An important characteristics of prototyping practice was role switching which is indicated by a cross inside the interaction box in the processes (refer to above RADs or Appendix D). That means one role performer may change his or her role to another during the process. The most frequent role switching appeared to be happening between manager and prototyper: the prototyper(s) sometimes performs the managing role or vice versa. Process 1 is a good example of this. The main reason for doing so is to increase efficiency in decision-making so as to speed up the development process. However, this is an indication of a possible problematic area and needs further investigation because the mixed responsibilities may lead to over relaxed control on areas such as configuration management, and is likely to result in poor quality in terms of maintainability.

Finally, it should be noted that there was one particular case (process 9) where there was no 'end-user role' involved because the 'end-user' was a machine rather than a person.

5.5.2 Role Interactions

The characteristics of role interaction are explained by looking at the number of role interactions between roles. To keep the interpretation simple and make the comparison between processes clear, only the number of interactions between roles are counted disregarding the number of iterations. In other words, only interactions within one iteration are counted however many iterations occur.

The following charts (from chart 5.3 and 5.3′ to 5.12 and 5.12′) show from different perspectives the interaction between roles. While charts 5.3 to 5.12 give a three dimensional view of the interactions for each process[7], charts 5.3′ to 5.12′ intend to show the number of interactions between roles and the total number of interactions each role had with the remaining roles. They are arranged in pairs, e.g. chart 5.3 is followed by its complementary chart 5.3′.

[7]The scales on the vertical axis are the number of interactions. Roles are arranged on both axes of the horizontal plane: role 1 to n-1 on one axis and role 2 to n on the other. To avoid showing the same interaction twice only half of the plane is used.

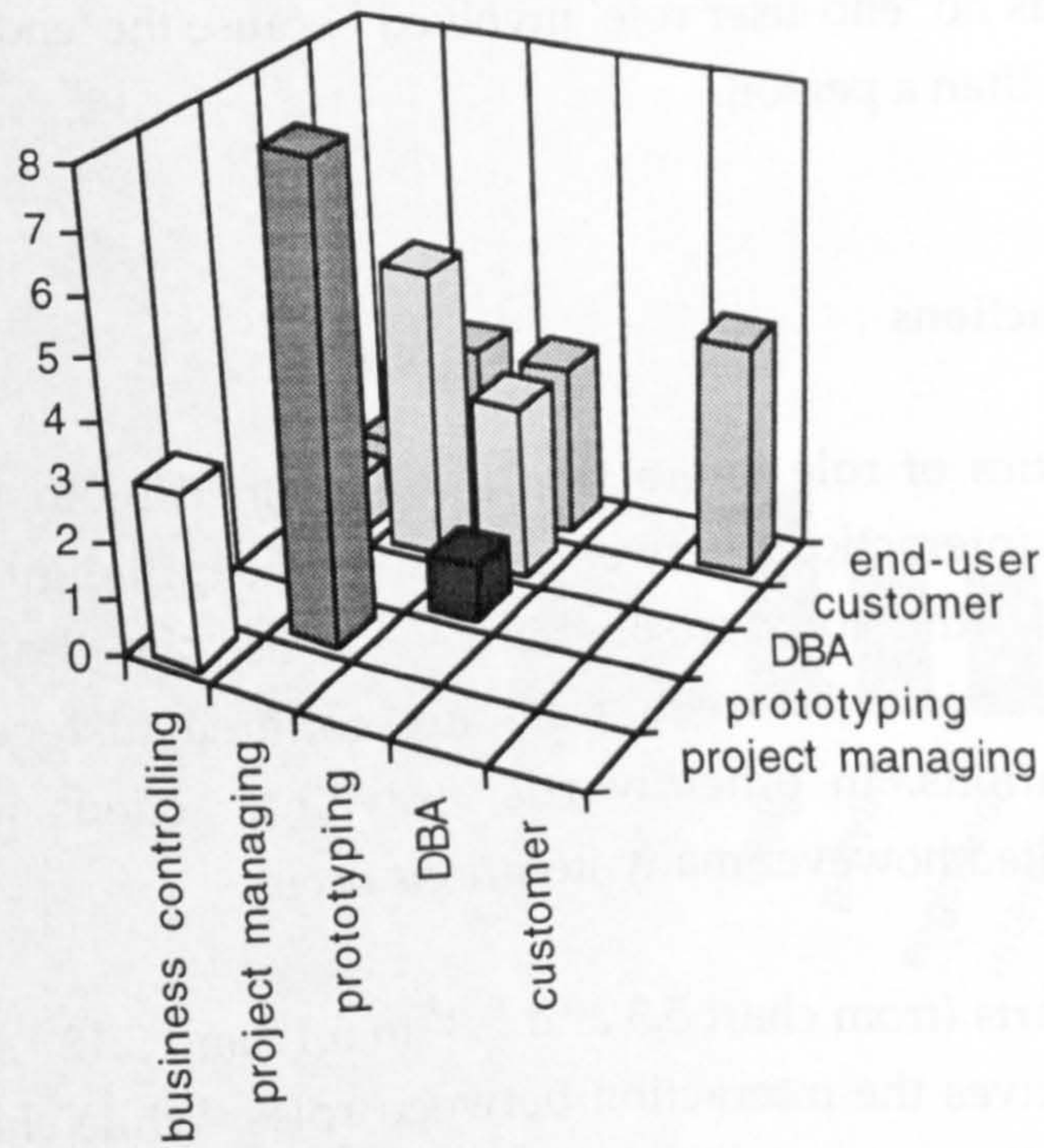


Chart 5.3 Role Interactions of Process 1

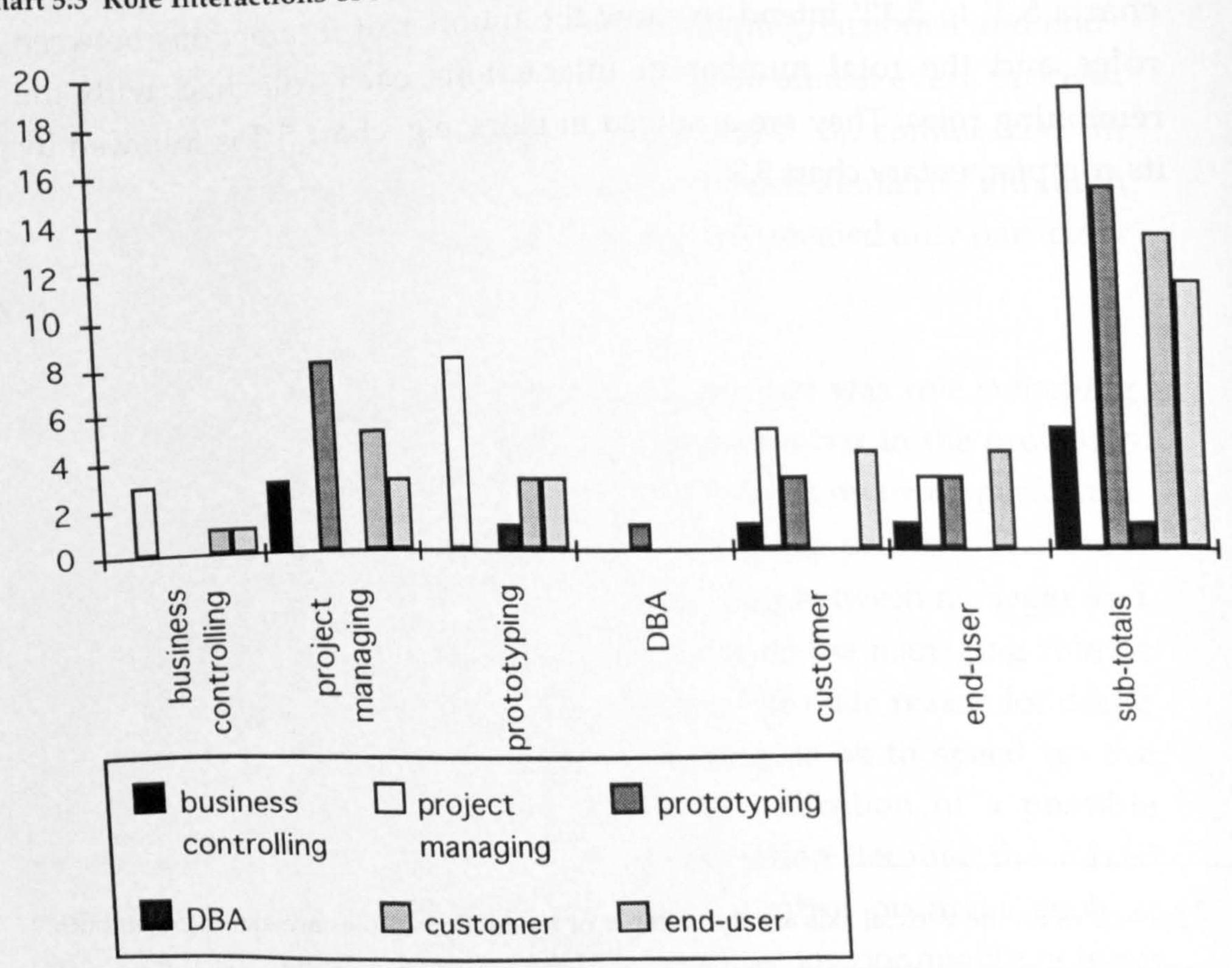


Chart 5.3 ' Number of interactions between roles in process 1

From chart 5.3 or 5.3′, it is clear that:

- project managing was the most interacted role of the process: the project managing role interacted with almost all other process roles except DBA with a total of 19 interactions during the process;

- DBA had least interactions, and had only one interaction with prototyping;

- project managing and prototyping were the two most closely interacted roles (8 interaction). Customer and user were also closely involved in the process;

- the business controlling role appeared mainly to interact with the project managing role at CR control stage.

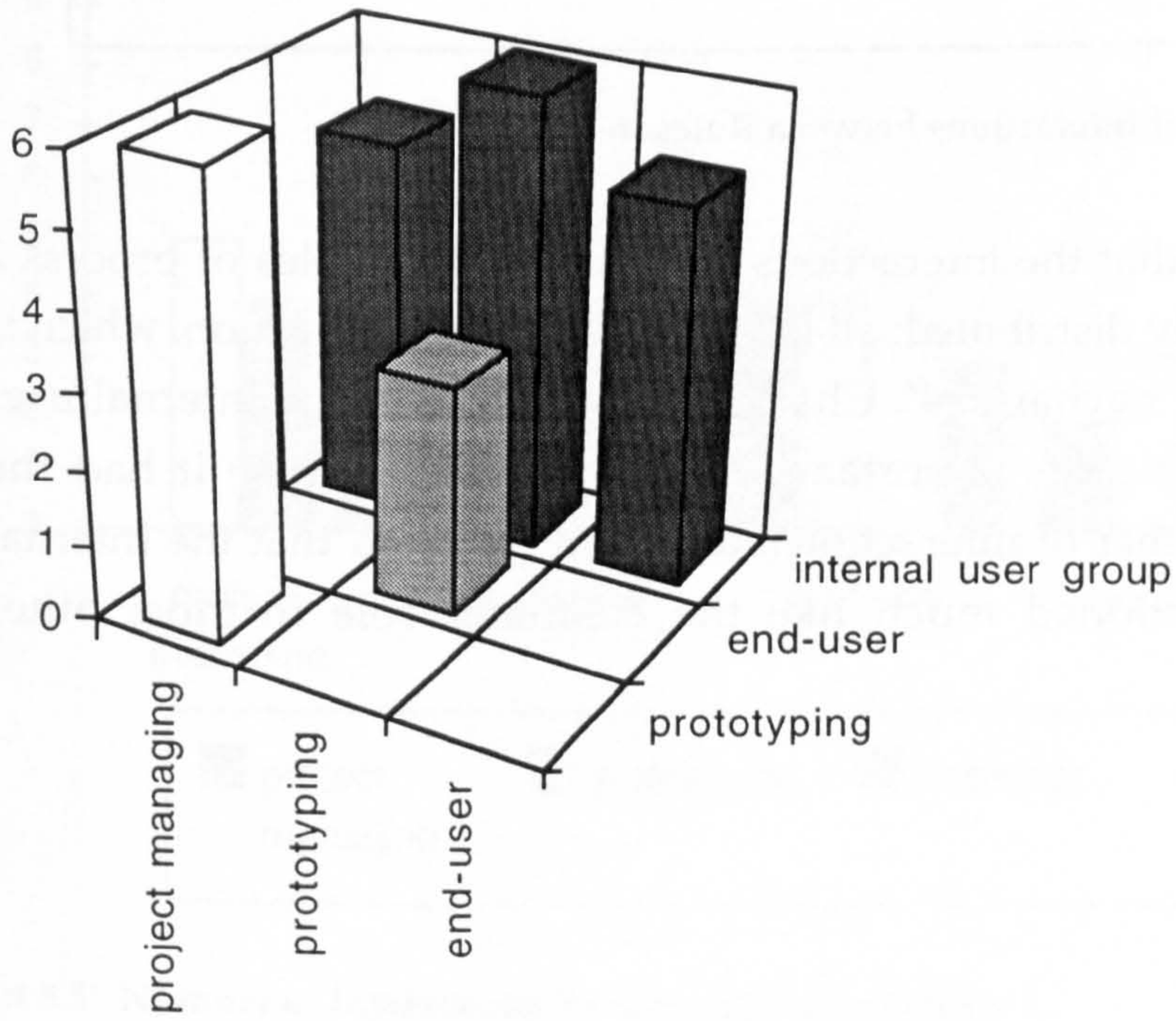


Chart 5.4 Role Interactions of Process 2

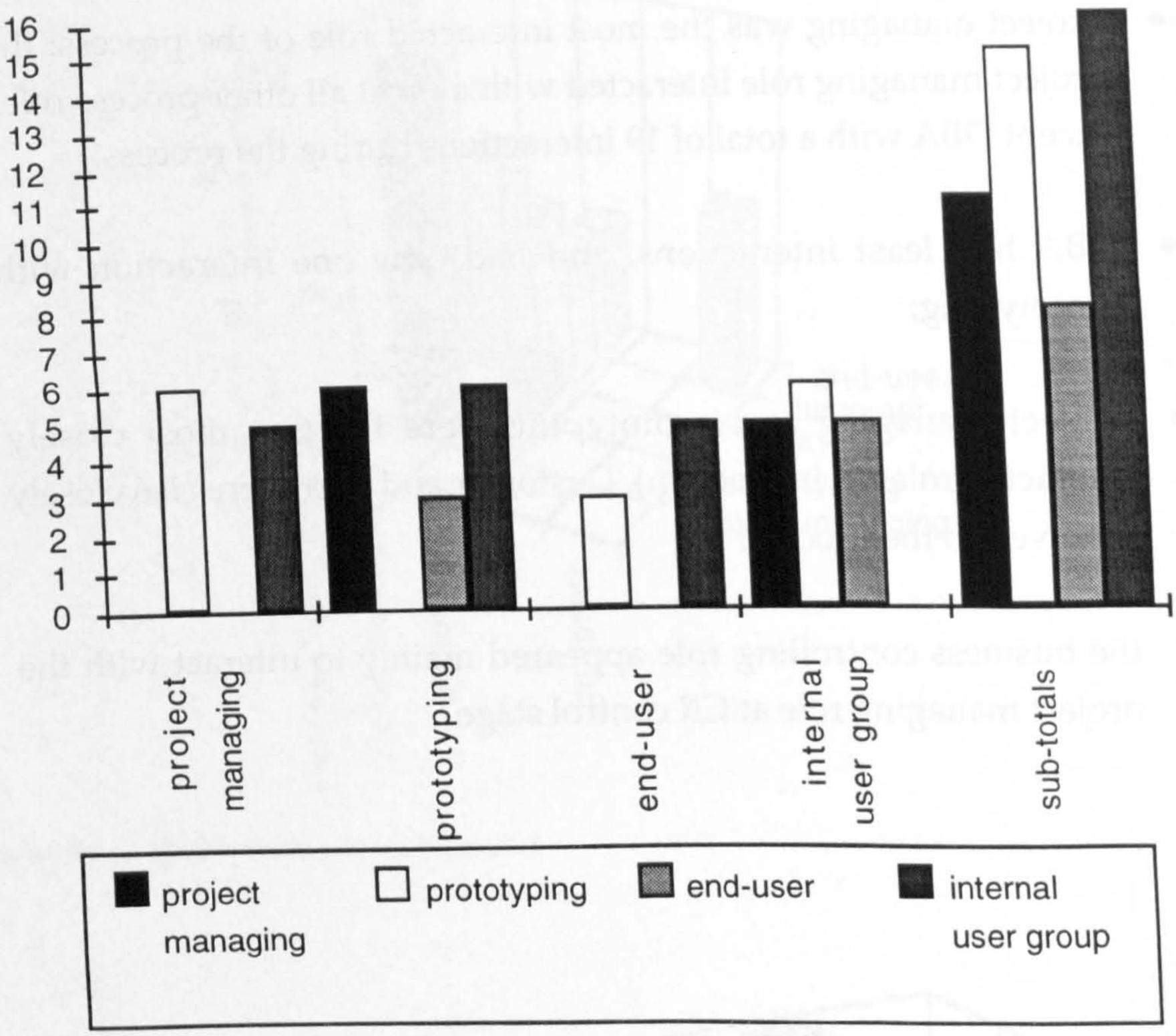


Chart 5.4′ Number of Interactions between Roles in Process 2

Chart 5.4 shows that the interactions between the four roles of process 2 were fairly equally distributed: all have high levels of interaction, which is further indicated in chart 5.4′. Chart 5.4′ also indicates the internal user group had a particular importance in this process because it had the highest total number of interactions. It should be noted that the internal user group functioned much like the customer role in most other processes.

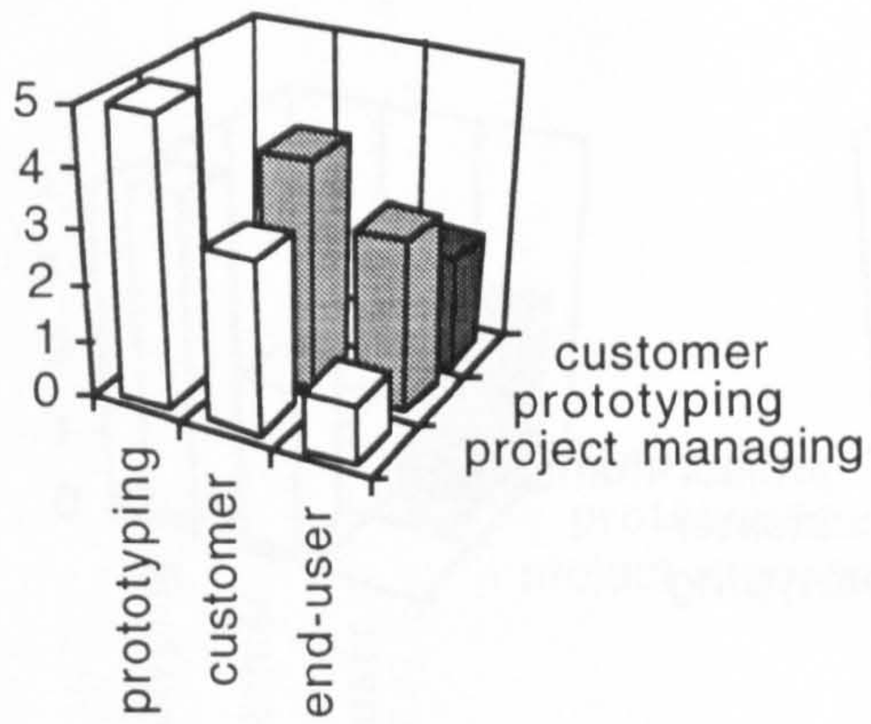


Chart 5.5 Role Interactions of Process 3

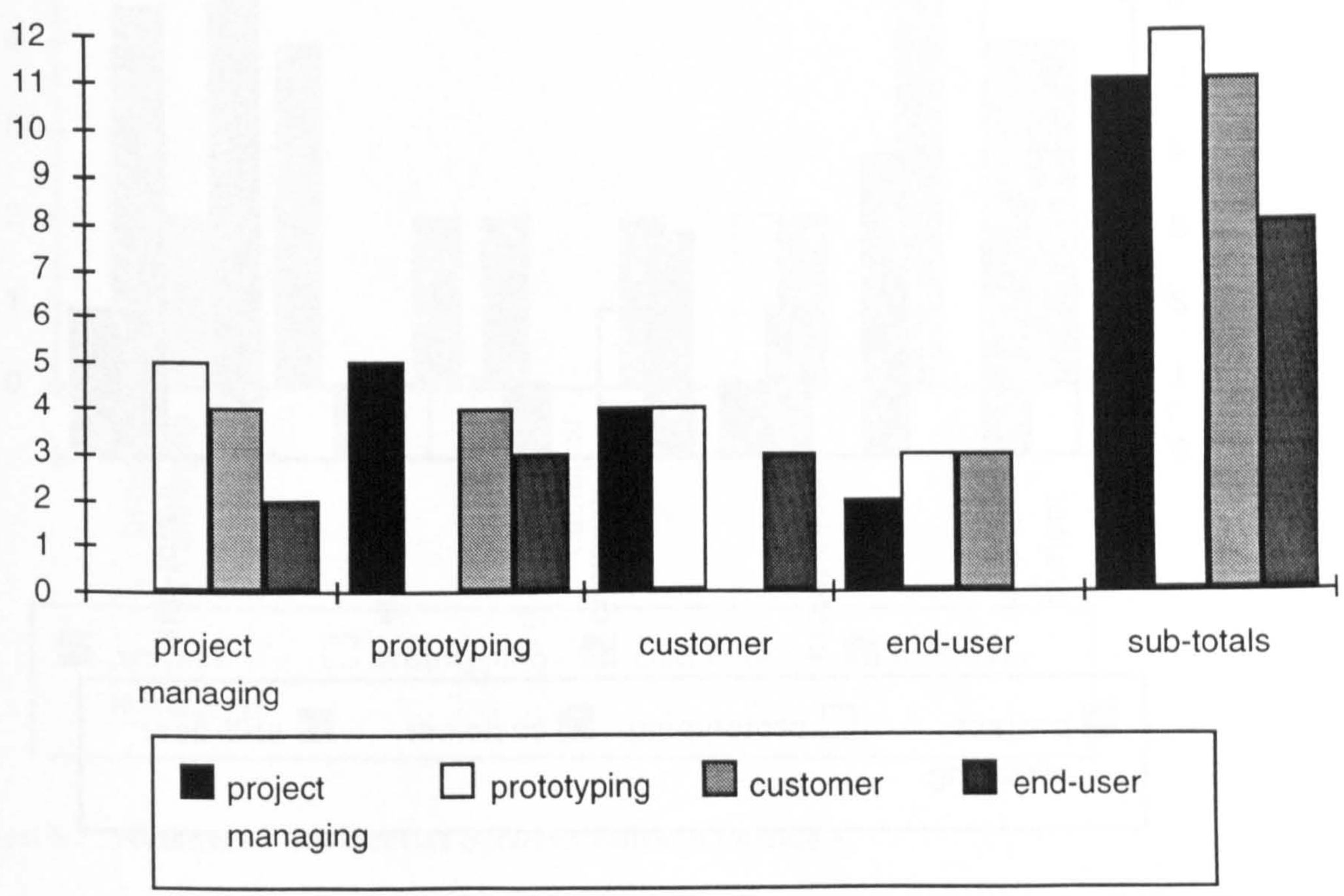


Chart 5.5′ Number of Interactions Between Roles in Process 3

Chart 5.5 and 5.5′ show that all the four roles were closely involved with each other: each role had interactions with the other three roles. From the 'sub-totals' it is also clear that prototyping was the most active role here.

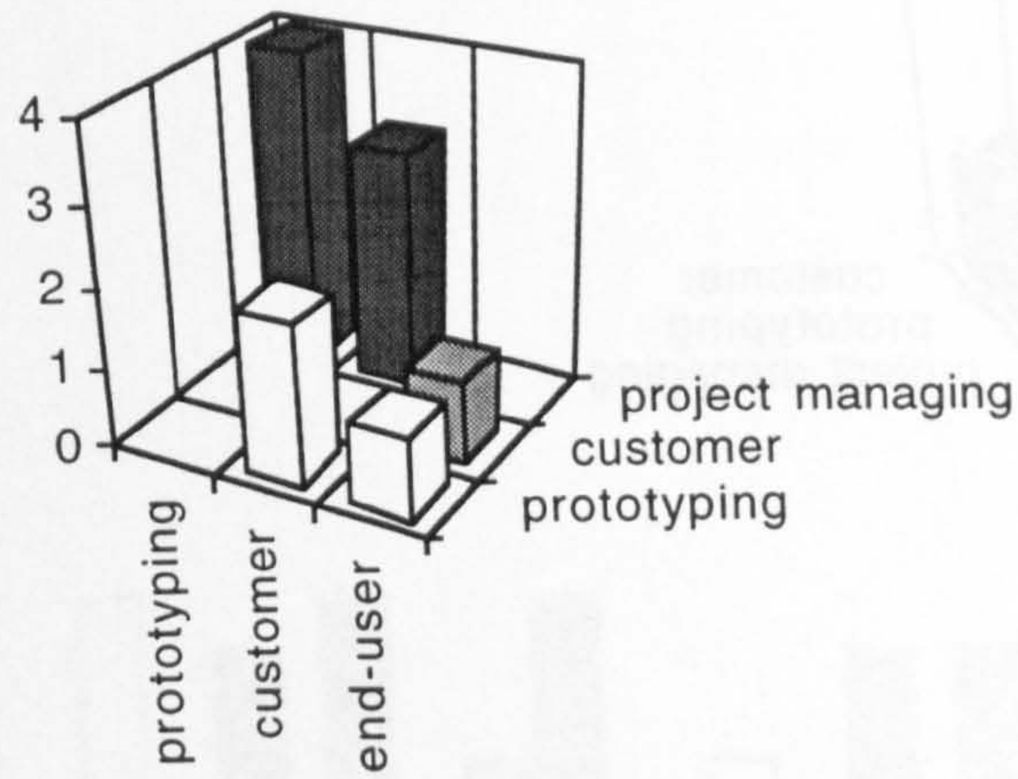


Chart 5.6 Role Interactions of Process 4

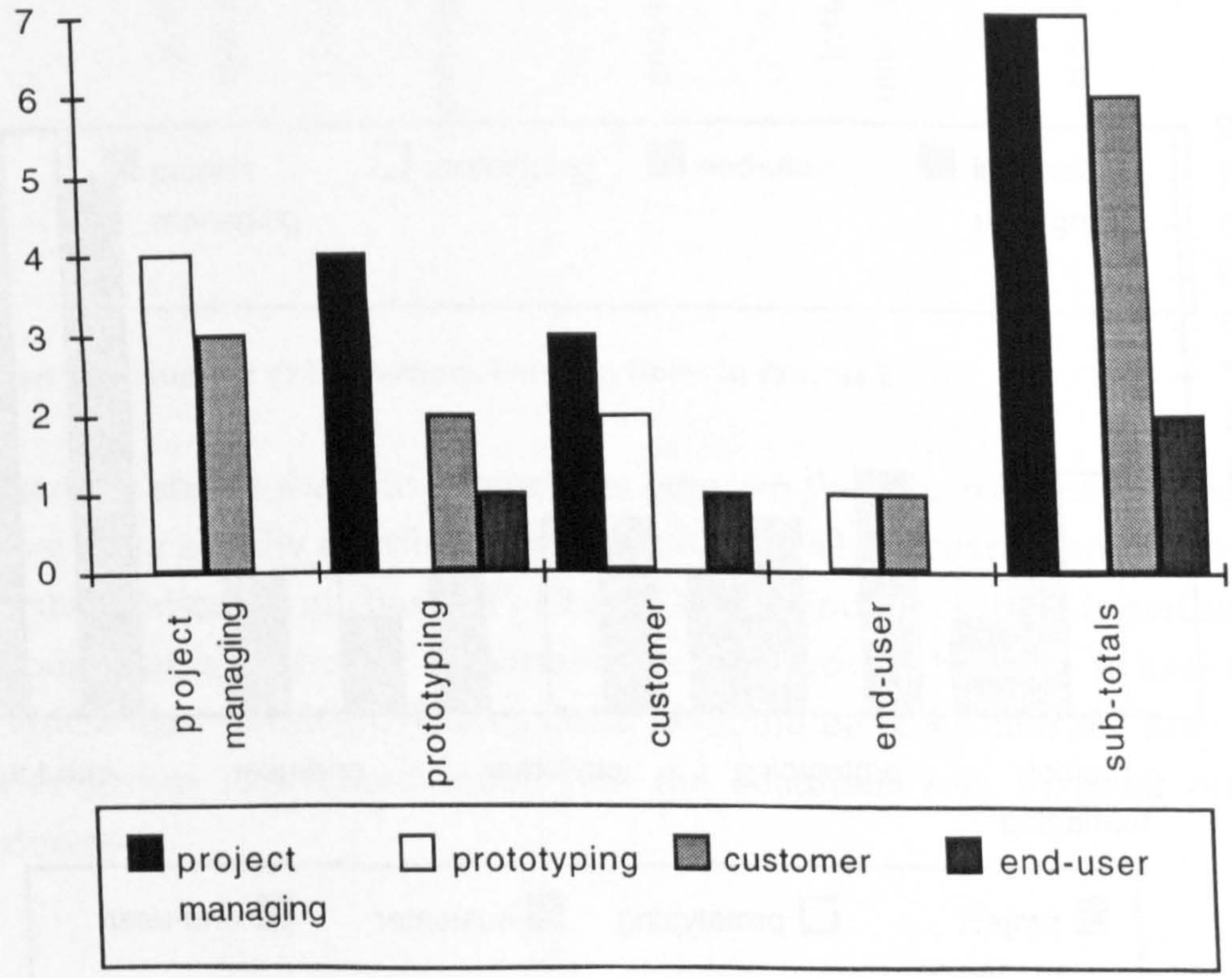


Chart 5.6′ Number of Interactions Between Roles in Process 4

The above charts 5.6 and 5.6′ indicate that project managing and prototyping roles had the same total number of interactions, both prototyping and customer role had interactions with the other three roles, and the end-user role appeared to be the least involved.

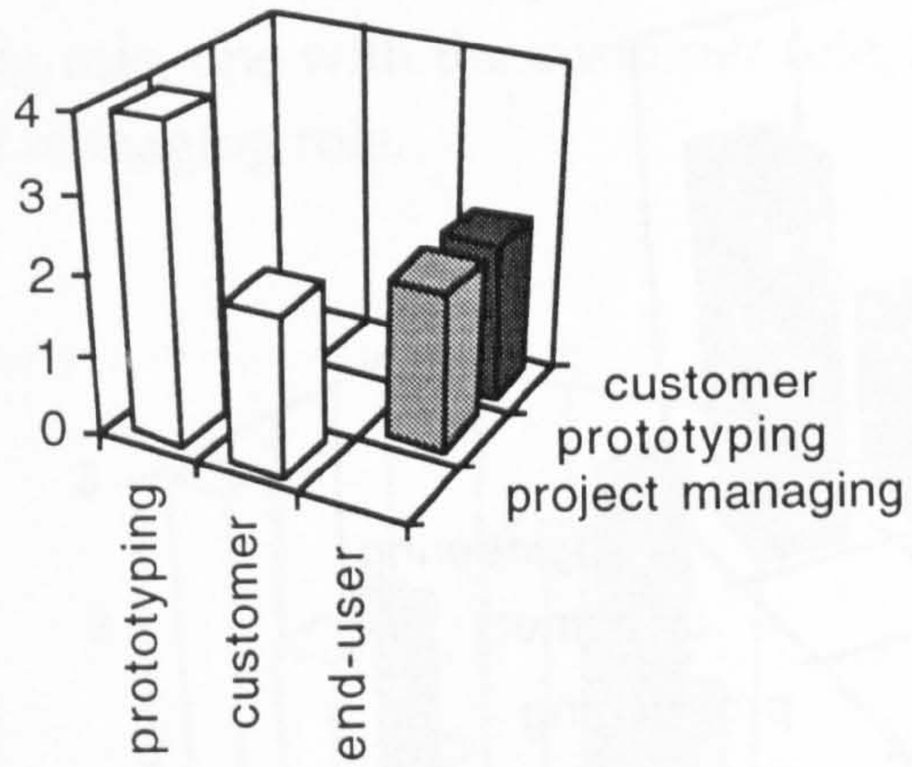


Chart 5.7 Role Interactions of Process 5

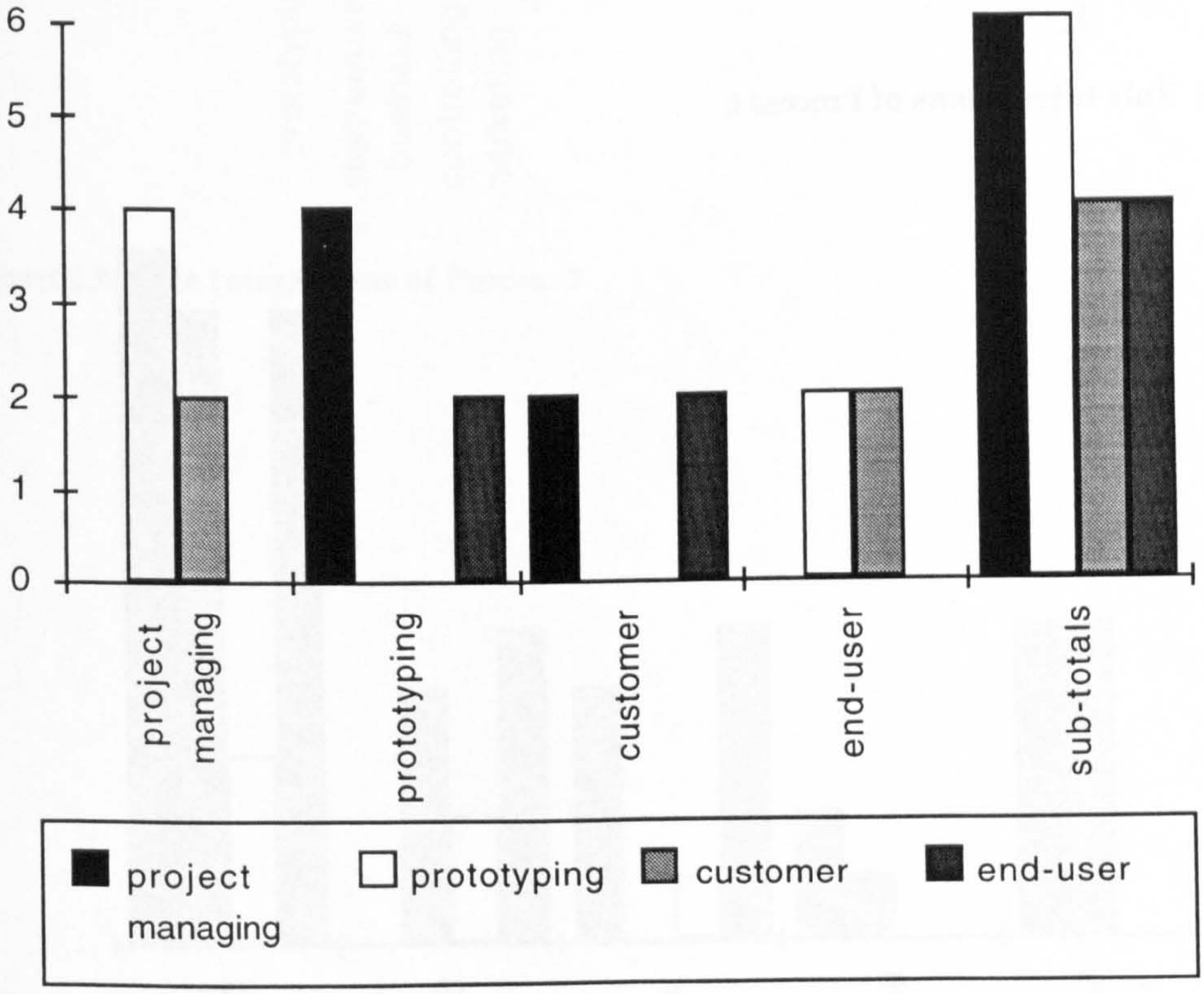


Chart 5.7′ Number of Interactions Between Roles in Process 5

Charts 5.7 and 5.7′ show that each role in process 5 interacted with the other two roles (2 out of three), and managing and prototyping had higher levels of interaction than the other two roles.

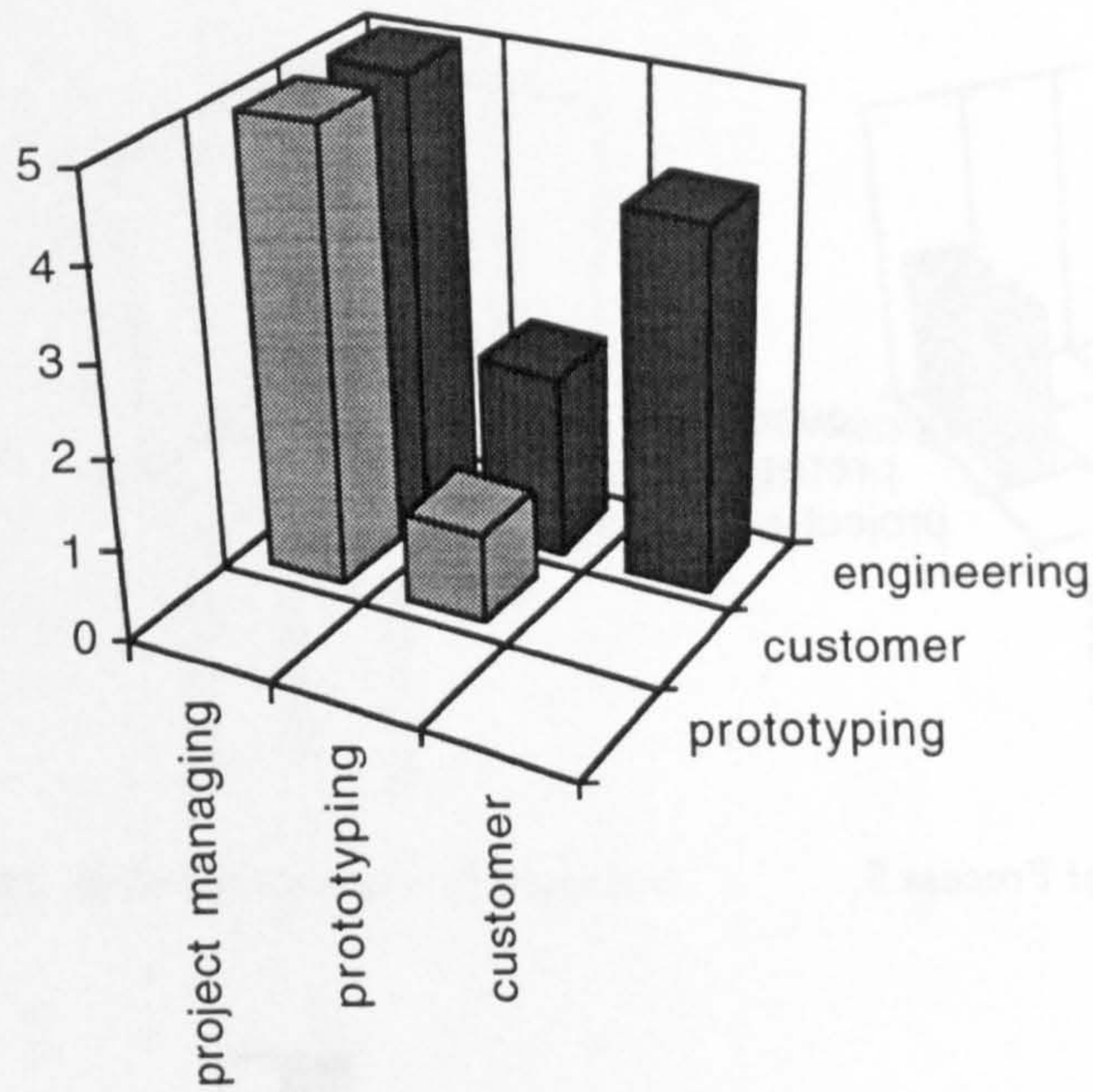


Chart 5.8 Role Interactions of Process 6

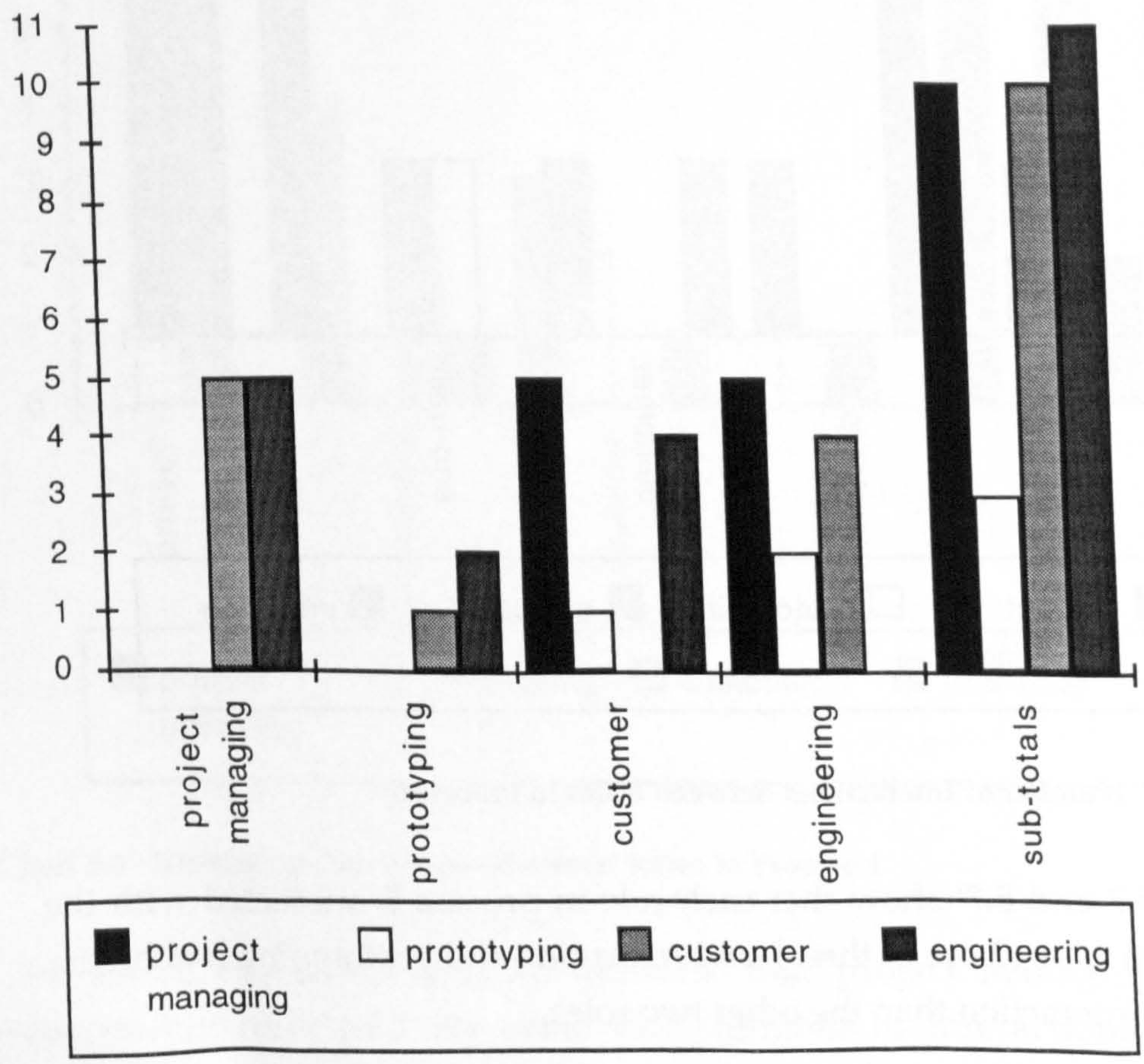


Chart 5.8′ Number of Interactions Between roles In Process 6

It is interesting to note, in process 6, that prototyping played a less significant role with the other processes: it had two interactions with the engineering role, one with the customer role, and had no interaction with the project managing role.

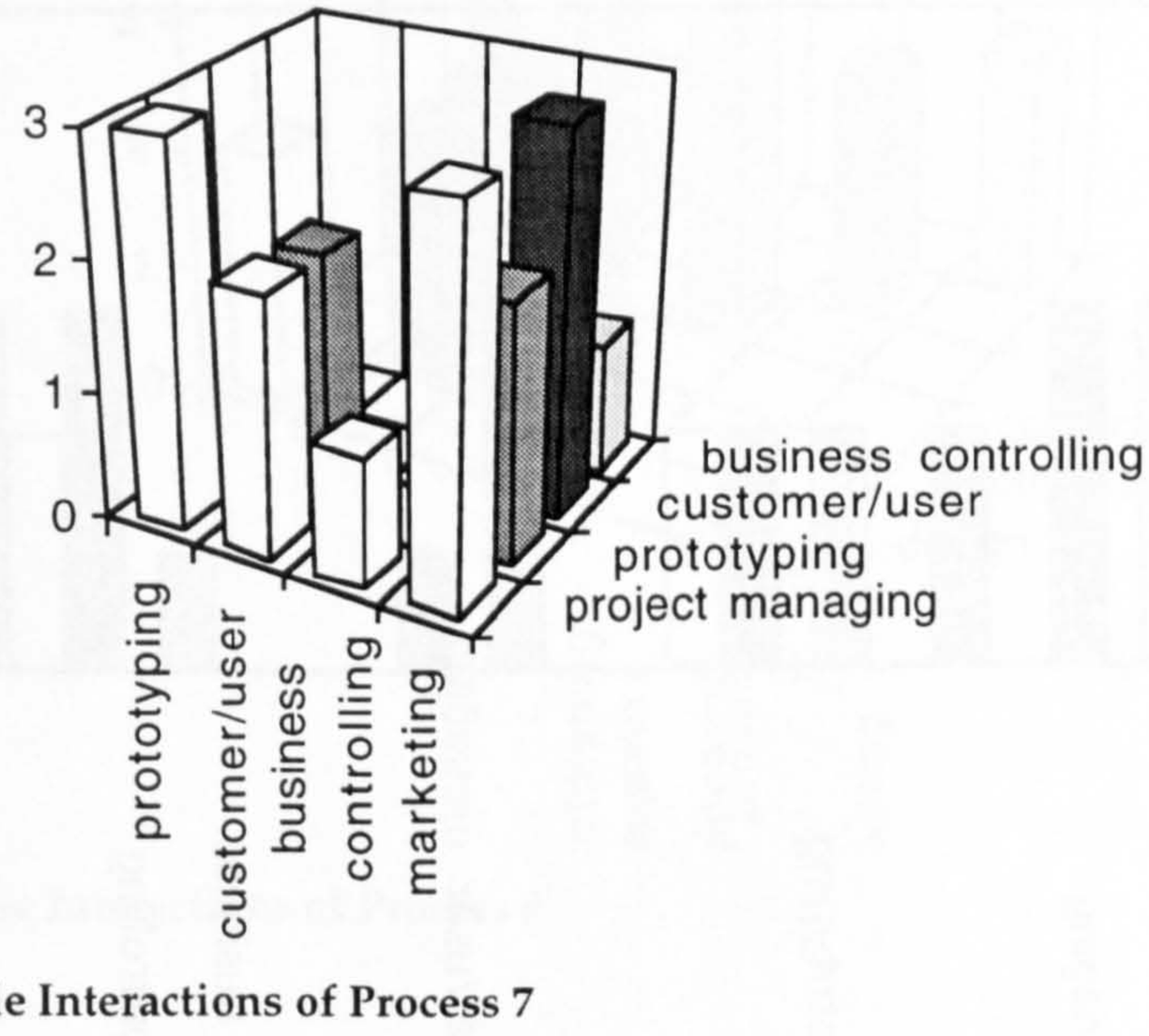


Chart 5.9 Role Interactions of Process 7

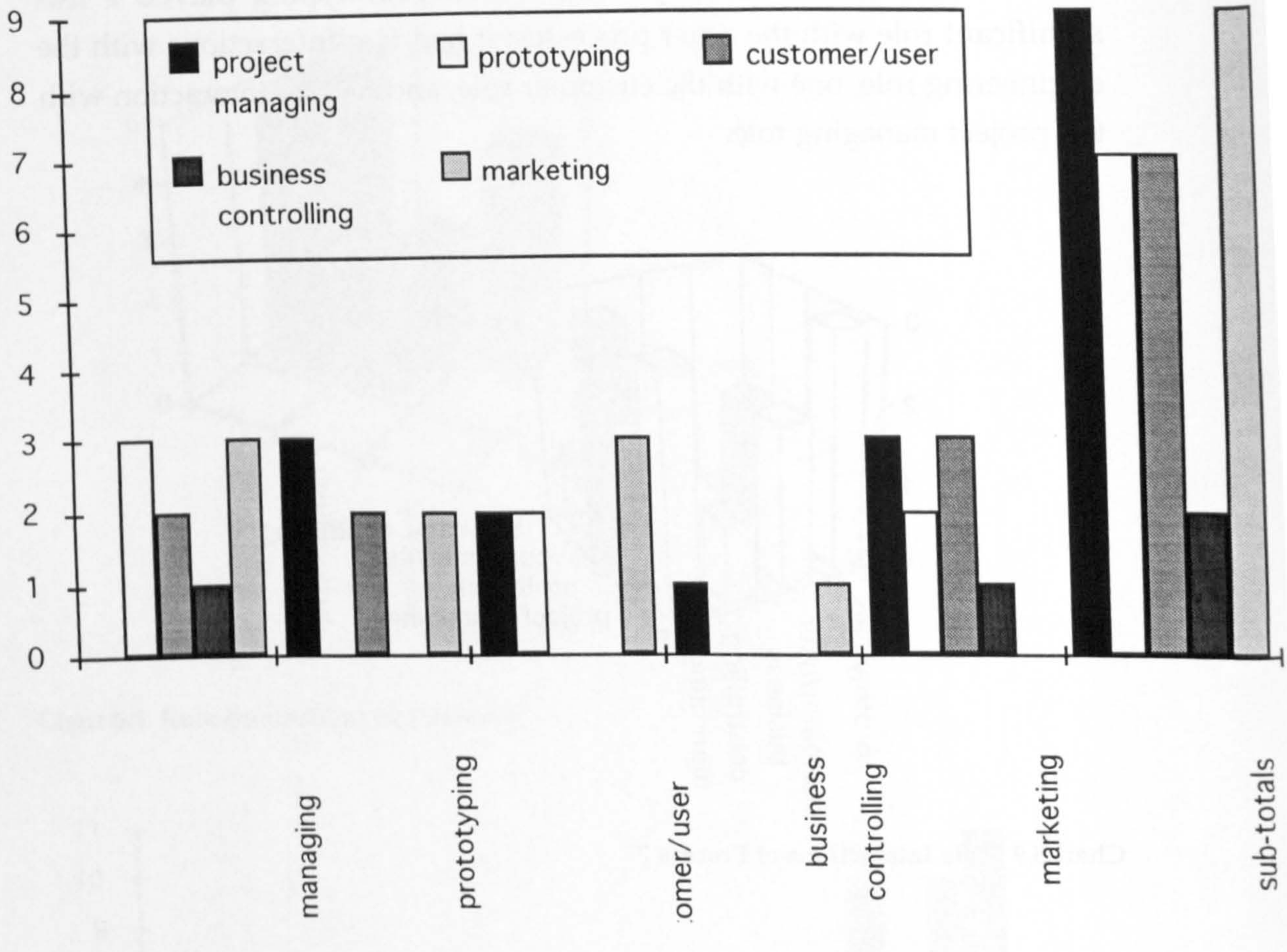


Chart 5.9′ Number of Interactions Between Roles in Process 7

Here in process 7 (charts 5.9 and 5.9′), project managing and marketing were the most interacted roles: each interacted with all the other four roles. The prototyping role had a fair level of interactions with managing (3 times), marketing (2 times) and customer/user (2 times). The business controlling role had only two interactions with marketing and project managing roles at the beginning.

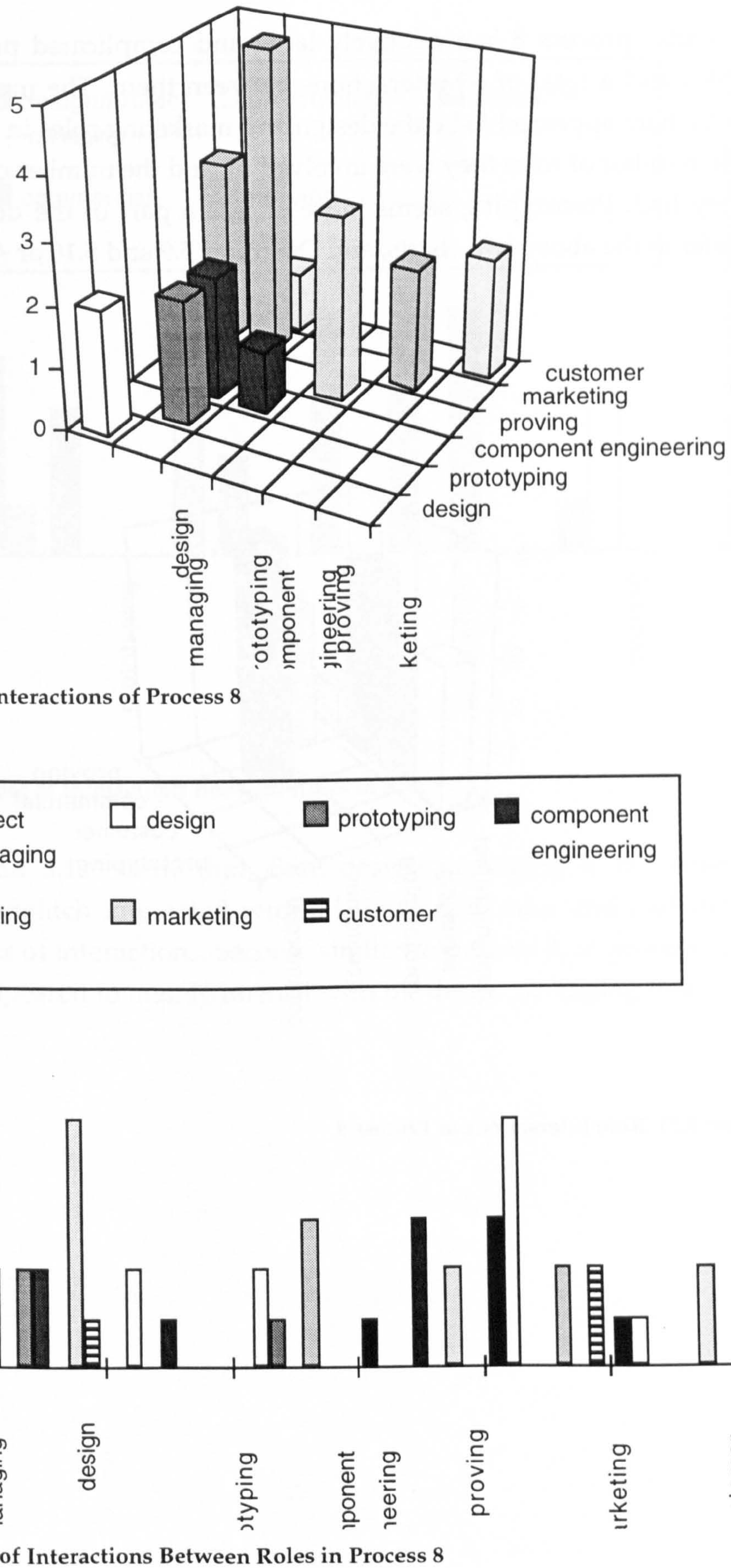


Chart 5.10 Role Interactions of Process 8

Chart 5.10′ Number of Interactions Between Roles in Process 8

Clearly, process 8 is a relatively large and complicated process with 7 roles and a total of 63 interactions between them. The most significant roles here appeared to be the design and marketing roles in terms of both the number of roles they were involved in, and the number of interactions they had. Prototyping seems more or less a part of the design activity (refer to the above two charts, and Diagrams 5.9 and 5.10 or Appendix D).

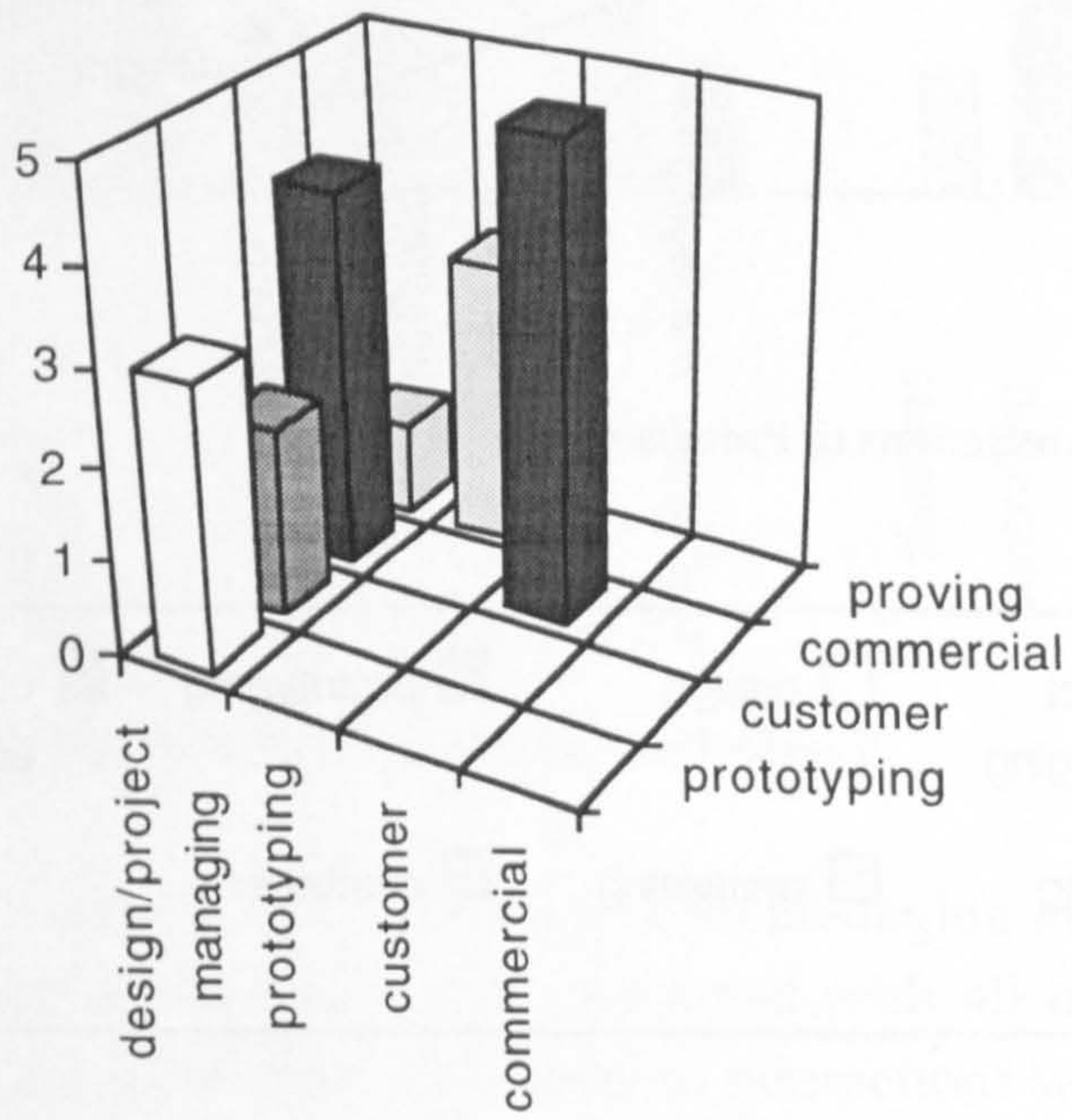


Chart 5.11 Role Interactions of Process 9

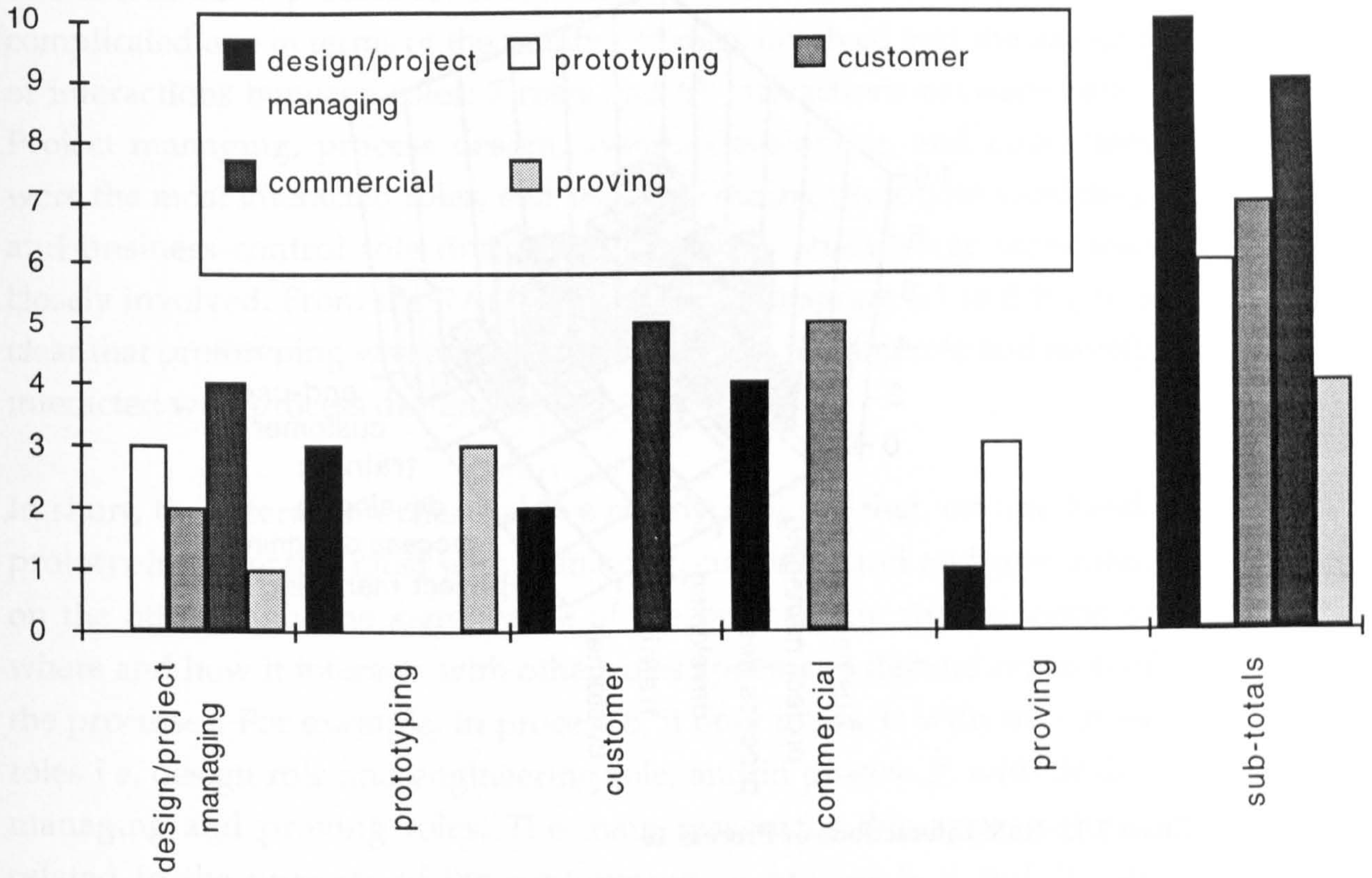


Chart 5.11′ Number of Interactions Between Roles in Process 9

Charts 5.11 and 5.11′ show that, first, design managing is the most dominant role which interacted with all four other roles and had the highest number of interactions. Second, similar to process 8, in process 9, prototyping appeared to mainly interact with the design managing role.

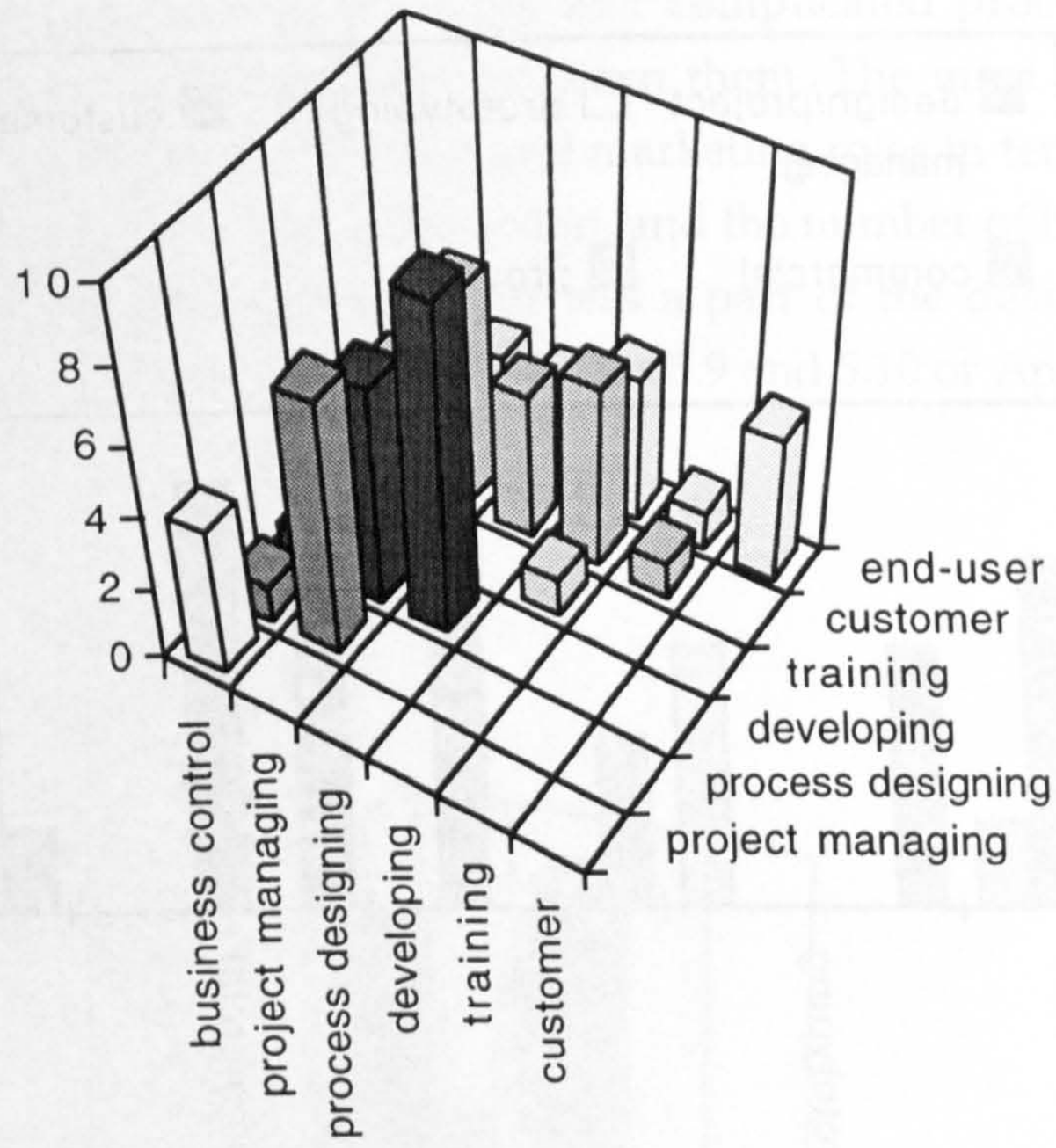


Chart 5.12 Role Interactions of Process 10

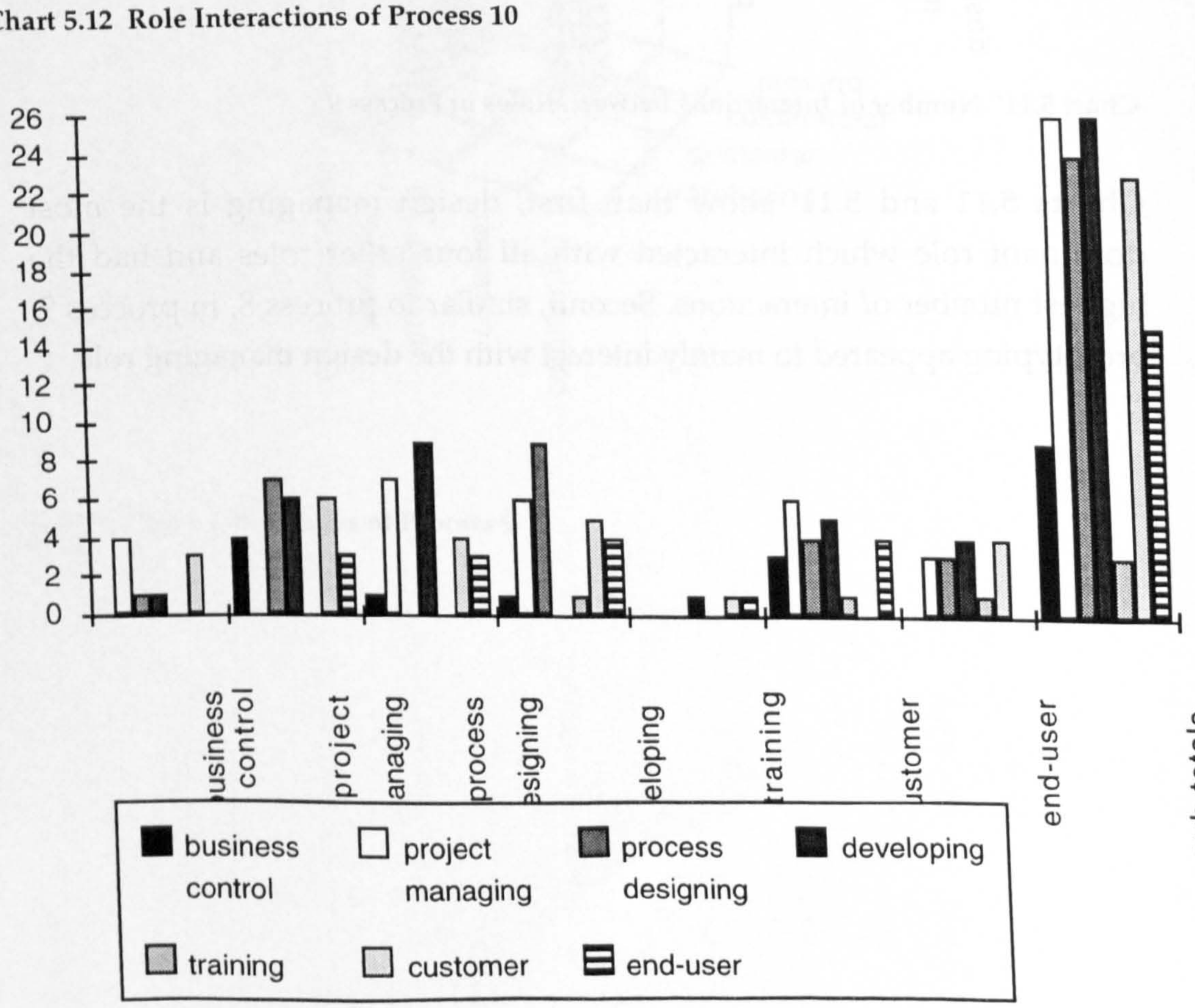


Chart 5.12′ Number of Interactions Between roles In Process 10

The interaction in process 10, as shown in charts 5.12 and 5.12', is the most complicated one in terms of the number of roles involved and the amount of interactions between roles: 7 roles and 63 interactions between roles. Project managing, process design, system developing, and customers were the most interacted roles; end-user role during prototype validating and business control role during the feasibility study stage were also closely involved. From the RADs of process 10 (Diagram 5.1 to 5.10), it is clear that prototyping was only part of system developing role and mostly interacted with process design role.

In short, the interaction charts above clearly indicate that, on one hand, prototyping interacts most with managing, customer and end-user roles; on the other hand, the significance of the prototyping role in terms of where and how it interacts with other roles appears to depend on each of the processes. For example, in process 8, it only interacts with two other roles i.e. design role and engineering role, and in process 9, with design managing and proving roles. The main reason for this appears to be related to the purpose of the prototyping — process 8, 9 and 10 use prototyping mainly for design experiment (Appendix B).

5.5.3 Process Activities

From the RADs of process 1 to 10 above, it is clear that the number of the activities under each process role varied significantly from role to role and process to process. In most cases, private activities (black boxes on the RADs) were only modelled on part of the manager and prototyper. This was due to the fact that: firstly, the managing and prototyping roles were the focus of the study; and secondly, other roles were unable to be included in the interviews. However, public activities or interactions of all process roles were modelled for each process.

The following table 5.1, which was derived from the RADs of the 10 processes (Diagrams 1 - 10 above or Appendix D), summarises the relationships between a number of key activities and their corresponding roles in each process:

processes	key activities	roles involved
process 1	initial requirements gathering	managing, prototyping, customer, end-user
	initial requirements sign-off	managing, customer
	work-break-downs	managing
	high-level /func specs	managing
	prototype building	managing, prototyping
	prototype validation	prototyping, customer, end-user
	CR control	business control, managing, prototyping, customer, end-user
	system acceptance & sign-off	managing, customer
process 2	initial requirements gathering	user group, end-user
	work-break-downs	managing
	high-level /func specs	managing
	prototype building	managing, prototyping
	prototype validation	prototyping, customer, end-user
	requirement specs sign-off	managing, prototyping, user group
	CR control	managing, prototyping
	system acceptance & sign-off	managing, prototyping, user group
process 3	initial requirements gathering	managing, prototyping, customer
	high-level /func specs	managing
	initial requirements sign-off	managing, customer
	prototype building	managing, prototyping
	prototype validation	prototyping, customer, end-user
	sign-off new requirements	managing, customer
	CR control	managing, customer
	system acceptance & sign-off	managing, customer, end-user
process 4	initial requirements gathering	managing, customer
	high-level /func specs	managing
	prototype building	managing, prototyping
	prototype validation	prototyping, customer, end-user
	CR control	managing, customer, end-user
	system acceptance & sign-off	managing, customer
process 5	initial requirements gathering	managing, customer
	high-level /func specs	managing
	prototype building	managing, prototyping
	prototype validation	prototyping, customer, end-user
	CR control	managing, prototyping, customer, end-user
	system acceptance & sign-off	managing, customer
process 6	initial requirements gathering	managing, customer
	work-break-downs	managing, engineering
	high-level /func specs	engineering
	prototype building	engineering, prototyping
	prototype validation	prototyping, engineering, customer

	product sign-off	managing, customer
process 7	initial requirements gathering	marketing, customer
	work-break-downs	managing
	high-level /func specs	prototyping
	prototype building	prototyping
	prototype validation	prototyping, customer and/or end-user
	requirement specs sign-off	managing, marketing
	CR control	n/a
	system acceptance & sign-off	managing, marketing
process 8	initial requirements gathering	design, marketing
	work-break-downs	design
	high-level /func specs	design
	prototype building	design, prototyping, engineering
	prototype validation	design, prototyping
	CR control	design, customer
	system acceptance & sign-off	design, customer
process 9	initial requirements gathering	commercial group, customer
	initial requirements sign-off	design managing, commercial, customer
	work-break-downs	design managing
	prototype building	prototyping
	prototype validation	design managing, prototyping
	requirement specs sign-off	design managing, customer
	CR control	design managing
	system acceptance & sign-off	commercial, customer
process 10	initial requirements gathering	business board, customer
	work-break-downs	managing
	high-level /func specs	business control board
	prototype building	software developing
	prototype validation	process design, developing, customer, end-user
	requirement specs sign-off	managing, process design, developing, customer, end-user
	CR control	managing, process design, developing
	system acceptance & sign-off	managing, process design, developing, customer

Table 5.1 Relationships Between the Key Activities and Process Roles

From the table above as well as Appendix D, it appears that:
(a) there were 8 key activities across the 10 processes:

- initial requirement gathering,
- initial requirements sign off,

- producing work-break-downs,
- producing high level specs/functional specs,
- prototyping building,
- prototype validation,
- requirements/prototype sign off,
- CR controlling,
- system acceptance & final sign off.

(b) while initial requirement gathering, producing high level/functional specifications, prototype building, prototyping validation, CR control, and system acceptance and final project sign off were common to all, activities such as work-break-down, initial and intermediate sign-offs varied from process to process. For example, initial requirements sign-off only happened in processes 1, 3, 9, 10, and work-break-downs appeared in processes 1, 2, 8, 9, 10.

(c) Although all the 10 processes had more or less similar types of interactions on those key activities, the roles and the number of roles involved in each interaction might be very different from process to process. For example, initial requirement gathering performed mainly by marketing and customer in processes 7 and 8, by commercial and customer in process 9, user group and customer in process 2, by managing and customer in processes 4, 5, 6, and by managing, customer and end-user in processes 1, 3. Similar differences existed in other key activities.

This section has simply pointed out the main characteristics in terms of process roles, activities and interactions. The following section will further look into some of the reasons behind them from the information gathered thus far, and their practical implications.

5.6 Lessons learnt

There are several lessons learnt from the field modelling, which are summarised as follows:

1) Process Diversity

The overall impression about software prototyping here is its diversity. Given the comparison made in section 5.4 above, the 10 processes clearly exhibit much more dissimilarities than similarities in term of what role and the number of roles involved, the number of role interactions and how they were involved. Such differences even existed in two development teams within the same company (e.g. process 8 and 9). This appear to be mainly linked with:

a) application domain;

b) size of projects;

c) company infrastructure and/or control culture .

This suggests that a generic process model for prototyping is unlikely to be practically useful to various application domains and vastly different business processes.

2) Customer and User Involvement

The discussion above in section 5.4 clearly demonstrates that the overall involvement of customer and user during the process seems to be much greater (than it would be in conventional structured development) in terms of the number of the activities in which they participated. At the same time, noticeably processes 6, 7, 8, 9 had no end-user role, and for processes 4, 5, 10, end-users were mainly involved in functional prototype validation, and appeared to have no involvement in the initial requirement gathering stage. This suggests that there is plenty of room for more end-user involvement, so that better understanding of user requirements can be achieved at an earlier stage.

3) DBA Role

Apart from process 1, no other processes defined explicitly a DBA role. The background information of the 10 process clearly shows that 8 out 10 processes were for information or database systems development (Appendix B), and yet only one process had a DBA role. It is therefore important, on one hand, to find out the reasons behind such a practice,

and on the other hand to raise alarm about potential maintenance problems that may be caused by neglecting the importance of such a role.

4) Frequent Role Switching

The 10 RADs processes demonstrate (1) frequent role switching and (2) intensive interactions involving many roles. Although one obvious benefit of this is to increase the speed, it may cause problems due to the likelihood of unclear responsibility, especially at the role switch points.

5) Two Levels of Iteration: Inner vs. Outer Loop

Referring to the 10 RADs above (or Appendix D), it is interesting to note that there are two loops within the process models: one is concerned with the first prototype(s) build involving few interaction, the other is more concerned with the evolution of the prototype(s) involving many interactions with different roles, as shown in figure 5.1 below. This indicates two levels of complexity, so looking at the characteristics and the relationships between the two loops may provide a better understanding of the nature of the process, hence improving the management and control of the process.

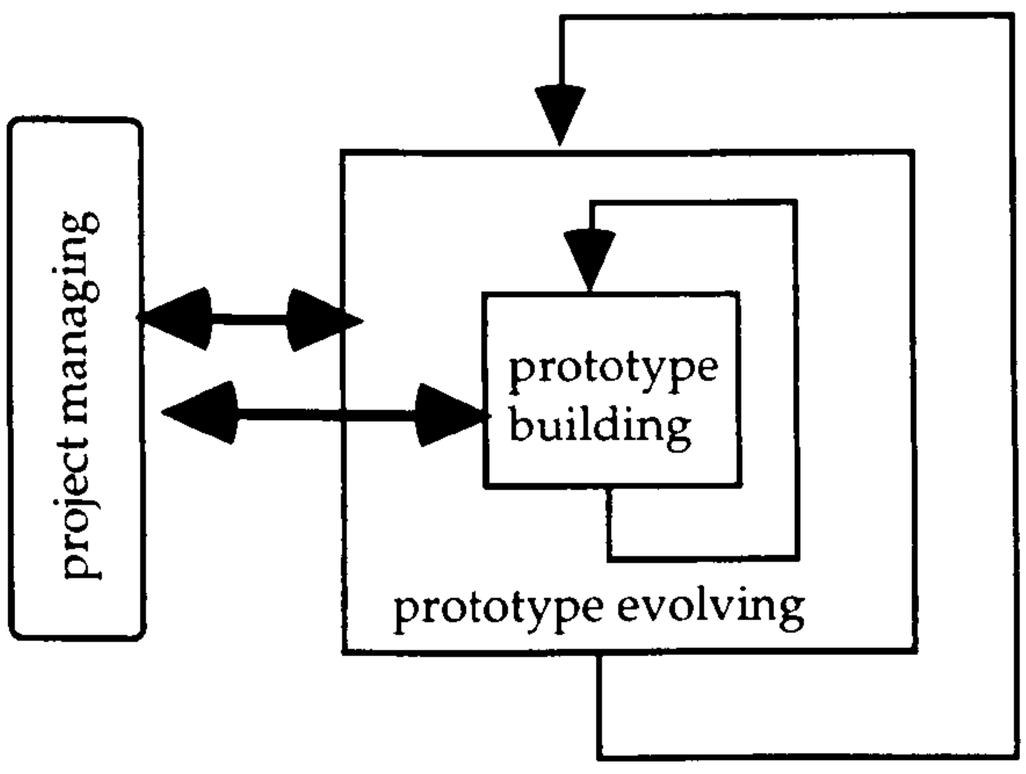


Figure 5.1 Intra and Extra Loops for Managing Software Prototyping

6) Project Sign-offs

Sign-offs appears to be frequently used as a mean of control for

prototyping projects, but probably to a lesser extent than is used in waterfall development where requirements need to be signed off before proceeding to next stage. For example, apart from processes 1, 3, 9, 10, the other 6 processes had no sign-off until after functional prototype building. The apparent reason may well be the recognition that early sign-offs would hinder customer's and user's active participation, and consequently result in poorly understood user requirements.

7) Organisation Infrastructure

To a large extent, these 10 process models seems to depend upon the organisation's infrastructure, which is visible for the 10 RADs and the process background information table [Appendix B]. For instance, process 8 is derived from modelling a software development team within a large telecommunication company, it clearly reflects the large and complicated company infrastructure.

8) Throwaways are Hardly Used in Practice

Apart from process 8, all other processes used their prototypes as part or as a basis for further development. This confirms the finding of the literature review, that is, throwaways are considered as either a waste of time or as luxuries, and many sophisticated development tools/environments also make throwaways often unnecessary.

9) Prototyping Classification

Having discussed about the process diversity in 1), there do exist some similarities in some respects. For example, most of the 10 processes had four similar roles: project managing, prototyping, customer and end-user; most processes (see 5.4.3) had a number of similar key activities and/or interactions; processes 8 and 9 bear some similarities because of similar company infrastructure and application domain. All these suggest that it may be useful to have a class of models that combines several criteria such as application domain, purpose of prototyping and size of system to

build. This may be more appropriate than most referred models such as throwaway and evolutionary prototyping which seem to only take into account types of prototyping.

5.7 Conclusions and Areas for Further Study

Thus far, despite the limitations pointed out in 5.3, field modelling has achieved its main objectives: it has not only provided more insights about the process, but also a framework and areas for further investigation.

1) More Detailed Knowledge of Each Individual Process

Based on the diversity of the processes modelled, more information about each individual process needs to be obtained to better understand the process. For example, questions need to be asked about why things happened under particular environments for each process.

2) Key Management and Control Activities

Activities including initial requirements gathering, prototyping task delegating, prototypes construction, sign-offs and CRs decision making needs to be looked into because, as discussed in 5.4 and 5.5, these appeared to be highly interactive and crucial to the management and control of the process. Another particularly important area, although not explicitly shown in the RADs of the 10 processes modelled, is the prototype team building, i.e. what are the key criteria for selecting prototyper and other project members?

3) Customer and End-User Perspective

As mentioned in section 2, it was not possible to include the customer/end-user's view in the processes, so it would be an advantage if this was included in next stage.

4) Combining the Quality Issues

To clarify those key quality issues (e.g. how quality standards are used and what is the relationship between the standards and product quality) is certainly important and worth scrutiny.

To answer the issues raised by field modelling, further investigations were carried out by conducting more detailed interviews within 5 organisations, which will be detailed in next chapter.

Chapter 6 Further Investigations and Data Analysis

SYNOPSIS Further Investigation and Data Analysis covers the final stage of the investigation, which aims to seek answers to the questions generated from the field modelling, i.e., to find out and reason about the way people conduct the prototyping or RAD projects. The primary method for further investigation was semi-structured interview. In depth knowledge about the key management and control of the process has been obtained by conducting further interviews with a number of managers and prototypers from five companies. Apart from many other interesting lessons learnt, such as prototypers' personality traits and managers' attitudes, the most important finding is the need for adequate methods and standards that are flexible to use and able to cope with fast business changes.

6.1 Introduction

In previous chapters (chapter 4 and chapter 5) the discussion has been centred on answering where, what and how software prototyping has been carried out in industrial or commercial environments. This chapter looks into those particular management and control areas identified from the field modelling. The discussion is based mainly on further interviews with 15 prototyping participants from five companies. It first describes and explains the methods used for the data analysis, and then discusses the data collected. Finally, it recapitulates the lessons learnt and points out possible directions for future studies.

6.2 Data Organisation and Method for Analysis

To make the task of analysing data which draw from multi-source evidence easier and clearer, data need to be properly organised. This section briefly describes the data organisation, the methods for data collection and data analysis.

6.2.1 RADs and the RADs models

RADs — Role Activity Diagrams — have been used as a tool for the field modelling. As discussed in Chapter 5 (5.2.2), RADs gives a clear, straight-forward view of a process in terms of the roles involved, the activities carried out under each role, and the interactions between roles. It is therefore well suited the purpose of the study, which is to investigate the management and control of the software prototyping process.

Ten RADs models, especially those five — company A to E [Appendix D] — upon which further investigation was conducted, are used for the analysis. These RADs models were the direct result of the field modelling and they form the basis for the further investigation and data analysis [Appendix B].

6.2.2 Scale, Scope and Limitations

This analysis is largely based on the interviews with 15 participants from five companies. Most of the interviewees were managers and developers or prototypers. Other organisations or individuals and their process models are also referred to when needed.

The scope of the analysis is from project initiation to project completion as well as some of the organisational and cultural aspects such as infrastructure, control culture, managerial attitudes and personality traits. The focus of the study is on the management and control of the process, especially from an ethnological viewpoint. The technical aspects such as prototyping tools and prototype building and testing etc. are not covered

in detail because of the limited time scale and resource.

6.2.3 The Interview Recording

1) Interview Template

In order to record the interviews and manage the data collection, an interview template (see Appendix E(a)) was designed and has been used throughout the investigation. All 16 interviews of the five further investigated companies are recorded using the same template. Sections on the interview forms are left blank if no applicable information available. Here is an examples of how the index for the interview forms work. Given reference [Form A3: 4.1: a: RM], it refers to the answer given by RM (a reference for an interviewee) to question a in section 4.1 of Form A3.

2) Interview Question Guidelines

This is general guidance for conducting the further investigation. It is designed to reflect the management and control issues raised during the field modelling, and also to make the data collection result more comparable and easier to analyse (see the Appendix E(c)).

6.3 Data Analysis

Our main discussion progresses along four management and control areas of the process:

- project initiation,
- initial requirements gathering,
- pilot or first prototype(s) building,
- user participation and CRs control.

In addition, the discussions are extended to some other issues such as organisation culture and individual managerial style.

The following discussions first draw the boundary of the area under scrutiny and point out the key management and control factors or issues, which came from either the process participants' concern or the authors observation. It then briefly describes their practice and reflects on their view of these key issues. Furthermore, comparison is made on their similarities and differences.

6.3.1 Project Initiation

Here project initiation is the period from project go-ahead to actual project start. The main concerns for this area are:

- how is a prototyping project team made up?
- how and why is each individual team member chosen?
- are any methods or guidelines used for prototyping?
- if yes, do they include clearly defined quality standards and/or checks, and responsibilities for managers and prototypers? if not, why?

The following sections (6.3.1.1 — 6.3.1.1) answer each of those questions in turn.

6.3.1.1 Project Team Makeup

Project team is defined as only those who are directly involved in a prototyping project from the company that undertakes the development. The focus here is team structure and team member selection criteria.

1) Team Structure

Before looking at the prototyping team structure of the companies, it is helpful to first look at that of conventional systems development.
Figure 6.1 shows the team structure for a conventional structured development:

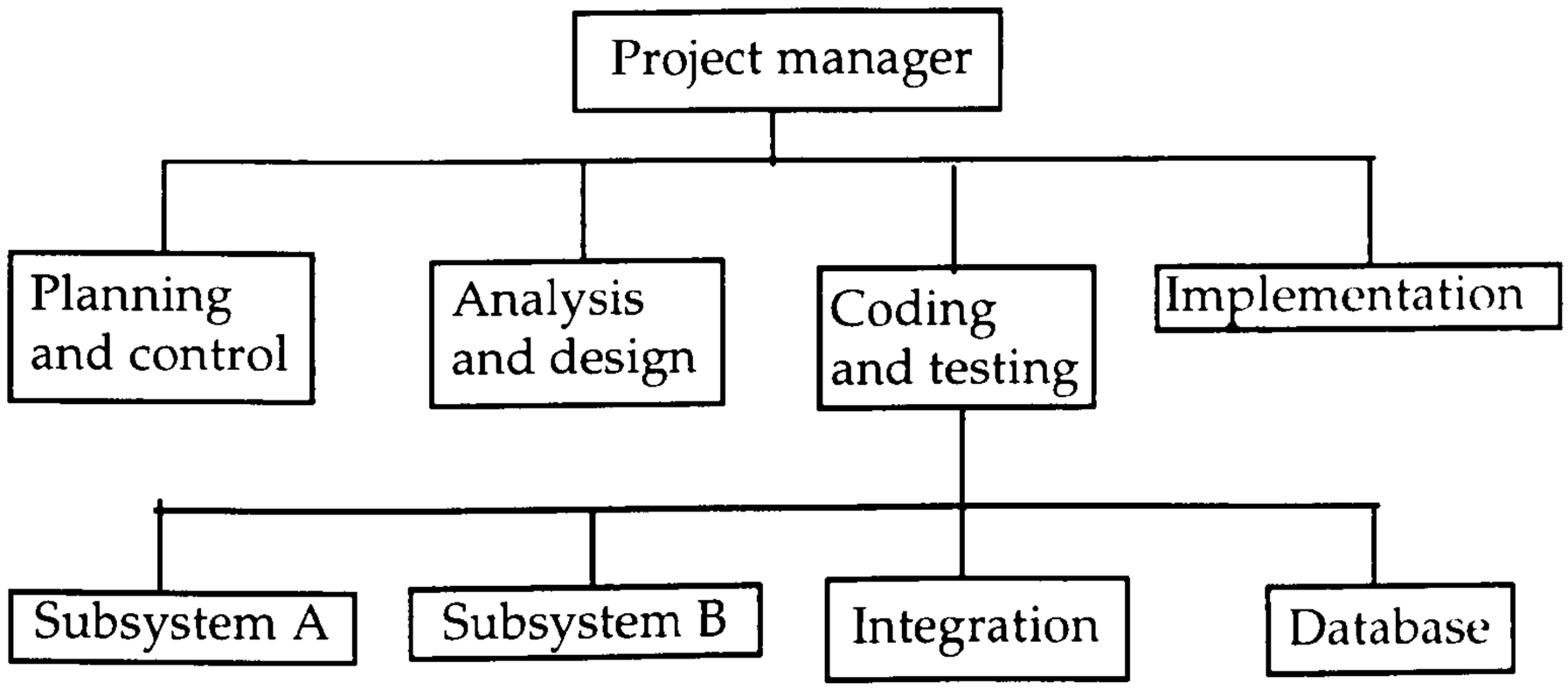


Figure 6.1 Conventional Work Breakdown Structure[8]

And the organisation for a small project is typically like this:

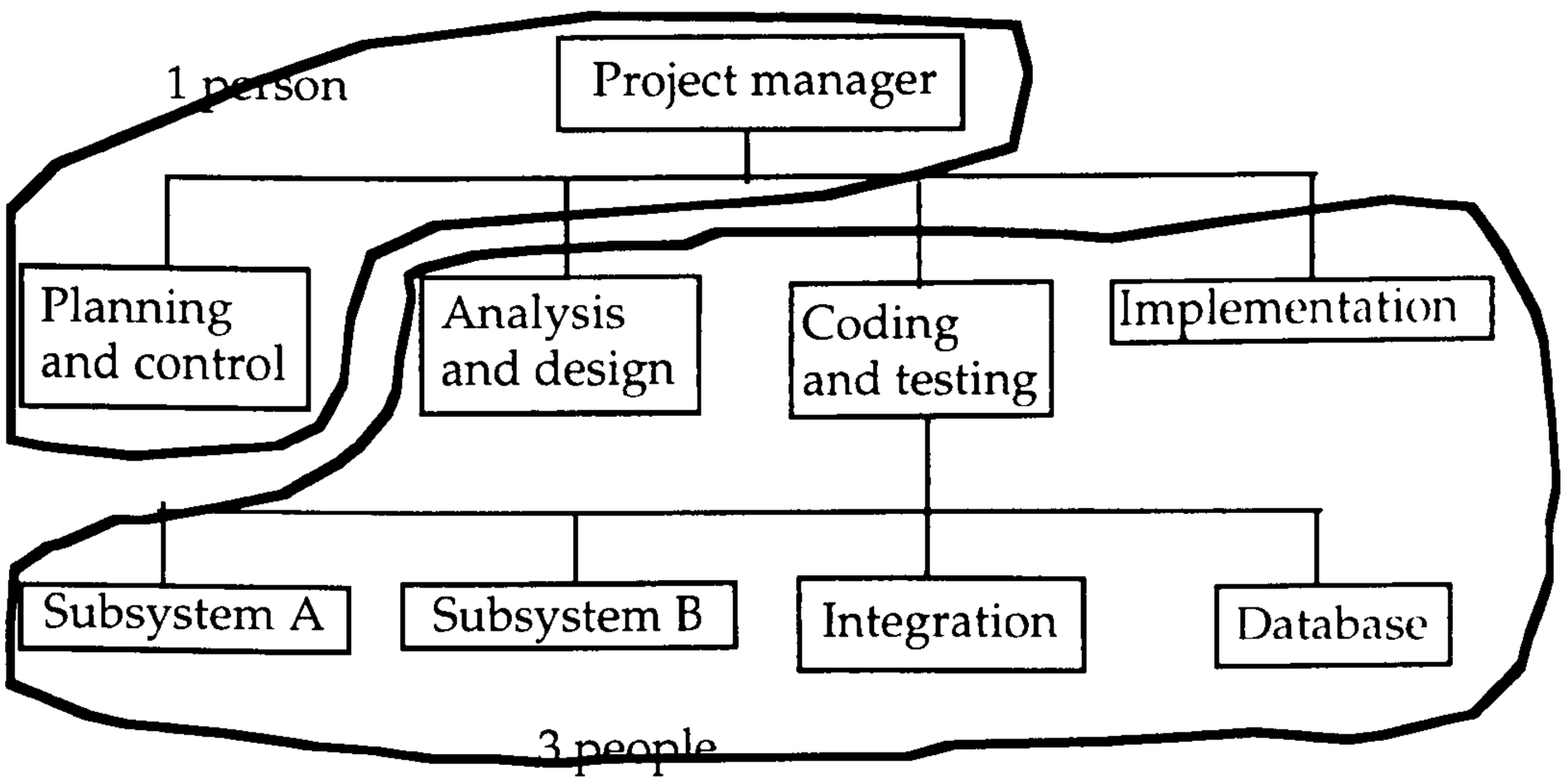


Figure 6.2 The Team Organisation for a Small Project[9]

It is clear that there are basically two separate roles here: manager and developer. Managers take over the planning and controlling tasks and developers take responsibility for the rest of the tasks.

The organisation changes greatly when the size of a project increases as indicated in the following figure 6.3:

---

[8]from section 32 of McMermid's Software Engineer's Reference Book (McDermid 1991).
[9]same as 1 above.

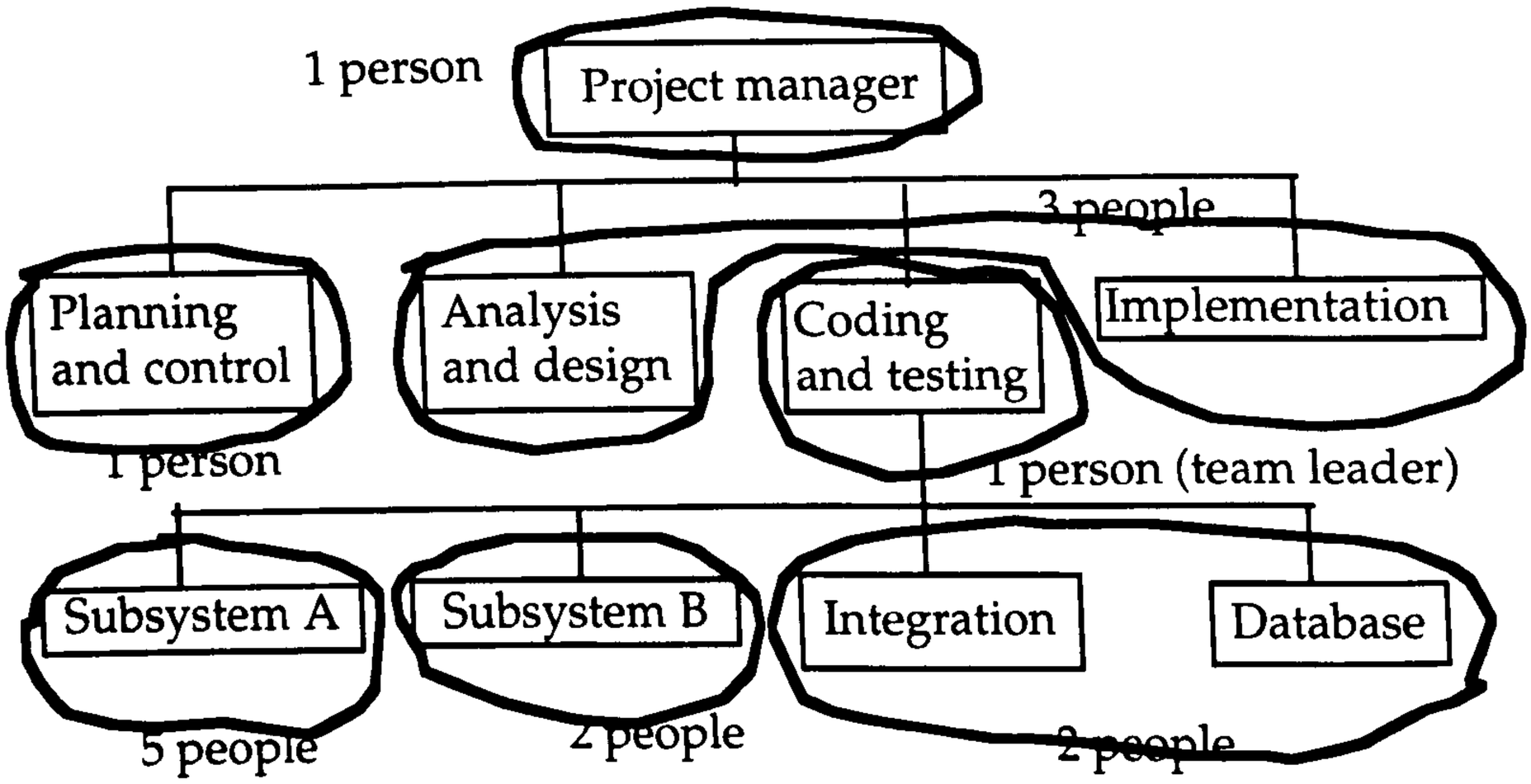


Figure 6.3 The Team Organisation for Large Project[10]

Clearly the team organisation for a large conventional project is much more centred around its work breakdown structure, in other words, the roles are much more tied in with the activities shown in the work breakdowns.

Now let us look at the typical practice from each of the five companies.

Table 6.1 below is a brief summary about the project initiation of the five companies further investigated, which is derived from Appendix E(b).

	A	B	C	D	E
a) team members?	1 senior manager, 1 or 2 prototypers	1 senior project manager, 1 design manager, 3 prototypers	no prototyping team as such, it is just same as the conventional team structure	1 senior project manager, 1 application manager, 1 or 2 prototypers	1 senior project manager, 1 design manager, 3 prototypers
b) prototyper selection criteria?	experience,; domain knowledge; communication skills; career development.	experience; domain knowledge; communication skills,	N/A	experience; communication skills,	experience; domain knowledge; communication skills,
c) procedures or guidelines?	no	no	N/A	no	no
d) role responsibilities defined?	not clearly	not clearly	yes but not fully	not clearly	clearly but not fully

[10]same as 1 above.

e) quality goals defined?	not clearly	not clearly	yes	not clearly	clearly but not fully
f) RAD methods available?	not yet but developing	none, but developing	none	none, but thinking about developing it	yes, newly developed

Table 6.1 Project Initiation

From a) of the table 6.1 above and the interview summaries [Appendix E(b)], we have the following figures illustrating their typical team structure for prototyping project:

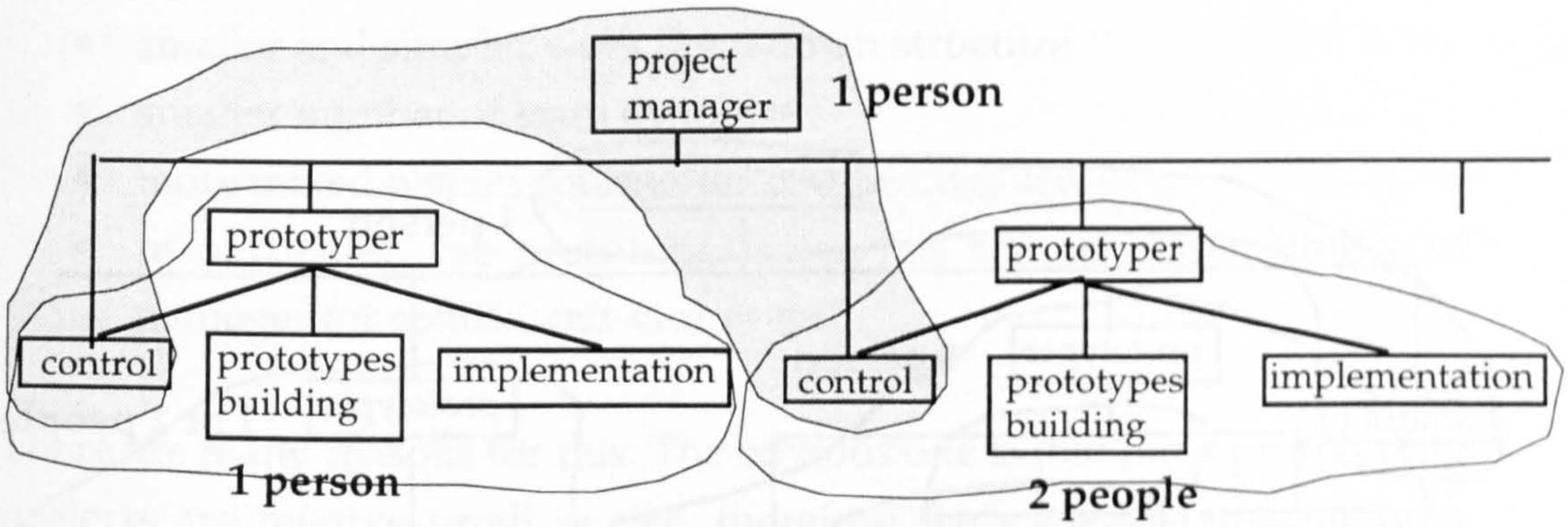


Figure 6.4 Team Structure of Company A (process 2 in Appendix B)

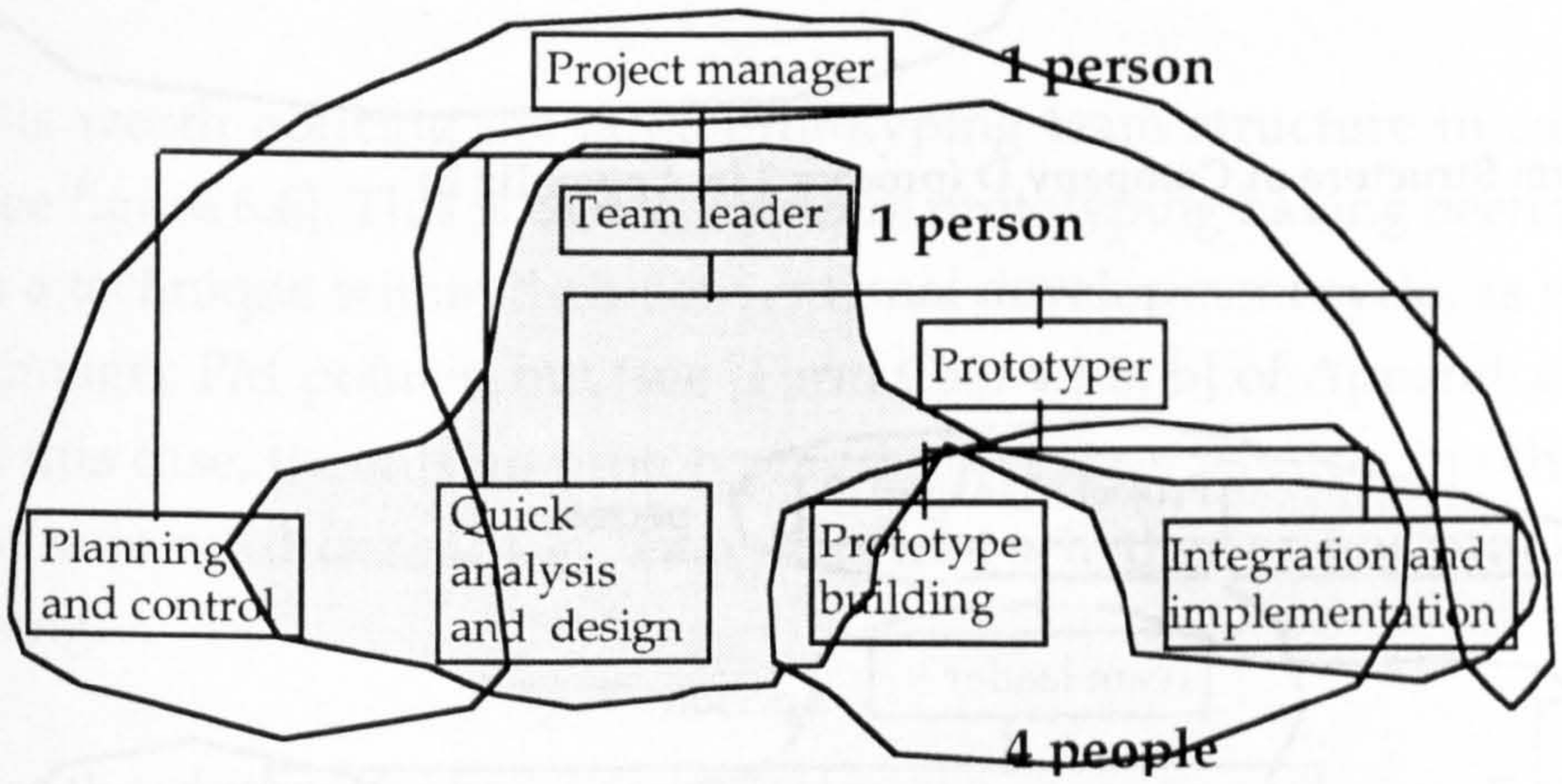


Figure 6.5 Team Structure of Company B (process 1 in Appendix B)

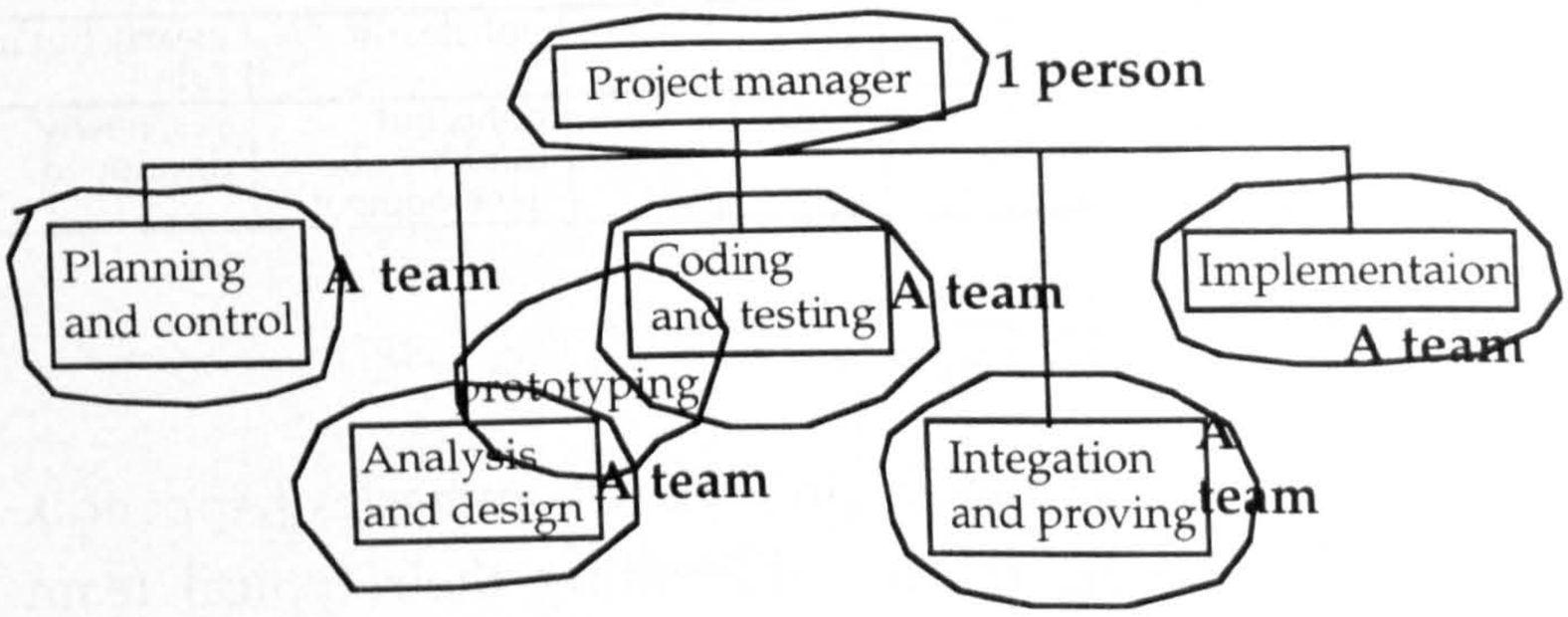


Figure 6.6 Team Structure of Company C (process 8 and 9 in Appendix B)

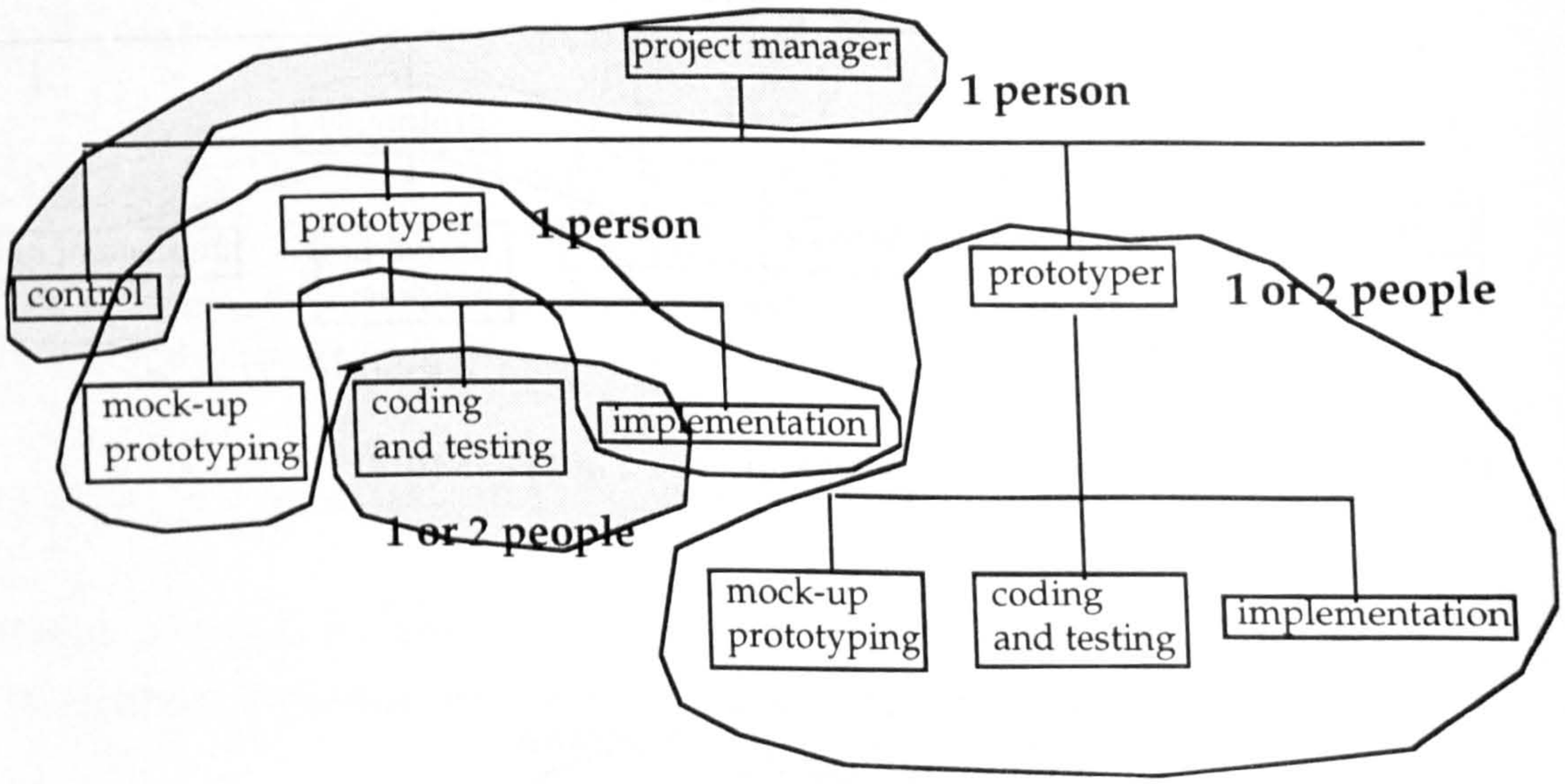


Figure 6.7 Team Structure of Company D (process 3 in Appendix B)

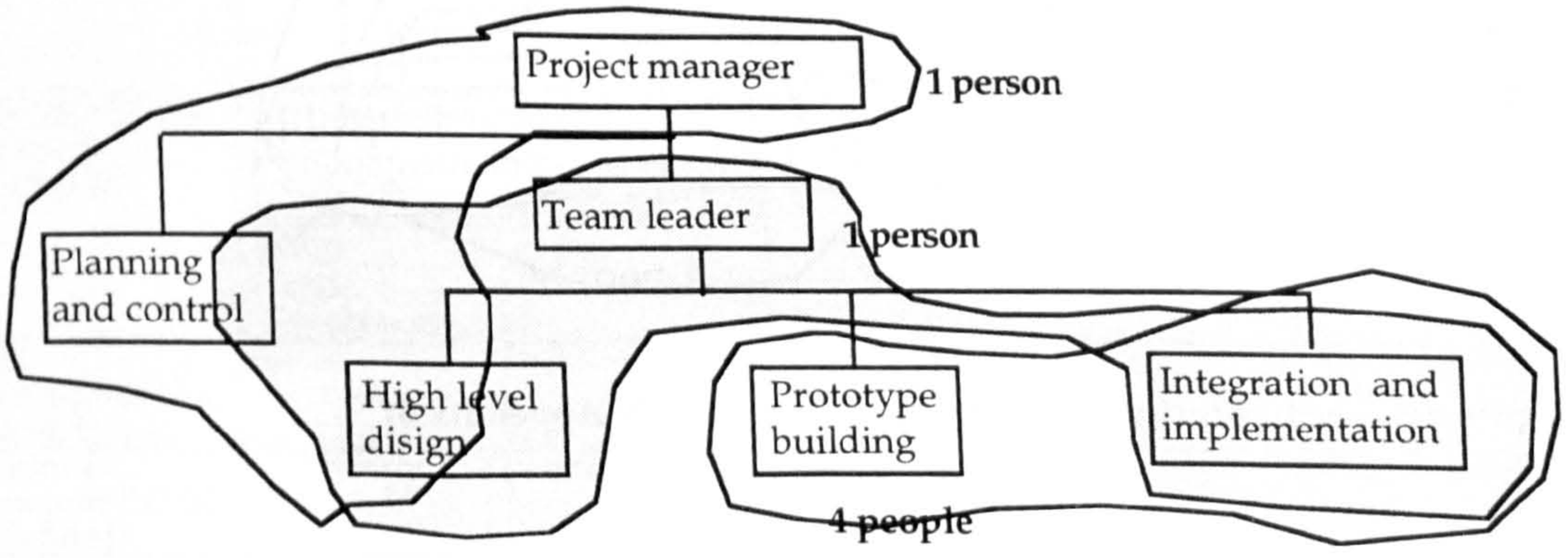


Figure 6.8 Team Structure of Company E (process 10 in Appendix B)

By comparing the typical conventional and prototyping practice as

illustrated above, it is clear that:

a) a conventional project typically has

- larger and more detailed work breakdown structure
- larger number of team members
- clearer responsibilities for each person involved, even for small projects;

b) at the same time, the prototyping project often appeared to have

- smaller and simpler work breakdown structure
- smaller number of team members
- more mixed responsibilities for people involved
- in particular, the prototyper often has shared responsibility of manager for control and vice versa.

There are many reasons for this. The obvious one is that most prototyping projects are relative small in size, therefore there are less management tasks. However, the more important reasons appeared to be the desire for simple communication, less time and cost, more flexible control and increasing prototypers' enthusiasm [see section 4.1.3 of interview forms A2-3, B2-3, and D2-3 of the Appendix E(b)].

It is worth noticing the large prototyping team structure in company C [see figure 6.6]. This is due to software prototyping having been used only as a technique within their conventional development cycle, as the design manager PM pointed out (see [Form Ca3: 4.1.3: b] of Appendix E(b)). So, in this case, there is no prototyping project team as such, in other words, there is no difference in team structure whether or not prototyping is using.

Another interesting case is the team structure in company E. It had six members in a prototyping project team: two managers and four developers. This is larger than the most companies' team size, which is between two to four. This was due to:

- the prototyping project undertaken was a large project — six people

working for about one year time, while the size of projects in most other companies were smaller;

• it was a project undertaken by the IT department of the company, which had much less money and time constraints.

The latter reason seems to have a particular influence not only on team structure but also on the whole project. This issue will be expounded in section 6.3.5.

2) Team Selection Criteria

Looking at the Table 6.1: b), it is clear that although there were no explicit procedures and/or guidelines that had been used for team selection, a set of factors or criteria were widely recognised and applied among the managers in practice. They are:

• **Availability** First of all, the prerequisite for the above criteria is the staff availability, that is to say, there is no point in using criteria if there are no other choices. As the senior project manager KM of company A described, the first thing is to find out all those who are available for a new project, then to apply other criteria to assign the right person for right job (see [form A2: 4.1.3: a] of Appendix E(b)).

On the whole, there are more staff available for selection in large organisations such as company A, company B, and company E, than small one like company D (see section 4.1.3 of form A2, B3, E2 and D3 of Appendix E(b)). In either case, it should also be noted that often there is much less choice for the project manager than for other team members.

• **Career development** Some managers, like KR of company A and DR of company B, have taken the career development of his staff into consideration in team selection. For example, KR is trying to let his developers in the RAD unit work in each different business area for two or three years so that they become more flexible and competent in the future [Appendix E(b): form A2: 4.1.3: a].

Whereas in company B, DR intentionally had a mixture of two experienced developers and three novice developers in the project team. The main reason was that it would give not only the new staff a chance to learn from experienced developers, but also for everybody to gain experience for the new RAD development approach. Therefore, the likely productivity fall for a particular project — owing to not including the best possible staff — would be compensated by long-term benefits such as increased staff mobility and new skills gained for future projects [Appendix E(b): form B3: 4.1.3: r].

• **System development experience** Apparently, systems development experience is one of the essential qualities required for prototyper. When asked, almost all the managers made it their first criteria [Appendix E(b): 4.1.3 of form A2, B3, Ca3 and Cb2, D3, E3]. As indicated in above figures (figure 6.4 - 6.8), prototypers are often involved with most development tasks from requirement gathering, quick design, prototype building to implementation and some managing tasks such as negotiation and sign-off. Therefore, it is very important for prototypers to have substantial hands-on experience of the whole development experience.

• **Application domain knowledge** This seems to have a special importance in RAD development, which was emphasised by many interviewees throughout the investigation [Appendix E(b): 4.1.3 of form A2, B3, Ca3 and Cb2, D3, E3]. For example, the senior project manager DR of company B pointed out that it was crucial to include domain expert in a RAD team, and the prototyping project[11] would not be successful if such a team member like ML, the design manager, were not included [Appendix E(b): Form B3: 4.1.3: r]. And "... they talk business languages with users rather than IT language", as KR, the senior manager of company A, emphasised [Appendix E(b): Form A2: 4.1.3: b].

• **Communication skills and personality** Good communication skills are obviously an advantage and this has been an the important factor in choosing prototypers, which can be seen in [4.1.3 of Form A2, B3, Ca2, D3, E1] of Appendix E(b).

---

[11]this is the first RAD project in his department.

The personality traits do not seem to be considered explicitly in their team selection criteria, though some awareness is visible [Appendix E(b): form A3: 4.1.3]. To further this issue, a personality test was given to most of the interviewees. For more detailed account see the summary report on personality test [Appendix F(a)] or a brief discussion in section 6.3.5.

6.3.1.2 Role and Individual Responsibilities

Apart from team structure and selection criteria, role responsibility is another important issue to discuss during project initiation. The main question to ask is that if roles and individual responsibilities are clearly defined at this stage? If so, what these are and how they are defined? If not, why?

From d) of table 6.1 above, it is clear that the responsibilities of each process role and person were either not clearly or not fully stated, especially those overlapping parts as indicated in the above figures (figure 6.4 to 6.8) in all five companies. They were often assigned loosely, by meeting, e-mail, telephone, or implied in some written form such as in the project proposal [Appendix E(b): form D3: 4.1.3], work breakdowns [Appendix E(b): form Ca3: 4.1.3] and so on.

The reasons for such loose control at this stage appeared to be:

- the emphasis on quickly producing a working version of the system. It was often the case, from a manager's view point [form A2: 4.1.3], that developer's responsibility was to get something useful working quickly;

- the need for flexible control and constant role switching and overlapping. Refer to the RADs models [Appendix D] and the interviews [Appendix E(b)];

- the dislike of filling over-detailed forms and being constantly checked [form A3: 4.1.3];

- RAD was new to them, and the corresponding methods and

procedures were mostly not yet available. See the company information table [Appendix B] and table 6.1: c) and f) above;

- relatively small project size. Refer to the company information table [Appendix B];

Therefore, although most people were aware of the fact that things weren't so clearly defined, they seemed more or less happy about the way things were done, because this often resulted in greater productivity [Appendix E(b): interview form B3: 4.1.3] and user as well as prototyper satisfaction [Appendix E(b): interview form A2, A3: 4.1.3].

However, the lack of clearly defined responsibilities for each of the roles and the participants, such as reporting procedures about CRs, intermediate and end product documentation, the interactions among members of the development team, and the co-operation between all parties involved, may lead to greater maintenance effort. This will be further discussed later in section 6.3.4.

6.3.1.3 Quality Goals and Standards

As identified in first stage of the investigation [chapter 4], product quality was one of the major concerns from practitioners. It was therefore also important to know what were the quality activities and/or checks defined explicitly at the beginning of a project, and how were they practised and why.

From e) of Table 6.1 above and the interviews (Appendix E(b)), the following were observed:

1) Quality goals and QA activities were rarely clearly defined at the beginning of a project. This is evident from the interviews [Appendix E(b): 4.1.3 of form A2, B2&B3, D3]. The main reason here seemed to be that quality was often considered mainly as 'fit to purpose', and the focus was on customer and user satisfaction [Appendix E(b): 4.1.3 of form A2, B2&B3, D3]. As a result, other important aspects of quality such as extendibility, portability and maintainability were often not given

adequate attention, which appeared to have caused a considerable amount of concern among managers and prototypers (see also 4) below).

2) Quality standards and their corresponding QA activities were largely ignored because they were often thought to be either too rigorous or inappropriate to follow or too tedious and time consuming to be practical. This view was expressed by most of the interviewees across different companies [Appendix E(b): 4.1.3 of form A3, B2, Ca3]: firstly most of the standards are aimed at large conventional developments, so were inappropriate for small or even medium sized projects; secondly, QA activities were often felt to be tedious and time-consuming, and conflicted with the speed required by most RAD methods.

3) It appeared that the project naming standards were mostly mandatory, however, the coding and documenting standards were often left to individuals [Appendix E(b): 4.1.3 of form A3, B3, Ca3, D3, E3].

4) One major concern among managers and developers was about the 'unknown quality'[Appendix E(b): 4.1.3 of form A2&A3, B2&B3, D3. 4.1.4 of E1]. Despite this, due to the reasons stated above (section 6.3.1.2), it had not been regarded as or dealt with as an urgent issue [Appendix E(b): 4.1.3 of form A2&A3, B2&B3, D3]. It may be the case, however, that care should be taken to ensure that short term productivity gain is balanced with long term maintenance effort. In other words, unless quality goals are clearly defined and subsequent activities and checks are implemented, some important aspects such as maintainability will remain unknown.

6.3.1.4 Rapid Application Development Methods

The objective here was to find out if any methods had been developed for RAD or software prototyping. If so, what are the implications, and, if not, why not?

The following discusses the findings based on the interviews (refer to Appendix E(b) and f) of table 6.1):

1) Four out of five companies investigated have no formally defined

methods for prototyping or RAD available at the time of the investigation. Firstly, software prototyping, especially RAD methods, is still a relatively new system development approach [see chapter 2], and most companies had only one or two years experience in RAD methods [Appendix B]. Therefore there had not been long enough to develop their own RAD methods.

Secondly, it seemed there was lack of the urgency for formally defined methods. The prototyping projects were mostly small and medium sized and the project teams were normally just two or three person [Appendix B] and they were therefore fairly manageable [Appendix E(b)]. On the whole, both managers and prototypers felt comfortable about their informal communications and controls, and no problems were at the time found to be causing great trouble.

2) One exception was company E. It developed its own prototyping methods into a company wide system development methodology, which was newly developed and this was first time they had been used in practice. Although it was evolved and simplified from their old methodology [Appendix E(b): form E(2): 4.1.5], it still was rather large in volume and rigorous to follow[12].

3) Most of the companies had the intention or were trying to develop methods for their prototyping practice [see Table 6.1 and Appendix E(b)]. Although, as explained in the above, there seemed not to be many problems with their current practice, some managers and prototypers did express concern about the loose control which might cause future maintenance problems [Appendix E(b): form A2&A3, B3, D3]. Some companies had already taken steps towards building practically useful methods and standards: company D was just about to start, company A and B were already on the way, and company E had recently updated their conventional methods and standards to fit and support the use of the prototyping approach [see Table 6.1 and Appendix E(b)].

To summarise, the key lessons learnt about the project initiation stage are:

---

[12] However, their practical implications was not been able fully investigated due to the company withdraw further co-operation due to work pressures.

- lack of clearly defined criteria for team and especially prototyper selection. Although a set of selection criteria were used in their practice [section 6.3.1.1 2)], none of the five companies had them clearly defined;

- systems development experience and domain knowledge appear to be the key ingredients for prototyper selection. In addition, their personality traits seemed to have significant influence on the process [refer to 6.3.5 or Appendix F(a)];

- the lack of clearly defined responsibilities and the potential problems it may cause suggests that more clearly stated individual responsibilities should be made clear at the very beginning, especially if frequent role switching happens;

- apart from the emphasis on developer and user satisfaction, few other quality goals were set. Although this seemed not to have caused problems, more and clearer quality goals need to be defined in order to achieve long term quality objectives such as extendibility and maintainability;

- the lack of suitable methods and standards for using software prototyping, in particular for RAD practice.

6.3.2 Initial Requirements Gathering

The initial requirement gathering is defined here as the period for gathering the essential requirements from which the first working version of the system can be built. As a basis for the rest of the development, this is one of the crucial stages of the whole development process.

The main aims here were to find out:

- the methods or practice used for initial requirement gathering;
- the controls used and the rationale.

It should be clear that, as mentioned earlier [section 6.3.1.4], at most of the companies, no formal RAD methods were in place.

6.3.2.1 Initial Requirement Gathering Practices

Let us first look at their methods for initial requirements gathering, which are summarised in Table 6.2 [refer to Appendix E(b): 4.1.3 of form A2&3, B2&B3, Ca2, Cb2, D2&D3, E2].

Company	sessions	effort	methods
A	1 to 3	No definite answer, normally < 5%	telephoning, tape recording, informal meeting, e-mailing, screen mocking up
B	3	No definite answer, probably < 5%	formal and/or informal meeting, screen mocking up
C	not applicable	not applicable	conventional methods[13]
D	1 to 3 or more depending on the size of the project	around 10%	telephoning, e-mailing, meeting notes, screen mocking up
E	N/A[14]	N/A	formal meeting

Table 6.2 Initial Requirement Gathering

- The overall picture from table 6.2 is that:

- this stage took only a small amount of the total effort, in most cases it was no more than 10 percent of whole project effort;

- more informal methods than formal methods were used. 3 out of 5 used informal approaches;

- typical number of sessions was three or less;

---

[13]clearly defined set of activities which use formal procedures and standards and cover all aspects of system requirements.

[14]Here N/A means unknown or no answer.

- screen mocking up was used widely as the main requirement gathering method.

In addition, the screen mock up tools were not just used for requirement gathering, it often become the front-end interface of the final system, because these tools were normally part of their 4GL development environment.

6.3.2.2 The Control and Rationale

In company A, the usual practice was that developers normally tried to get some essential functional requirements from one or two sessions with customer and/or end-user and then went on to build a working prototype or prototypes. The overall impression was that the control depended largely on individuals in terms of the amount of requirements to be gathered and the effort to be spent at this stage. This practice, apart from the fact that all the RAD developers in the unit were very experienced software developers, was mainly the result of the recognition and mutual understanding from both the manager and prototypers that the enthusiasm for work and flexible control is the key to success [Appendix E(b): 4.1.3 of form A3].

Company B and D had rather similar control structures: a project manager who was in charge of the overall project including resource allocation, and two types of prototypers, one type being more experienced in dealing with customer and/or end-users, especially at the initial requirement gathering stage, and the other more involved in developing prototypes. They were often given some managing responsibility, e.g., project leader in company B [Appendix E(b): 4.1.3 of form B3] and application development manager in company D [Appendix E(b): 4.1.3 of form D3]. The other type of prototypers were less experienced in dealing with customer and more involved in coding and later user validation [Appendix E(b): 4.1.3 of form B3, D3].

Managers in both company B and D put their emphasis on the importance of a sign off at this stage, and the result was that, while the sign-off carried out in company D was a must [Appendix E(b): 4.1.3 of form D3], it did not

happen as intended in company B [Appendix E(b): 4.1.3 of form B3]. But these were very different reasons for such a sign-off. For company B, such a control was mainly due to:

(a) lack of experience in RAD, it was their first attempt to undertake a prototyping project by formally making up a RAD team; and
(b) the control culture of relatively formal conventional systems development approach.

In company D money was the main concern: "... we simply could not afford do it the way IT departments in a big company do: take time, no worry about money, get paid anyway..." as the sales manager JJ [Appendix E(b): 4.1.3 of form D3] of company D put it. Hence, while loose control at early stages is generally a good thing for customer and/user involvement and should be encouraged, tighter control at this stage under particular business settings, as company D, seemed to be practical and necessary.

For company C the initial requirements gathering were mostly done by their marketing and/or commercial group, so there seemed no 'initial requirement gathering stage' as such from the project development team's point of view. This kind of practice was due to the fact that

(a) the customer/user requirements were normally well understood and documented, and

(b) it had an historically relatively complicated infrastructure and rigid process boundaries such as design group, implement group, proving group, QA group and so on and so forth, which had been developed for large conventional stage oriented projects.

The situation had been changing recently, however, as smaller projects come into play, together with new 4GL tools and the ever-increasing demand for fast solutions. For example, one of their new product launch project was carried out by a team using a new 4GL tool and RAD approach, i.e., most of the projects members (managers and prototypers) were involved in most parts of the project from initial requirement gathering and prototypes building, to implementation [section 5.4 and

Appendix D].

On the one hand, some of the project members felt that the much increased productivity was not only because of the use of new tools but also because of the relaxation of tight controls. On the other hand, some worried about the product quality such as the possible poor design and documentation [Appendix E(b): 4.1.3 of Form A2 and A3, D3]

One interesting situation was that they wrote the requirement specifications alongside the prototype building as some were not sure if the prototype itself would be adequate as the specification. When discussed, the feeling was mixed: some felt writing the specifications was waste of time when developing the prototype, while the writer of the specification claimed that the specification was clearer and easier for reference [Appendix E(b): 4.1.3 of Form Ca2 and Ca3] .

In company E, the initial requirements gathering was carried out by various formal interviews with customers and/or users, and the procedures for this stage are well defined in their methodology [Appendix E(b): form E2: 4.1.5]. The main reasons for this were [Appendix E(b): 4.1.3 of form B 1-3]: (a) they were just starting to practice the new methods; (b) large infrastructure and tight control culture like company B; and (c) the project was an internal IT project. In other words, they have less time and finance pressure than those of commercial projects in other companies, as mentioned elsewhere [section 6.3.5].

6.3.2.3 Summary

To summarise, for most companies, the methods used for their initial requirement gathering practice tend to show that:

- while methods used appeared to depend on company infrastructure and control culture, informal approaches, on the whole, seemed more preferable than formal ones.

- it has no more than three iterations, mostly two sessions with customer and/or users;

- the effort for this stage is normally between five and ten percent of the whole development effort;

- screen mocking up and informal meeting were widely used;

- for those companies practising prototyping within their conventional development methods, prototyping was mainly used for design experiment rather than other purpose such as initial requirement gathering;

- no clear framework on what and how much initial requirements should be gathered;

- the level of control here varies from company to company, but on the whole, lack of clear framework for control;

- sign-off were mostly emphasised here due to lack of experience in RAD approach, there is, however, a need for it if cost is the main concern.

6.3.3 Pilot or First functional Prototype(s) Building

Here the pilot or first working prototype(s) means one or more prototypes which are built immediately after the initial requirement gathering and before the first time customer and/or user validation.

Conventional projects tend to have a clear boundary between each stage such as requirement gathering, analysis and design, implementation. In contrast, the boundaries for the RAD projects are very much blurred [refer to figure 6.4 to 6.8]. The observation was that once the initial requirements were gathered, the next thing was to quickly build something working for early customer and/or user validation. This might involve doing a quick design and/or producing work-break-downs as well as the coding and testing.

6.3.3.1 Approaches and Rationale

All the companies performed some sort of quick design formally or informally. Different companies seemed to have different ways: in Company A, each project was undertaken by one or two experienced prototypers with the supervision of a senior manager, and the quick design was performed largely by the individual prototypers [figure 6.4; Appendix E(b): 4.1.3 of form A2 and A3].

Whereas, the quick design in company B was designated to one person — the project team leader — who has both expertise in systems analysis and domain knowledge [Appendix E(b): 4.1.3 of form B2 and B3].

Company D had a similar practice to company A, the difference was that company A's developers were all experienced all-rounders, whereas some of the developers in company D were relatively inexperienced.

The practice in companies C and E had one thing in common: both had their own methods for the analysis and design. The difference here, however, was that in company C prototyping was used only as a technique within their conventional structured development methods, whereas in company E methods and standards were extended to include software prototyping.

On the whole, there seemed to be three different approaches:
Ad hoc approach. This was typical in company A [Appendix E(b): 4.1.3 of form A3] and partially in company D [Appendix E(b): 4.1.3 of form D3], in which the prototypes were built almost directly out of the initial requirements gathered — the meeting notes or recording, screen mock-ups etc. Usually the prototypers had some design ideas either in mind or in written form and tried them out using their 4GL tools until they found the working and acceptable ones that matched the initial requirements.

Semi-formal approach. Company B used the data modelling and a high level functional specification [Appendix E(b): 4.1.3 of form B3; Appendix G] before the coding and testing. This approach was also sometimes practised in company D.

Near-formal approach. Some of this type of practice were very much similar with conventional methods, the only difference was that some detailed design were not finalised until the feedback from some implementation experiments. This was evident in company C [Appendix E(b): 4.1.3 of form Ca3 and Cb2] and company E [Appendix E(b): 4.1.3 of form E1-3].

There were several main factors which influenced these practices:

1) Size of projects It seemed that companies using more formal approaches were often involved in developing large projects, and vice versa [see Appendix A and table 6.3].

2) Experience and domain knowledge The degree of formalism adopted by these companies was influenced by their developers' software development experience and their domain knowledge. For example, the developers in company A — where ad hoc approaches were largely practised — were all experienced and had a great deal of business knowledge [Appendix E(b): 4.1.3 of form A3]. By contrast, the project team in company B — where semi-formal approaches were used — had 4 junior developers and only one senior developer [Appendix E(b): 4.1.3 of form B3].

3) Manager's attitude The managers' attitude seemed to influence significantly the particular practice adopted. For example, the managers in company A [Appendix E(b): 4.1.3 of form A2] emphasised that the productivity and quality would be greatly increased by loose and flexible control. At the same time, the manager in company E — where the near formal approach was used — claimed that it would be a total chaos if tight control were not enforced [Appendix E(b): 4.1.3 of form E1]. It should be pointed out that the difference in attitude was also related to the level of experience their developers had [Appendix E(b): 4.1.3 of form A2, D3]. The managerial attitude is discussed further in section 6.3.5.

Other factors like the nature of project, type of the organisation and company culture were also important to the ways that pilot or first prototype were built, this is discussed in section 6.3.6.

6.3.3.2 Quality Control in Pilot or First Prototype Building

Sections 6.3.3.1 and 6.3.3.2 have briefly described the practices and rationale behind in the first iteration of the actual building process. This section discusses the quality control practice — guidelines, standards and checks — during the period of the pilot or first prototype(s) building.

The following table demonstrates the level of control at this stage [refer to the Appendix E(b)].

control\company	A	B	C	D	E
(a) is high-level functional specification required?	not explicitly, but sometime practised by individuals	yes	yes	yes in principle, but rarely practised	yes
(b) is it mandatory and checked?	no	yes	yes	no	yes in principle
(c) are there any standards or procedures for this?	exist, but almost never used	exist, but loosely followed	exist, followed	no	exist, but unknown if followed
(d) is 'quick-design' required and practised?	not explicitly, but somehow practised individually	yes, designated to expert	yes	yes, but on individual basis	yes
(e) are there any guidelines or procedures used for quick design ?	not, but developing	not explicitly	no for quick design [use conventional methods], yes for w-b-ds	no	yes for quick design, no for w-b-d
(f) are they documented?	no	yes	yes	no	yes
(g) is the 'quick design'[15] mandatory and checked?	no	yes	yes	no	yes in principle
(h) are coding & documenting standards existed?	yes	yes	yes	yes	yes

[15]'quick-design' often involves some high-level data modelling and architectural design. It is used in contrast with system design in structured development methods where much more rigorous methods and standards are required.

(i) are they conformed?	yes	yes	yes	yes	yes
(j) are they mandatory and checked?	no	not sure	not sure	not sure	yes
(k) is the extendibility of the first prototype considered and controlled?	not explicitly	not explicitly	N/A	not explicitly	N/A

Table 6.3 Control Practice at Pilot Or First Prototype Building

The important observations are:

- high level functional specifications were mostly required or intended but often not produced or not produced at this point;

- the 'quick design' was performed largely on an individual basis and loosely controlled or not controlled at all;

- the coding and/or documenting standards are established in all cases, but their conformance were mostly unchecked;

- the typical RAD projects as in company A, B and D had much looser control than those who use prototyping within conventional development like company C and E;

- there were few explicit methods or guidelines for quick design and work breakdowns.

The main reasons behind this, similar to those discussed in 6.3.3.1, were cost and speed pressure. Other factors such project size, managers' and developers' attitudes also appear to be important. This is further explored in 6.3.5.

6.3.3.3 Summary

There appear to be three basic practices at pilot or first prototype building in terms of control formality. The deciding factors for their different practice seemed to be their control culture and manager's attitude. 'Ad hoc' methods were mostly used among the companies, that is, the 'high-level specifications' and 'quick design' largely depended on individuals and little control were imposed.

Although such a loose control practice had some advantages such as increased productivity and participants enthusiasm [Appendix E(b): 4.1.3 of form A2&A3, B3], adequate attention should be given to a pilot or first prototype(s) because they are the basis of the final system, and they should be included as a baseline for the configuration management.

6.3.4 User Participation[16] and CRs Control

Sections 6.3.1, 6.3.2 and 6.3.3 above have discussed prototyping practice for three key management and control areas identified: project initiation, initial requirement gathering and first prototype building. This section examines other important issues such as how customers and/or users participated throughout the process, what and how change requests were controlled, what the problems were and why.

6.3.4.1 User Participation

One of the strengths of prototyping over conventional approaches is its earlier validation and greater user involvement (Carey 1990; Mayhew 1990; McDermid 1991). This is also evident in the results of the previous investigation, the preliminary survey and the field modelling [chapter 4 and 5]. The following looks into this further from the viewpoint of the managers and developers.

First let us look at where and how users participated. This is summarised

[16]Here user validation = customer and/or user validation.

in the following table 6.4 [Appendix E(b)].

	A	B	C	D	E
(1) any direct user participation during project initiation?	yes, customer and sometimes end-user normally have direct contact with manager or developer.	yes, but only between the customer and senior managers.	yes, but only between the customer and managers.	yes, but only between the customer and managers.	yes, but only between the customer and managers.
(2) any direct user participation in the initial requirements gathering, how?	yes, by taking part in formal or informal meetings, or the mock-up sessions.	yes, by taking part in formal meetings.	no. [the initial requirements were submitted in written form.]	yes, by taking part in formal or informal meetings, or the mock-up sessions.	yes, by taking part in formal meetings.
(3) screen mock-ups used for user validation?	yes.	no, screen mock-up not used here.	no, screen mock-up not used here.	yes.	no, screen mock-up not used here.
(4) initial requirements signed off by user?	no. not normally signed off [not required by manager].	no. user reluctant, but desired by the manager]	yes.	yes.	yes
(5) any user participation during prototype building, how?	yes, normally by telephone and informal meeting.	no.	no.	yes, normally by telephone and E-mail.	no
(6) first working prototype(s) validated? signed off by user?	validated, sign-offs depend on projects and individuals.	validated, but not signed off [user reluctant].	no.	validated and signed off if it matched the signed off initial requirements.	validated, but not signed off.
(7) intermediate working prototype(s) validated? signed off by user?	validated, but no sign-offs normally. [not required]	validated, but not signed off [user reluctant].	no.	validated and signed off if it matched the signed off new requirements.	validated, but not signed off.
(8) final systems validated and signed off by user?	yes	yes	yes	yes	yes

Table 6.4 User Participation

Looking horizontally at the answers given to each question by all five companies, the overall impression was the extensive users' participation throughout the process — from initial requirement gathering, prototypes building, validation, to final system acceptance [also see Appendix B—the RADs models].

Furthermore, looking vertically at their practices in each company, the following can be observed:

1) companies A and D had a high level of users' involvement which was mostly informal, the main difference was company D had tighter control, e.g., more formal meeting and more sign-offs;

2) company B also had a high level of user involvement, which was mostly by formal meeting, sign-off was intended along the process but did not happen until the end of the project;

3) company E and, especially, company C appeared to have a low level of user's involvement, which seemed largely owing to the fact that they used prototyping mainly for experimenting with design.

To further illustrate customer and/or user participation, chart 6.1 is drawn from Table 6.4 and Appendix E(b). In the chart, the horizontal axis indicates the different methods used at each stage; the vertical axis shows a total number of companies for a given method. For example, at project initiation: three companies used formal meeting and two used informal meeting.

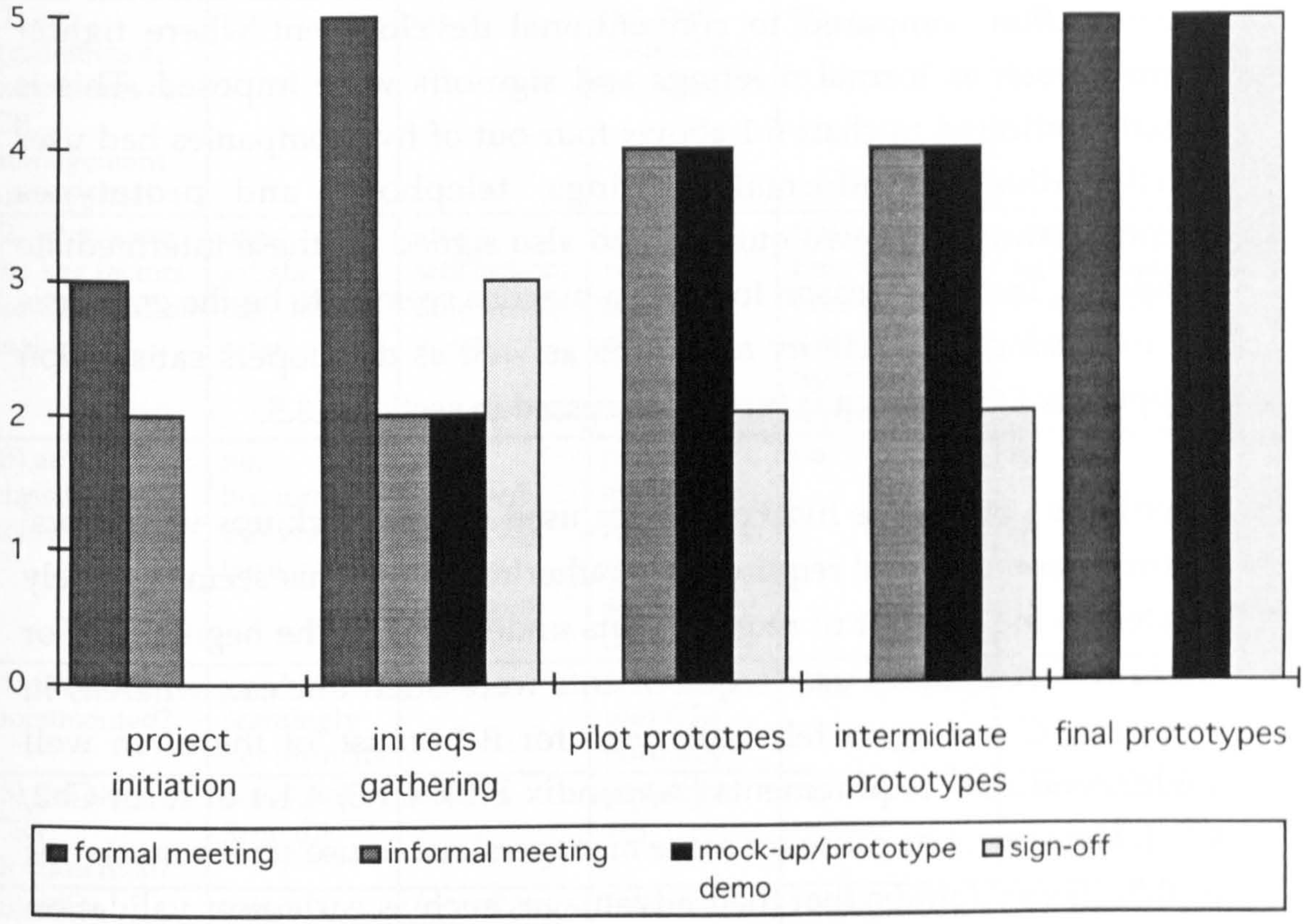


Chart 6.1 Number of Companies Using Each Control Method at Each Stage

4) the level of user participation was high and tightly controlled at both initial requirement gathering and final system validation. From chart 6.1, it is clear that all companies had users participating here by taking part in formal meetings; three out of five companies had users' participation in prototype validation, and sign-off at the end of initial requirement gathering; and all five companies had users participating in their formal meetings and sign-off at the final stage of validation.

The situation here is similar to that of conventional development but with looser control on signing off, particularly at the initial requirements, although most companies had strong intentions to have a sign-off here. This seemed due to either their past experience of conventional methods or their tight financial constraints [Appendix E(b): 4.1.3, 4.1.4 of form A2, A3, B2, Cb1, D3];

5) most companies had more relaxed control over customer and/or users participation compared to conventional development where tighter control such as formal meetings and sign-offs were imposed. This is clearly indicated in chart 6.1 above: four out of five companies had user participation by informal meetings, telephone, and prototypes demonstration; only two out five had also signed off these intermediate products. The main reason for such a practice seemed to be the emphasis on increasing productivity and users as well as developers satisfaction [Appendix E(b)], which is further discussed in section 6.3.5.

6) only two out of the five companies used screen mock-ups with users' participation at initial requirement gathering stage. This seemed closely related to the amount of requirements understood at the beginning. For these two companies user requirements were often unclear, whereas in company C managers felt little need for it because of the often well understood user requirements [Appendix E(b): 4.1.3, 4.1.4 of form Cb2, Cb3]. However, one cause for some managers' not to use screen mock-ups at this stage might be that their advantages such as early user validation were not fully appreciated;

7) most companies — four out of the five — had no end-users and developers involved at the project initiation [refer to table 6.4 above]. As discussed earlier in section 6.3.1 above, it is important to include users at project initiation — the earliest possible stage — in order to establish effective communication and clear individual responsibilities.

6.3.4.2 CRs Control

Change request control is one of the most important control aspects of any software development. Here the change requests may come either from the developer or the customer and/or user, or from both. The following table, which is derived by summarising the interview summaries [Appendix E(b)], gives the overall picture of how CRs were managed and controlled by those companies.

	A	B	C	D	E
(1) any guidelines or procedures on CR management and control?	no	no	yes, but seemed not followed	no	yes
(2) what were the key factors for CR decision making?	user's satisfaction; time limits; effort.	user's satisfaction; time limits; effort.	technical viability; finance; effort;	finance; time limits; user's satisfaction; technical feasibility;	user's satisfaction; time limits; effort.
(3) any CR classification scheme used?	no, but used on individual basis	no, but used implicitly by the managers	not explicitly used by manager and developers	no	no
(4) user CRs documented?	yes, but seemingly not formally	yes	N/A [no user CRs normally]	yes	yes
(5) developer CRs documented?	no, or not formally	no	probably yes	no	yes
(6) main methods for user CRs control?	informal discussions between managers and developers and users;	formal meeting may involve some or all the following: senior project managers, developers and users; or by managers judgement	informal discussions between managers and developers, sometimes with users	formal meetings and sign-offs.	N/A
(7) methods for developer CRs control?	informal discussions between managers and developers	all changes required to be reported to and judged by the managers	informal discussions between managers and developers	informal discussions between managers and developers	CR forms were requested to be completed
(8) any quality considerations in CRs controls?	not explicitly	not explicitly	not explicitly	not explicitly	not explicitly

Table 6.5 CRs Management and Control

The table 6.5 shows clearly the following:

1) guidelines or procedures for CR management and control were not in place in most companies, or hardly used if there were any such procedures [refer to (1) of table 6.5];

2) the main factors in the decision-making of CR control were:

- users satisfaction,
- finance,
- effort,
- technical viability.

Noticeably user satisfaction was the most important factor in their decision making: four out of the five companies considered it an important factor and three of them took it as the first priority [refer to (2) of table 6.5];

3) no classification schemes for CRs were used in the CRs managing and control practice [refer to (3) of table 5].

4) all had tighter CR control and decision-making for users than for developers. User CRs were normally documented and the decisions on the CRs were mostly made by formal and/or informal meeting involving the manager, developers, customer and/or users, agreement normally reached and sometimes signed off [company D]. By contrast, the CR decisions from developers were made by informal meeting or discussions between the managers and developers, and they were, if at all, not documented fully and formally [see (4)-(7) of table 6.5];

5) no explicit considerations and controls were given on product quality issues such as extendibility, reusability, portability etc. in their CR decision making, and the main quality drive seemed to be the user's satisfaction [see (8) of table 5].

On the whole, it is clear that their CRs were loosely managed and controlled, especially on the developer side: there were no explicit quality controls in CR decision making. The obvious reasons here were the

normally tight time scale and small project size. The tight time scale resulted in relaxing the conventional control procedures or standards, and small projects allowed the manager and developer to have a clear overall picture of the project, so there was less the need for tight control. Another important reason appeared to be that most of the managers and prototypers were very experienced system developers and application domain experts, and they worked closely throughout the projects, so the CR decisions were often made there and then by discussions with or simply approval of their manager [Appendix E(b): 4.1.3, 4.1.4 of form A2, B2, Cb3, Ca2, D3].

Furthermore, it is worth noticing the different attitudes among the managers and developers about the quality control. Some felt that, on one hand, there were concerns about the need for tighter quality control because the fear of "unknown quality" might cause future maintenance problems, on the other hand, the usually tight time scale as well as sometimes a relaxed quality control culture meant it was often the result of "no choice's choice" [Appendix E(b): 4.1.3, 4.1.4 of form A2, B3, D3].

Meanwhile, some thought positively about it, argued that loose control was the very advantage of prototyping because it would, not only save time and money, but also be a liberating force for developers enthusiasm. This viewpoint was particularly advocated by KR, the senior manager of company A, who believes that good quality come from the passion for job. Therefore giving 'extra' incentives and trust towards his staff had been high priority for his managing practice [Appendix E(b): 4.1.3, 4.1.4 of form A2, A3].

6.3.4.3 Summary

From the above detailed discussion about practice in managing user participation and CR control, the main lessons learnt here are:

- there were more frequent and extensive user participation through their prototyping or RAD processes than that of conventional development, however, user involvement in project initiation was not as common as expected;

- users were usually involved in much less formal way, such as informal meetings and mock-up sessions, compared to that of conventional development;

- CR control was widely practised, but with more emphasis at the customer and/or user rather than the developer end;

- Few explicit CR control framework were used, and there was little explicit quality consideration in their CR decision making practice;

- Manager attitudes along with their control culture, time and cost were the critical factors to managing and control practice.

Clearly, as the above discussion demonstrates, there is trade off between flexible and tight control over user participation and CR control. Therefore balance should be reached between the speed, the participants enthusiasm, and the quality requirement for RADs type projects. For recommendations given on these issues refer to next chapter [chapter 7].

6.3.5 Organisational Issues

Thus far, this chapter has examined five companies' practice in four key management and control areas of the rapid development process, their particular practices and rationale behind them. This section addresses some other related issues that emerged from the investigations:

- Infrastructure and culture
- Managerial attitude
- Methods and standards
- Participant education
- Participant personality traits

Although most of these issues are more or less discussed in previous sections (6.3.1 — 6.3.4), they need to be more closely looked at because of their significant impact upon the process as a whole.

Based on the interviews [Appendix E(b) — interview summaries, and Appendix B — company background information], the following summary table shows briefly the above points.

	A	B	C	D	E
(a) organisation size? and control culture?	large, tight	large tight	large tight	small loose	medium tight
(b) main project or business orientation?	**internal commercial** project; developing **non-standalone sub-systems** for other departments of same company	**internal commercial** project, developing **stand-alone sub-systems** for other departments of same company	**external commercial** project, developing stand-alone systems for internationa l market	**external commercial** project, developing **stand-alone systems** for other small or medium sized companies	**internal IT** project, developing **stand-alone systems** for its own business needs
(c) managers' attitude?	against tight hierarchical control, emphasis on individual power and co-operation	want reasonably tight hierarchical control, and emphasise co-operation among dept	want tight but more 'flat' control, and emphasise co-operation among dept	want tight hierarchical control	want tight hierarchical control
and actual practice?	loosely controlled, but closely co-operated	tightly controlled but not as tight as intended	tightly controlled	loosely controlled on developer's side but tight on customer and user side	tightly controlled
(d) methods & standards available at :industry or community level?	no	no	yes.	no	no
(e) how are they used?			seemed for quality control		

(f) methods & standards available at company or organisational level?	yes, but no particular ones for RAD developmen t	yes, but no particular ones for RAD developmen t	yes, but no particular ones for RAD developmen t	no	yes, and also have particular ones for RAD developmen t
and how are they used?	used for reference	used for reference	rigorously followed		followed
(g) any guidelines on their use at work group level?	no	no	no	no	no
(h) any need for participants education?	yes, but seemed just on customer and/or user part	yes, but seemed just on customer and/or user part	yes, but seemed just on customer part	yes, seemed just on customer and/or user part	not known at the time

Table 6.6 Organisational Issues

6.3.5.1 Infrastructure and Control Culture

From (a) of table 6.6, it is clear that most of the companies were large organisations having complicated infrastructures and tight control culture, which, to some extent, conflicted inevitably with the application of RAD where simple interaction and flexible control are particularly required. For example, the RAD projects in company A were often a part of a bigger and more conventional project which often involved many departments, and their RAD projects were frequently halted or slowed simply because of the fact that some people or departments were reluctant to co-operate [Appendix E(b): 4.1.3, 4.1.4 of form A3]. This situation, of course, was often beyond the developers and even the project managers' control, "we simply haven't got the muscle", as described by one of the developer [Appendix E(b): 4.1.3, 4.1.4 of form A3].

By contrast, the simple infrastructure and loose control culture in small companies would naturally be better fitted for RAD approach, which was particularly evident in company D — small company with loose control culture [Appendix E(b): 4.1.3, 4.1.4 form D1, D3, D4]. At same time, the loose control seemed to be the cause of the fear for their product quality [Appendix E(b): 4.1.3, 4.1.4 form D1, D3, D4].

6.3.5.2 Business Areas and Control Focus

The (b) of table 6.6 clearly indicates three project orientations: internal commercial, internal IT, and external commercial. Here the 'internal commercial' project means that the project and the project team are internal to their company or organisation, but are financially independent or self-contained; the 'internal IT' project is the project which is internal to its organisation and focus on finding maximised technical solutions; and the 'external commercial' project is to provide commercial software to external customers.

Naturally the difference in project orientation would result in a difference in the control focus. This was indeed reflected in their practice: projects in company A and B both were the 'internal commercial', and both had their control emphasis on producing something quick and useful. Their user validation was therefore relatively tighter controlled [reference to their RADs models in Appendix B] [Appendix E(b): 4.1.3, 4.1.4 of form A3, B3].

Company E , the 'internal IT oriented', seemed to have its control emphasis placed on robust and optimum business solutions, which was manifested in their tight control on requirements gathering, and especially, at design stage [reference to their RADs models in Appendix B] [Appendix E(b): 4.1.3, 4.1.4 of form E1].

In parallel, company C and D had the type of 'external commercial', which was clearly geared towards the financial gains [Appendix E(b): 4.1.3, 4.1.4 of form Ca3, Cb2, D2, D3], however this was reflected differently: one placed tight control on the customer side throughout the process, and the other on the design process. This difference was largely because of their distinct application domains [see Appendix B].

So it is clear that many differences could have been made to the control focus according to distinct business or project orientation even within the same or similar application domain. This suggests that general control procedures and standards are likely to be inadequate, therefore changes should be made to them whenever there is a shift in the business or project orientation. More importantly, the guidelines should be given on how such changes should be made to their processes, in other words, a

meta- process is needed.

6.3.5.3 Managerial Attitudes

Apart from the business culture and project orientation as discussed in (1) and (2) above, the managerial attitude or philosophy seemed also to have a significant influence upon project management and control practice. Look at (c) of the above table 6.6, it is clear that there was apparent consistency between their attitudes and their practice. Comparing above (a) and (c) of table 6.6, it further indicates that such consistency was also largely in accordance with their company control culture, i.e., there was a strong correlation among company control culture, managers attitude and their practice.

At the same time, companies A and D were exceptional either in attitude or practice: managers in company A tried to gain maximum flexibility against traditionally tight control culture, while managers in company D tried to tighten the control in order to balance the simple control structure and lacking of methods and standards [Appendix E(b): 4.1.3 of Form A2, A3 and Form D2, D3]. In both cases, managers attitude appeared to have more or less positive influences on their prototyping practice under the circumstances as indicated in (c) of table 6.6. However, the reverse could occur if their attitude were not properly guarded: over-relaxed control would inevitably aggravate 'unknown quality', and too much control might override other considerations such as individual personality traits. Personality traits are discussed in the "Summary Report of the Personality Test Result"[see Appendix F(a)] and (7) below).

6.3.5.4 Methods and Standards

As shown in (d) — (g) of table 6.6 above, although the company wide procedures and standards were widely available, none of them actually had guidelines on how they could and should be used in what circumstances. The managers and developers were fully aware of this situation throughout the investigations [Appendix E(b): 4.1.3, 4.1.4 of form A2, A3 , B2, B3, D2, D3]. Obviously, procedures and standards at such a

level would not only increase confidence in both managers and developers in their RAD practice, but also help to ensure a adequate level of product quality.

This is evident throughout the investigation [(d) to (h) of table 6.6 above]. Among the five companies, two had made an attempt to develop new methods and standards to adapt to the new approaches. Company A joined an industry wide consortium aiming to develop industry wide RAD methods and standards, and company E had already evolved their conventional methodology to incorporate prototyping approaches.

While, in the case of company A, it is difficult to say at this stage how well such an industry wide standard would fit their own RAD practice, the newly developed system development methodology and standards in company E seemed still too rigorous to follow[17]. Take for instance, a simple count of roles involved in their methodology book gave as many as 18 distinct roles [Appendix E(b): form E2: 4.1.5], and there were only six roles in their actual project undertaken [refer to their RADs model in Appendix B]. When asked, the manager was surprised by the number of the roles supposedly involved, and explained that some roles were combined. However, there was no guidelines on how these situations should be dealt with [Appendix E(b): 4.1.3, 4.1.4 of form E2].

Among those companies who had not had any methods and standards or guidelines for RAD or software prototyping, the need for them was obvious [Appendix E(b): 4.1.3, 4.1.4 of form A2, A3, B2, D2, D3, E2]. The main concern was the lack of a mechanism and standards for the quality controls. For example, one situation occurred in a RAD type project in company C was that they had developed in parallel the prototypes and the requirement specifications, which appeared to have a very similar function. The issue raised was: was there any need for requirement specifications when prototypes would suffice?

All this suggests that the need for appropriate RAD or prototyping methods and standards, and, more importantly, adequate guidelines on

[17]this was an impression that gathered by reading the methodology book provided by the company.

their use.

6.3.5.5 Participant Education

(h) of table 6.6 above shows that four out of the five companies encountered problems with working with customer and/or users.

One type of common problem was the over expectation of customer and users [Appendix E(b): 4.1.3, 4.1.4 of form A2, A3, B2, D2, D3]. At its best, this kind of problem caused no more than feelings of frustration, but it may damage the relationship between customer and developer, and therefore limit the advantage of great user involvement. For example, in company C, one of their uses of prototyping was for cost estimation, but the prototypes were 'hidden-away' from their customers because of the fear that their customer might under estimate the effort and therefore demand to pay less [Appendix E(b) 4.1.3 of Form C2].

Another type of problem was the pressure and resistance from customer and/or users because of the ignorance of the changes of the new approach. This seemed to be especially the case for large companies, "often the customers or users would ask me questions like: have you filled such and such form? Have you got approved from such and such a person?..." so described by RK, the RAD developer of company A [Appendix E(b): 4.1.3, 4.1.4 of form A3].

The implication is that these problems may still widely exist but are neglected by organisations practising or adopting the RAD and/or software prototyping approaches. This further indicates that adequate education should be given to customers and users involved in such a practice, particularly those of large organisations in order to increase the efficiency of communication and therefore to realise the potential of these new approaches.

Finally, one of the general impressions throughout the investigation is that managers and developers often have different views such as what these new approaches are and how they should be used. Although this was not pointed out as a problem by the interviewees, it is clear that a

consensus view on these issues at a work group or least project team level is important and beneficial.

6.3.5.6 Participant Personality Traits

In addition to the issues (6.3.5.1 — 6.3.5.5) discussed above, a personality test (Meyers-Briggs) was carried out as part of the further investigation. This is because prototyping team make-up has been identified as one of the key control areas (chapter 5). The main aim for this testing was to try to understand and learn about the significance and impact of personality on process roles and their interactions. By analysing the initial results from a total of 12 people (five managers and seven prototypers), some interesting results emerge. For example, whilst managers tended to have diverse personalities, (apart from having a common element of strong 'judgement'), most of the prototypers had similar personality types with strong indications of the characteristic of 'extrovert' and 'intuitive'.

Although the results at this stage cannot be generalised due to the small sample size, it is hoped that these observations will bring some useful insights, or least awareness, for managers in their team selection practice. A summary report of the personality test can be found in Appendix F(a).

6.4 Summary

Based on the further investigations of five companies, this chapter has discussed in depth those practical issues about management and control of the RAD or software prototyping project.

The main findings and conclusions on each key control areas are as follows (see also (Chen and Shepperd 1996)).

(a) At project initiation:

- good domain knowledge along with sound systems development experience seem to be the two most important features in selecting a prototyper;

- more clearly defined role responsibilities, especially their interactions are needed;

- more quality considerations are needed at the beginning;

- personality traits and their compatibility seem to be worth considering in the selecting project team.

(b) At initial requirement gathering:

- there appeared to be no more than three iterations here, and to have between five and ten percent of the whole development effort;

- screen mocking up and informal meetings appear to be efficient and effective techniques. Although they are used in some cases, more of their use should be encouraged;

- it seems that there is need for a clearer framework on what and how much initial requirements should be gathered as well as the corresponding controls.

(c) At pilot or first prototype building:

- it seems desirable to have high level functional specifications. They were mostly required or intended, however, often not produced at all or not produced at this point;

- an explicit framework for the 'quick design' is needed. The task were performed largely on individual basis and loosely controlled or not controlled;

- although the coding and/or documenting standards were mostly established, their conformance need to be checked;

- more adequate control such as CR, configuration management and internal product quality are particularly needed for RAD projects.

it should be clear that the pilot or first prototype(s) is particularly important in terms of quality control as it forms the basis of a system to be built, and the baseline for the configuration management.

(d) User participation and CRs control:

- most companies had strong intentions to have a sign-off either due to their experience of conventional methods or tight financial constraints. However, it should be clear that imposing earlier commitment by signing off is likely to reduce users enthusiasm;

- most companies had more relaxed CR control on prototypers than users during the first and later prototypes building and validation, which is likely to cause future maintenance problems; therefore clearer framework and tighter CR control are needed, particularly on developers.

Moreover, other important issues such as organisational culture and infrastructure, methods and standards, managerial attitude, participant education and personality traits have also been discussed in relation to the process. The main lessons learnt here are:

- large companies tend to be more problematic than small ones in practising the RAD approach;

- the control focus of the process need to adjust according the different business orientations;

- managerial attitude seems to have significant impact on their managing and control practice;

- there is a lack of procedures and guidelines on the application of the company wide methods and standards at work group or unit level;

- there is a need for new methods and standards as well as guidelines for RAD;

- there is a need for education about the new approach to all participants in general and customer and end-user in particular;

- there seem to be quite significant differences between manager and prototyper personality profiles as assessed by the Meyers-Briggs personality test.

To conclude, it is clear that the RAD and/or software prototyping process is still far from mature — lacking not only the corresponding methods and standards but also the guidelines on the use of such methods and standards. At the same time, it is evident throughout this investigation, that RAD or software prototyping processes were visibly diverse and could be influenced by many factors. Therefore, to be practical and useful, no single method and standard would be adequate enough, they must be developed or evolved to fit each particular business' needs and environment.

Chapter 7 Conclusions

Synopsis The final chapter starts with a brief review of the work done thus far. It is then followed by two sets of recommendations. The first part presents the main lessons learnt about prototyping including team member selection, prototyper characteristics, methods and standards, and other organisational issues such as infrastructure, control culture and managerial style; the second part gives lessons learnt about conducting the field modelling. The main limitation on the investigation is the lack of direct inclusion of customer and user perspectives, and the main limitation on the method for work is the lack of control over the investigation. Finally, further work is particularly needed in the area of developing 'lean methods' for RAD practice, and a simple management toolset for RAD would also be useful and helpful for the management and control of the process.

7.1 Review

7.1.1 Evaluation of the Research Methodology

As a result of comparing various options (Chapter 3), case study was adopted as the overall research strategy, and questionnaire survey as a supplementary tool. Meanwhile, process modelling techniques were employed to provide a framework for data collection and analysis, and semi-structured interviews were used as the main method for data collection supported by personality testing. The use of process modelling techniques, in particular, was a novel approach to study prototyping projects in industry and proved to be successful.

By applying such a methodology, the following benefits were obtained:

1) the concerns at each stage were further focused and confirmed by the next and/or later stages, and at the same time new concerns were uncovered and included for further study; in other words, each earlier stage functioned as an evolving base for the next stage. Therefore the three-staged multi-case studies approach has been well suited to the open-ended nature of the study.

2) questionnaire survey provided broad views and feedback from practitioners, and an opportunity for identifying and stimulating further co-operation.

3) field modelling as a framework for data collection and analysis proved to be a powerful and particularly useful technique for an empirical study.

4) semi-structured interviews provided both the uniformity of information that was common to all cases, and the flexibility for collecting specific information.

5) the methodology was able to combine evidence from multiple data sources such as observations of their practices, direct views from managers and developers, company standards and/or methods, and project documents.

Meanwhile, the main limitation of this methodology was:

6) the researcher had little control over the other participants of the investigation. This often resulted in delaying or even cancelling visits or meetings, and difficulty in obtaining project documents. Consequently, some variations in the data collected made the data analysis more difficult and less meaningful than it could have been, and access to customers and users was not possible.

7.1.2 Summary of the Work

The main work and findings at each stage is summarised as follows:

1) A review of the literature uncovered the fact that, while many languages, tools and environments were widely available for prototyping, little work had been done in the area of management and control of software prototyping. There seemed to be a particular need for empirical work in order to gain better understanding of the process.

2) The questionnaire survey was designed to have a general opinion about software prototyping from practitioners. 80 organisations were selected from the student placement data base of the Computing Department of Bournemouth University, to which the questionnaires were sent (chapter 4). The most significant finding was that management and control of the process was the most problematic area of concern compared with tools and environments. This finding confirmed the relevance and therefore consolidated the direction of the study from the literature review.

2) Ten prototyping processes were modelled using RADs in eight organisations (two out of eight of the organisations had two different processes each, refer to Appendix B). As a result of the field modelling, four key control areas were identified: project initiation, initial requirements gathering, first prototypes building, and CR and user participation control. The most striking finding was that these models demonstrated a great diversity of prototyping or RAD practices.

The diversity of the process in practice suggests the unlikely usefulness of contriving a universal process model. At the same time, the key control areas and factors that are common to all the processes investigated, point to the importance of the process guidance.

3) Further investigation was carried out in the light of previous findings, which involved interviews with seven managers and nine developers of five companies (many interviews were conducted, of which 16 were recorded and transcript summaries produced [Appendix E(b)]). The main findings were:

- lack of methods and standards for the process and/or lack of procedures and guidelines on using the existing ones for the process;

- lack of clearly defined responsibilities for the process participants;

- lack of quality goals and controls;

- inadequate CR and configuration controls;

- user satisfaction as well as developer enthusiasm played an important role in quality;

- software prototyping as a system development approach (e.g. RAD) appeared to work well for small and medium sized projects;

- software prototyping was still often used as a technique at one or more stages of structured development for large projects.

4) A personality test was carried out among managers and prototypers across the three companies (out of five companies further investigated) as part of the further investigations [Appendix F(a)]. Although only a small sample size, the result seems to show some interesting characteristics about managers and particularly prototypers. For instance, there were some significant similarities of personality traits among prototypers, and differences of personality traits between the managers and the prototypers. In addition, extrovert personalities seemed to predominate.

7.2 Recommendations

The most important lessons learnt from the empirical work led to the following recommendations. For clarity, the main part of the recommendations is further grouped into five topics: project initiation, methods and standards, controls, and organisational issues. In addition, it also includes the lessons learnt about conducting the field modelling, particularly in using Role Activity Diagrams.

At this stage, a note of caution should be sounded. As was discussed in chapter 3, case studies are not intended to generalise into populations but rather into theory. On the other hand, there is an element of replication in this work due to the use of questionnaires, empirical observation and interviews. Moreover, this work addresses a real practical need. Industry needs immediate feedback concerning the conduct of prototyping projects. Whilst, strictly speaking, the findings cannot lead to definite conclusions, it still seems appropriate to make recommendations. Such recommendations might then form propositions for further empirical research.

7.2.1 Recommendations on Management and Control of RAD practice

Sections 7.2.1.1 to 7.2.1.4 (recommendations 1 to 16) below (based on the lessons learnt about RAD project management and control), are therefore particularly given for RAD project managers.

7.2.1.1 Recommendations on Project Initiation

Recommendations 1 to 6 are based on the lessons learnt about RAD team size, team structure, prototyper requisites and role responsibilities (refer to 6.3.1).

Recommendation 1: aim for a small team size.

The normal team size for RAD projects (managers and prototypers) among the companies investigated was relatively small: ranging from two to five persons. This appeared to be the result of the desire to minimise communication overheads. It is also noteworthy that Company E (process 10) had larger projects which appeared to experience some difficulties. This fact is also broadly in line with guidelines given elsewhere (Mayhew 1990; Mayhew and Dearnley 1990; McDermid 1991).

Recommendation 2: keep a flat team structure as possible.

Here 'flat' means less control hierarchy. There appeared to be two basic types of team structure in the investigation, it had either "project manager, design manager, prototypers" or "project manager and prototyper(s)". The former were seen only in medium sized projects and the latter were seen in both medium and small projects, though mainly for small projects. For the same reason given above in 1) the latter practice seemed to be less problematic in which both manager and prototypers felt that the communication was more comfortable and confident, and therefore more effective and efficient.

Recommendation 3: prototypers require a good domain knowledge.

The most important prototyper qualities found by the investigation, were good domain knowledge, experience in most aspects of software development and communication skills. Domain knowledge was particularly emphasised by most of the interviewees as the key element for effective communication with customer and users.

Recommendation 4: have a mix-skilled project team.

Most managers seemed to be aware of the need to have a team of mixed technical skills, as well as other considerations such as developing different skills and training where possible, which is the same as general management practice (McDermid 1991).

Recommendation 5: avoid personality clashes.

Projects appeared to be more successful when choice was given to prototypers forming their own group to avoid personality clashes.

Recommendation 6: clearly define process roles and individual responsibilities at the beginning of a project.

One of the findings indicates that there is a great deal of role overlapping between managers and prototypers. While this is generally a positive thing and should be encouraged as it allows flexible control and increases individual's enthusiasm, care needs to be taken to ensure that a certain level of control is carried out properly. It is important to define clearly the responsibilities including where and how the overlapping may happen for each role. It is also essential to unambiguously assign responsibilities to each team member.

7.2.1.2 Recommendations on Methods and Standards

Recommendation 7 to 9 are about important lessons learnt on the use of methods and standards (refer to 6.3.5.4).

Recommendation 7: clearly define a set of simple principles for conducting the 'rapid analysis and design'.

Instead of formally going through the requirement analysis, high level design and detailed design in separate stages, a rapid analysis and design was mostly performed due to the need for a quick solution. The apparent advantage was the often increased productivity. However, the fact that it was largely practised on an ad hoc individual basis (i.e. hardly any procedures and/or standards were followed) would likely result in 'unknown' quality (internal software quality such as design consistency, extendibility and maintainability, etc.) of the resulting systems.

Recommendation 8: to manage and control effectively, new methods and standards are needed to be developed for RAD.

Although most companies have company wide development methodologies and standards, they were rarely used for RAD projects. Apart from the time pressure, the main problems were that they were

adopted or developed for large conventional development, and were felt to be too rigorous and unsuitable for software prototyping.

Recommendation 9: methods and standards should incorporate guidance on 'exception handling'.

Another problem which was often felt by prototypers was the lack of mechanisms for handling exceptions on the use of their existing methods and standards for RAD approaches. For instance, under what circumstance could things be done differently and what minimum resource requirements are needed? This is to allow prototypers to tailor their standard process to their local circumstances whilst still imposing discipline for core activities. e.g. high-level specifications, CR control and so on.

7.2.1.3 Recommendations on Key Control Areas

Recommendation 10 to 14 are based on the lessons learnt about key control points including initial requirement gathering, high-level specifications, change requests, and sign offs (refer to 6.3.2 to 6.3.4).

Recommendation 10: avoid spending too much effort on the initial requirement gathering stage.

It is interesting to note that, regardless of the formality that each company adopted or the effort being put in for their initial requirement gathering, different approaches resulted in more or less the same figures: about 1/3 of final functional requirements. Projects that had spent more time on initial requirements gathering didn't seem to have significantly increased requirements. Furthermore, the finding shows that typically two iterations and five to ten percent total prototyping effort were normally adequate at this stage.

Recommendation 11: try to use mock-ups.

Mock-ups[18] were found to be an effective means for initial requirements gathering, especially where few requirements were known or clear.

Recommendation 12: high-level functional specifications should be included as a configuration baseline.

Most project managers planned, but often failed, to produce high-level functional specifications after initial requirements gathering: they were often produced at the end of a project, or in some cases not at all. However, they are important not only for a pilot or first functional prototype building, but also, and more importantly, for later system maintenance and enhancement.

Recommendation 13: CR control methods are necessary for a visible and repeatable process.

In most cases, change requests were handled informally and implicitly. There also tended to be an imbalance between the CR control on customer and/or user, and developer part: tight control on customer and user requests, looser or none at all on prototypers. However, in order to better control the process and to ensure an adequate level of software quality, a CR classification method should be clearly defined and applied to both development team and customer and/or end-users. A example of CR classification can be found in Mayhew's change classification approach (Mayhew, Worsley et al. 1989).

Recommendation 14: conventional sign-offs need adapting to fit with RAD approaches.

Possibilities include:

- avoid too many formal sign-offs (a formal sign-off has budgetary

[18]A mock up is a prototype that mimics functionality without its implementation.

implications) if possible, however, one informal sign off at the end of initial requirements gathering (or after the mock-ups if any) would be helpful for maintenance purposes;

- a formal sign-off may be appropriate at the end of the pilot or first functional prototype or prototypes, before further increments;
- one at the end of the fully functional prototypes;

Ideally all main participants should be involved in these sign offs. As a minimum, they should involve managers and prototypers. This is important for maintenance purposes, regardless of whether customer and/or users are involved.

7.2.1.4 Recommendations on Organisational Issues

These include issues such as infrastructure, culture, managerial attitude, and customer and user education (refer to 6.3.5).

Recommendation 15: a balance needs to be reached between control and encouraging participants' enthusiasm.

As the study shows [chapter 6: section 6.3.3.1], managers' attitude seems to have a significant influence on their managing and control practice. Looser control normally leads to an increased enthusiasm for both developers and end-users, but also increases the 'unknown quality'. Therefore a balance needs to be reached between the level of control and the participants' enthusiasm.

Recommendation 16: RAD participants, especially customers and users, need educating about the new approach.

This issue was brought forward by some managers and developers who felt frustrated by the ignorance of the customer and users about the new approach: conventional procedures and bureaucracy were often

demanded by customers and users so that work was sometimes delayed unnecessarily. Another problem is the belief by users that an early mock up or prototype can be used as a production system.

7.2.2 Recommendation on software prototyping in structured development

The following recommendation is given particularly for managers using software prototyping as one of techniques within structured approaches.

As part of the study, the practice of software prototyping as a technique for structured software development was also investigated. Such use has been found in almost every stage of a conventional development: from 'buy-in' demonstration, feasibility study, requirement gathering to design and implementation, though it seems to have been mostly used in the area of design — experimenting with design alternatives. The main problems appeared to be the lack of guidelines and control for use of such a technique, which often caused redundant or 'repeated work' and subsequent frustration to both management and developers.

Recommendation 17: be explicit concerning the use of prototyping techniques to be used in the project plan.

It should include:

- whether or not the prototypes developed are going to be used in later development, and if yes, how.
- under what circumstance the normal conventional standards, procedures and controls can be bypassed.

As was discussed earlier [chapter 6], when prototyping was used as a technique within structured development methods, little attention was given to its management and control. However, this may cause conflicts if care is not taken to harmonise the two different approaches between staged-wised specifying and the iterative prototyping.

7.2.3 Recommendations on Process Modelling

Process modelling has been employed as a research vehicle for this empirical study and the rationale was to uncover problems by modelling the processes under investigation. In principle, many graphical notations may be used to model software processes such as DFDs, OMT, Statemate and many other general modelling notations [chapter 5]. Relatively speaking, RADs were a good candidate for modelling a process from an organisational viewpoint, as they were simple to use and to understand. The following guidelines are drawn from the field modelling experience of the researcher. They may be useful for modelling processes in general or Role Activity Diagrams (RADs) modelling in particular.

7.2.3.1 Recommendations on field study and modelling.

The following recommendations (18 and 19) may be useful for researcher conducting field study and modelling.

Recommendation 18:process modelling is an effective way to highlight problematic areas and uncover overlooked problems for empirical investigation.

It has been clearly demonstrated that, as part of the research methods, process modelling has played an important role — framework for data collection and analysis, highlight and uncover problem areas — throughout the investigation.

Recommendation 19: have clear objectives before starting to model a process.

Even for simplest process there are always different aspects and different level of details to model about, and as was experienced in the early stages of the field modelling, time may be wasted by too much detail modelled for a less important part in one place, and neglecting an important part in another. Therefore, clear objectives and focuses are essential for an effective modelling.

7.2.3.2 Recommendations on Modelling Using RADs

The following (recommendations 20 to 26) are lessons learnt about field modelling using Role Activity Diagrams (RADs). They therefore may be useful for anyone who would use RADs in his field study.

Recommendation 20: interviewing one person at a time is more efficient than group interviews.

Experience in the early stages of the field modelling indicated that it was less effective when two or more people were present at the same time. This was because time was easily wasted, either the cautious attitudes about expressing oneself in front of others, or the likely disagreement in detailed issues such as terminology for activities or interactions.

Recommendation 21: identify a key process role as a starting point.

For example, it was found to be more effective when starting with the project manager's viewpoint because the project manager normally has a better overview of the process than other process participants.

Recommendation 22: identify all the key process roles before modelling any activity.

The significance of a role normally depends on its significance to the process. For example, for a large conventional project, activities such as the specifying, designing and testing should all be treated as distinct process roles, but for a small or medium prototyping project, these roles may treated as one role as prototyping. A practical guideline, from my experience, is that if an activity needs to be dedicated to as one person's job during the process then this activity should be modelled as a distinct process role. Otherwise, it should be modelled as an activity under one or more roles.

Recommendation 23: identify the role interactions once the key roles are identified.

An interaction here means an activity involves two or more roles. Because it is an essential part of the RADs modelling, most of the interactions should appear on the diagrams and have higher priority than activities done within one role.

Recommendation 24: focus upon key activities after identifying roles and interactions.

Apart from the role interactions, there are some other important activities under each process role that also need to be added. The activities should have adequate significance for the purpose of the modelling (i.e. try to avoid too general or too trivial activities).

Recommendation 25:modelling chronologically is an effective approach.

Although the order of activities to be modelled should have no particular importance, from experience of the field modelling, it was easy and clear to model them chronologically.

Recommendation 26: include more than one process participants' views where possible.

Once a draft model was derived with the project manager, it was normally reviewed by prototypers, which often resulted in many valuable inputs. Although no customer and user viewpoints were able to be included, it is conceivable that their inclusion would lead to a more complete picture of the process.

7.3 Limitations of the Study

The limitations of the study both on research methodology and research scope are recapitulated as follows.

7.3.1 Limitations on Research Methodology

As was discussed at length in chapter 3, and briefly evaluated in above 7.1.1, the main shortcomings of the methods for work have been:

- **little control over the investigation.** As a result of this, meetings were often delayed for days, or even months, or worse still, cancelled altogether. The implication was that some missing or incomplete data might weaken the argument in places.

- **difficult to analyse data which varied from company to company.** Either owing to variance in the process within each organisation, or their willingness to co-operate, the data gathered often varied from company to company. This might have reduced meaningful comparisons.

- **difficult to balance the level of generalisability and the level of detailed knowledge of the process.** Given the limited time and resource, there is a trade off between more cases and less detailed knowledge. To reach the 'right balance' appeared to be particularly difficult. In retrospect, three instead of five companies might be a more appropriate for the investigation and would of yielded more detailed insights, although admittedly for a smaller number of projects.

7.3.2 Limitations on research scope

They are:

- **limited cases.** Although less cases normally mean more detailed information, they may also be less representative of the wider

population of prototyping projects.

- **limited application domain.** The investigation was mainly carried out in the area of information system development, therefore the applicability of the results in other application areas is largely limited;

- **lack of customer and user perspectives.** They were unable to be included directly though they were investigated from the development team view points. This was mainly due to the limited resources available from both the companies and the researcher. The main drawback of this imbalance of viewpoints is the reduced validity regarding customer and user involvement;

- **imbalance of cases between software prototyping as techniques within conventional structured development and as a system development approach.** The initial intention of the research scope was to balance, on one hand, the software prototyping as a technique within structured development, and an alternative software development approach on the other. However, the emphasis has been more on the latter, i.e., the RAD approach due to the fact that it is the more popular systems development approach. Among the five further investigated companies, only one had the former use of prototyping. That is, there was less compelling evidence regarding the use of software prototyping as a technique within structured development.

7.3.3 Summary

Given the limitations stated above in 7.3.1 and 7.32, the results of the study are inevitably restricted to some extent. For instance, there were no control of cases and with only five processes studied in depth; it is not possible to be confident of their representativeness. Nevertheless, the overall objectives of the empirical study have been met in the form of insights gained into the management and control of software prototyping in general and for RAD in particular. In addition, a number of propositions have been made for further study.

7.4 Further Work

As a result of this empirical study, five areas for further work have been identified in the following sections (7.4.1 — 7.4.5). They are:

- customer and user perspective;
- the 'inner loop' of the process or prototype building;
- process metrics and quality measures;
- 'lean' methods and standards for RAD;
- simple and easy to use toolset for RAD.

7.4.1 Customer and User Perspective

To better understand software prototyping in general and RAD in particular, different views from all process participants need to be considered. However, as pointed out in 7.3.2, the customer and user view points were unable to be included directly in the study. Further work is therefore needed to look into more closely customer and user viewpoints. The questions to be asked might include:

- what are the ways in which they would like to participate in the process?
- how do they want CR to be managed and controlled?
- what and when to sign off in their view?
- what are their views of quality?

7.4.2 The 'inner-loop' of Prototyping Process

The 'inner-loop', as identified earlier, is the iteration of the prototype building process performed by one or more prototypers. In contrast, the 'outer-loop' is more concerned with the interaction between roles from project start to finish, which the study has mainly dealt with. Given the time and resources available, it has been intentionally treated as one activity or management control entity. However, as the study suggests, this stage forms the basis of system evolution and quality of the end product, and is therefore worth being examined more closely. The

important issues here may include:

- **the documentation and coding standards.** For example, how well are they observed and updated?

- **'quick design'.** While the study has shown it has been largely performed on an individual basis, it would be interesting to find out how it is actually done, and what are the good principles, and are they shared among the developers?

- **prototype building techniques.** Although they may vary according to development environments and individuals, what are the good practices that are more or less 'constant' which can be shared among practitioners?

- **design quality.** For example, are issues such as extendibility, reusability and portability are addressed in practice? and how?

- **tools and environments.** How do different tools and environments affect their management and control practice?

7.4.3 Process Metrics and Quality Measures

To answer the questions asked in 7.4.1 and 7.4.2 above, it is important to collect data about those key factors in each key management and control areas that have been identified through this study. The data could then be used to :

- perform quantitative analysis on the process to observe the relevance or significance of these factors to the process;

- compare the quantitative results with the qualitative analysis of the study so that complementary recommendations can be made for process improvement;

- form a basis for further propositions as well as subsequent experiments about better process models.

7.4.4 Simple and Flexible Prototyping Methods and Standards — Lean Methods

One important finding was that conventional methods and standards are often too rigorous and inflexible for software prototyping, especially for RAD type projects. Indeed, a comment was made by one manager that the DSDM method, intended for RAD projects, was too complex. Therefore, adequate and appropriate methods and standards need to be developed. It is important that, as the finding suggests, they should include 'exception handling' mechanisms in order to allow greater flexibility.

7.4.5 Simple and Easy-to-Use Toolset

At present many tools and environments have been developed for software development, particularly for the RAD approach. However, few practical QA and project management tools are available for software development, particularly for RAD. Therefore, a simple and easy-to-use toolset that incorporates the lessons learnt from this study could be useful.

7.5 Final Conclusions

This thesis has described an empirical study into the management and control of software prototyping in a field that sadly lacks this type of analysis. The empirical study has made the following contributions to a better understanding and improvement of the process:

- The study has demonstrated that our understanding of software prototyping or RAD processes can be enhanced by applying simple and direct modelling techniques.

- The study has shown that the management and control process of software prototyping is of great diversity. This suggests that a universal model for such a process is unlikely to be of practical use.

- The study provides evidence of the need, and some ground, for

developing simple and flexible methods and standards for RAD as well as for prototyping within structured development.

- The study has identified a number of key management and control areas and factors for software prototyping in general, and RAD approaches in particular, which could provide valuable inputs for measurement and further improvements of the process.

- The study has resulted in a set of recommendations for practitioners.

In conclusion, the empirical study has shown that management and control of software prototyping is indeed problematic, in that few good management and control practices were seen across the companies investigated. One of the most important findings is that of the lack of adequate methods and standards for the process. The study has indicated many key factors which affect the management and control practice for practitioners. Furthermore, the study results in the identification of five areas for further work. At the same time, it is recognised that the main shortcomings of the research are the lack of customer and user perspectives and the technical aspect of the process, which also have direct impact on the management and control practice. Nevertheless, the study has achieved its overall objective that is to have a better understanding of, and to make some practical and theoretical contributions to, the management and control of software prototyping.

References

Aaram and Jarle (1984). The BOP Prototyping Concept. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 179-187.

Abeysinghe, G. and K. Phalp (1997). "Combining Process Modelling Methods." Information & Software Technology **39**(2): 107-124.

Alexander, H. (1990). Software design and prototyping using *me too*. Prentice Hall.

Austin, S. and G. Parkin (1992). Notes on the +/- of Formal Methods. NPL, Teddington.

Basili, V. R. (1985). Measuring the Software Process and Product: Lessons learned in the SEL. Proceedings Tenth Annual Software Engineering Workshop, NASA Goddard Space Fight Centre, Greenbelt MD 20771,

Basili, V. R. and H. D. Rombach (1987). Tailoring the Software Process to Project Goals and Environments. Proceedings 9th International Conference on Software Engineering, Monterey,

Basili, V. R. and H. D. Rombach (1991). "Support for comprehensive reuse." Software Engineering Journal **6**(5): 303-316.

Baskerville, R. and S. Smithson (1995). "Information technology and new organizational forms: choosing chaos over panaceas." European Journal of Information Systems **4**(2): 66-73.

Berosff, E. H. and A. M. Davis (1991). "Impacts of Life Cycle Models on Software." Communications of the ACM **34**(8): 104-118.

Bidoit, M., H.-J. Kreowski, et al. (1991). Algebraic System Specification and Development - A Survey and Annotated Bibliography. Springer Verlag.

Bischofberger, W. and G. Pomberger (1992). Prototyping-oriented software development: concepts and tools. Springer-Verlag.

Blazy, S. and P. Facon (1995). "Formal Specification and Prototyping of a Program Specializer." Lecture Notes in Computer Science **915**: 666 - 680.

Blum, B. I. (1985). Thoughts on the software process. 2nd Intl. Softw. Process Workshop, Cota de Caza, Ca, USA, ACM Sigsoft Softw. Eng.

Notes.

Blum, B. I. (1986). "Application Systems Prototyping. In: Lipp, M.E. (ed.): Prototyping - State of the Art Report." Pergamon Infotech Ltd, Maidenhead **14**(4): 3-14.

Boar, B. H. (1984). Application prototyping. Wiley-Interscience.

Boehm, B. W. (1984). "Software engineering economics." IEEE Transactions on Software Engineering **10**(1): 4-21.

Boehm, B. W. (1988). "A Spiral Model of Software Development and Enhancement." IEEE Computer **21**(5): 61-72.

Borba, P. and S. Meira (1997). "A system for translating executable VDM specifications into lazy ML." Software Practice & Experience **27**(3): 271-289.

Bourke, M. (1986). "Actual Experience in Prototyping. In: Lipp, M.E. (ed.): Prototyping - State of the Art Report." Pergamon Infotech Ltd, Maidenhead **14**(4): 15-26.

Boyer, P. (1995). "Is RAD all wet?" Datamation **41**(16): 84.

Brice, J. L. and L. B. Connel (1989). Structured rapid prototyping. Yourdon Press.

Brooks, F. P. (1995). "The Mythical Man - Month After 20 Years." IEEE Software **12**(5): 57 - 60.

Budde, R. (1984). Approaches to Prototyping. The Working Conference on Prototyping, Namur, Springer-Verlag.

Budde, R. (1992). Prototyping - an Approach to Evolutionary System Development. 15th international conference on software engineering, Springer-Verlag.

Budde, R. (1992). "What is prototyping?" Information Technology & People **6**(2-3): 89-96.

Budde, R., Reinhard, et al. (1984). From Application Domain Modelling to Target System. The Working Conference on Prototyping, Namur, Springer-Verlag.

Budde, R. and H. Zullighoven (1992). "Prototyping revisited."

Information Technology & People **6**(2-3): 97-108.

Budgen, D. (1984). The Use of Prototyping in the Design of Large Concurrent Systems. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 49-57.

Buti, L. (1996). "GD-Workbench: A System for Prototyping and Testing Graph Drawing Algorithms." Lecture Notes in Computer Scicene **1027**: 111 -122.

Callender, E. D. (1985). A meta model for the software process. 2nd Intl. Softw. Process Workshop, Coto de Caza, Ca, USA,

Card, D. N. (1995). "The RAD Fad: Is Timing Really Everything?" IEEE Software **12**(5): 19 - 22.

Carey, J. M. (1990). "Prototyping : alternative systems development methodology." Information and Software Technology **32**(2): 191-126.

Cavano, J. P. and J. A. McCall (1978). A framework for the measurement of software quality. Softw. Quality Assurance Workshop, San Diego, Calif.,

Cavaye, A. L. M. (1996). "Case study research: a multi-faceted research approach for IS." Information Systems Journal (6): 227-242.

CCTA (1993). Prototyping in an SSADM Environment. HMSO.

Cerpa, N. and J. Verner (1996). "Prototyping : Some New Results." Information & Software Technology **38**(12): 743 - 755.

Chapin, N. (1983). "Prototyping--Quick, Not Dirty..." Data Management **21**(10): 46-48.

Chen, J. and J. Wang (1989). An integrated framework for software prototyping. Proceedings of the 13th Annual International Computer Software and Applications Conference, Orlando, FL, USA,

Chen, L. and M. Shepperd (1996). An Empirical Investigation of the Quality Control in Rapid Application Development: Problems and Suggestions. Fifth European Conference on Software Quality, Trinity College, Dublin, Ireland,

Chistensen (1984). Prototyping of User-Interfaces. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 58-67.

Chroust, G. (1986). Backtracking in software process models. 3rd Intl. Software Process Workshop, Breckenridge, Co, USA,

Curtis, B. (1990). Empirical studies of the software design process. NTERACT '90, Elsevier Science.

Curtis, B., M. Kellner, et al. (1992). "Process Modeling." Communications of the ACM **35**(9): 75-90.

Curtis, B., H. Krasner, et al. (1988). "A field study of the software design process for large systems." Communications of the ACM **31**(11): 1268-1287.

Davis, A. M. (1992). "Operational prototyping: a new development approach." IEEE software **9**(5): 70-78.

Dearnley, P. A. and P. J. Mayhew (1983). "In Favour of System Prototypes and Their Intrgration into the System Development Cycle." Computer Journal **26**(1): 36-42.

Deiters, W., V. Gruhn, et al. (1989). Systematic development of formal software process models. Proc. 2nd European Software Engineering Conference, Warwick, Coventry, UK, Springer.

DeMarco, T. (1982). Controlling Software Projects. Management, Measurement and Estimation. NY, Yourdon Press.

Dowson, M. (1985). The structure of the software process. 2nd Intl. Softw. Process Workshop, Coto de Caza, Ca., USA,

Dowson, M. (1986). Iteration in the Software Process: Workshop introduction and overview. 3rd International Software Process Workshop, Breckenridge, Colorado, USA,

DTI (1992). TickIT Guide to Software Quality Management System and Certification using EN29001. Department of Trade and Industry.

Editorial (1996). "Visual RAD Gets Very Small." Byte **21**(9): 32.

Endres, A. and H. Weber (1991). Software Development Environments and CASE Technology. Software Development Environments and CASE Technology, Konigwinter Germany, Springer-Verlag.

Eom, S. (1996). "A survey of operational expert systems in business (1980-

1993)." INTERFACES **26**(5): 50-70.

Fenton, N. (1993). "How effective are software engineering methods." Journal of Systems and Software **22**(2): 144-146.

Fenton, N. and B. A. Kitchenham (1991). "Validating software measures." Journal of Software Testing, Verification and Reliability **1**(2): 27-42.

Fenton, N. and A. Melton (1990). "Deriving structurally based software measures." J. of Systems & Software **12**(3): 177-187.

Fenton, N. and S. L. Pfleeger (1994). "Science and Substance: A Challenge to Software Engineers." IEEE software **11**(4): 86-95.

Fernstrom, C. (1992). "Tools and environments to improve the software porcess." Information & Software Technology **34**(10): 659-673.

Finkelstein, A. (1992). "A software process immaturity model." ACM SIGSOFT Software Engineering Notes **17**(4): 22-23.

Finkelstein, A., J. Kramer, et al., Ed. (1994). Software Process Modelling and Technology. Research Studies Press (distributed by Wiley).

FIPS (1993). Integration Definition For Function Modelling. National Institute of Standards and Technology.

Floyd, C. (1984). A Systematic Look at Prototyping. Approaches to Prototyping. Namur, Spring-Verlag Berlin Heidelberg. 1-18.

Frailey, D. J., R. R. Bate, et al. (1991). Modeling information in a software process. 1st IEEE International Conference on the Software Process, Redondo Beach, CA,

Friel, G. and D. Budgen (1997). "Design transformation and prototyping using multiple viewpoints." Information & Software Technology **39**(2): 91 - 105.

Galliers, R. D. and S. King (1994). "Modelling the CASE process: empirical issues and future directions." Information & Software Technology **36**(10): 587-596.

Galliers, R. D. and F. F. Land (1987). "Choosing Appropriate Information Systems Research Methodologies." Communications of the ACM **30**(11): 900-902.

Goldberg, A., C. Green, et al. (1986). Iteration in the software process. 3rd International Software Process Workshop, Breckenridge, Col, USA,

Goldberg, A. and D. Robson (1983). Smalltalk-80m the language and implementation. Addison-Wesley.

Gomaa, H. (1986). "Prototypes - Keep Them or Throw Them away?. In: Lipp, M.E. (ed.): Prototyping - State of the Art Report." Pergamon Infotech Ltd, Maidenhead **14**(4): 41-54.

Gordon, S. and B. J (1991). Rapid Prototyping and Software Quality: Lessons From Industry. Department of Computer Science, Colorado State University.

Graham, D. R. (1989). "Incremental developmemt: reveiew of nonmonolithic life-cycle development models." Information & Software Technology **31**(1): 7-20.

Grant, F. J. (1985). "The Downside of 4GLs." Datamation **15**(7): 99-104.

Greenwood, R. M., I. Robertson, et al. (1995). Active Models in Business. Business IT Conference , Manchester,

Grehan, R. (1997). "Java's RAD Route to Data Access." Byte **22**(2): 192.

Grief, I., Ed. (1988). Computer Supported Cooperative Work. Morgan Kaufman.

Gutierrez, O. (1993). "A contingency perspective on effective prototyping." Journal of Information Technology **8**(2): 99-109.

Hakim, C. (1987). Research Design. Allen&Unwin.

Hammersley, M. (1992). What's wrong with Ethnography? Routledge.

Hansen, G. A. and M. I. Kellner (1988). Software process modelling: the Triad approach. 6th Symposium on Empirical Foundations of Information & Software Sciences,

Hardgrave, B. C. (1995). "When to prototype: decision varibles used in industry." Information & Software Technology **37 (2)**: 113-118.

Harker, S. (1988). "The Use of Prototyping and Simulation in Development of Large Scale Applications." The Computer Journal **31**(5): 420-425.

Harrison, R. (1995). "Estimating the Qaulity of Functional Programs - an Empirical Investigation." Information & Software Technology **37**(12): 701-707.

Hekmatpour, S. H. and D. C. Ince (1988). Formal Methods, software prototyping and VDM. Lodon, Addison-Wesley.

Henderson, P. (1986). "Functional programming, formal specification and rapid prototyping." IEEE Transactions on Software Engineering **12**(2): 241-250.

Henhapl, W., S. Kaes, et al. (1991). Utilizing Fifth Generation Technology in Software Development Tools. Software Development Envrironments and CASE TEchnology, Konigwinter , Germany, Springer-Verlag.

Heping, H. and H. Zedan (1996). "An Excutable Specification Language for Fast Prototyping Papallel Responsive Systems." Computer Languages **22**(1): 1-13.

Hilal, D. K. and H. Soltan (1992). "To prototype or not to prototype ? That is the question." Software Engineering Journal **7**(6): 388-392.

Hollinde (1984). Experience of Prototyping in Command and Control Information Systems. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 80-91.

Holt, A. W., H. R. Ramsey, et al. (1983). "Coordination System Technology as the Basis for a Programming Environment." Electrical Communication **57**(4): 307-314.

Hudak, P. (1989). "Conception, Evolution, and Application of Functional Programming Languages." ACM Computing Surveys **21**(3): 359-411.

Hughes, J. (1989). "Why functional programming matters." The Computer Journal **32**(2):

Humphrey, W. S. (1988). "Characterising the software process: a maturity framework." IEEE Software **5**(2): 73-79.

Humphrey, W. S. (1989). The SEI software process program. IFIP 11th World Computer Congress, San Francisco,

Humphrey, W. S. and M. I. Kellner (1989). Software process modeling: principles of entity process models. 11th International Conference on

Software Engineering, IEEE Computer Society Press.

Iggulden, D. (1986). "Prototyping Development. In: Lipp, M.E. (ed.): Prototyping - State of the Art Report." Pergamon Infotech Ltd, Maidenhead **14**(4): 55-63.

Janson, M. A. and L. D. Smith (1985). "Prototyping for systems development: a critical approach." MIS Q. **9**(4): 305-316.

Kammersgaard, J. (1984). A Discussion of Prototyping Within a Conceptual Framework. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 294-321.

Kautz, K. (1993). "Communication Support for Prototyping Projects." **35**(11-12): 647 - 651.

Kautz, K. (1994). "The Failure to Introduce System Development Methods - A Factor Based Analysis." IFIP Transactions A - Computer Science and Technology **45**: 275 - 287.

Kellner, M. I. (1989). Software process modeling: Value and experience. Carnegie-Mellon, Software Engineering Institute.

Kellner, M. I. (1990). Supporting software process through software process modeling. 6th International Software Process Workshop, Hokhaido, Japan,

Kellner, M. I. (1991). Software process modeling support for managment planning and control. 1st IEEE International Conference on the Software Process, Redondo Beach, CA,

Kellner, M. I. and G. A. Hansen (1988). Software Process Modeling. Software Engineering Institute, Carnegie Mellon University.

Kellner, M. I. and G. A. Hansen (1989). Software process modeling: A case study. 22nd IEEE Annual Hawaii International Conference on System Sciences, Hawaii, IEEE.

Kitson, D. H. and W. S. Humphrey (1989). The role of assessment in software process improvement. Software Engineering Institute, Carnegie-Mellon University, Pittsburgh.

Kraushaar, I. M. and L. E. Shirland (1985). "A prototyping method for applications development by end users and information systems specialists." MIS Q. **9**(3): 189-196.

Kruchten, P. and E. Schonberg (1984). "The Ada/Ed System: A Large-Scale Experiment in Software Prototyping Using SETL." TSI-Techniques et Science Informatiques **3**(3): 193-200.

Lantz, K. E. (1986). The protoyping Methodology. Englewood Cliffs, New Jersey, Prentice-Hall.

Leibrandt, U. and P. Schnupp (1984). An Evaluation of Prolog as a Prototyping System. Approaches to Prototyping, Berlin, Springer-Verlag.

Lemieux, J. (1996). "Using RAD tools to develop secure client/server applications." Computers & Security **15**(4): 289 - 295.

Lewis, J. A., S. M. Henry, et al. (1991). An Empirical Study of the Object-Oriented Paradigm and Software Reuse. OOPSLA '91, ACM.

Lichter, H., M. Schneider-Hufschmidt, et al. (1993). Prototoyping in Industrial Software Projects: Bridging the Gap Between Theory and Practice. 15th international conference on software engineering, Baltimore, Maryland, IEEE Computer Society Press.

Liebowitz, J. (1994). "Expert Systems and Interactive Multimedia Techonologies in Telecommunications." Information and Decision Technologies **19**(6): 499-505.

Lim, K. Y. and J. B. Long (1992). Rapid prototyping, structured methods and the incorporation of human factors into system development. St. Petersburg,

Linton, M., J. Vlissides, et al. (1989). "Composing User Interfaces with InterViews." IEEE Computer **22**(2): 8-22.

Lipp, M. E. (1986). "Analysis. In: Lipp, M.E. (ed.): Prototyping - State of the Art Report." Pergamon Infotech Ltd, Maidenhead **14**(4): 129-184.

Livesey, P. B. (1984). Experience with Prototyping in a Multi National Organization. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 92-104.

Long, J. (1996). "Specifying relations between research and the design of human-computer interactions." International Journal Of Human-Computer Studies **44**(6): 875-920.

Lott, C. M. (1993). "Process and measurement support in SEEs." ACM

SIGSOFT Software Engineering Notes : 83-93.

Lott, C. M. (1994). "Measurement Support In Software Engineering Environments." International Journal Of Software Engineering And Knowledge Engineering **4**(3): 409-426.

Luqi (1992). "Status Report: Cpmputer-Aided prototyping." IEEE Software **9**(6): 77-81.

Luqi and M.Ketabchi (1988). "A Computer Aided Prototyping Systems." IEEE Software **5**(2): 66-72.

Luqi and Y.Lee (1989). Interactive control of prototyping process. Proceedings of the 13th Annual International Computer Software and Applications Conference, Orlando, FL, USA,

Maresh, P. and D. Wastell (1990). Process Modelling and CSCW: An Application of IPSE Tenchnology to Medical Office Work. INTERACT '90, Elsevier.

Martin, J. (1986). Forth Generation Languages. Englewood Cliffs, New Jersey, USA, Prentice-Hall.

Martin, M. P. (1991). "Converting prototypes to operational systems: evedence from preliminary industrial survey." Information & Software Technology **33**(5): 351-356.

Mathiassen, L., J. Stage, et al. (1992). "The principle of limited reduction in software design." Information Technology&People **6**(2-3): 171-185.

Mayhew, P. J. (1989). An Investigation of Information Systems Prototyping. UEA, Norwich, UK.

Mayhew, P. J. (1990). "Software Prototyping: Implications for the People Involved in Systems Development." Lecture Notes in Computer Science **436**(1): 290-305.

Mayhew, P. J. and P. A. Dearnley (1987). "An Alternative Prototyping Classification." Computer Journal **30**(6): 481-484.

Mayhew, P. J. and P. A. Dearnley (1990). "Organisation and management of systems prototyping." Information & Software Technology **32**(4): 245-252.

Mayhew, P. J. and P. A. Dearnley (1990). "Organization and management

of systems prototyping." Information & Software Technology **32**(4): 245-252.

Mayhew, P. J., C. J. Worsley, et al. (1989). "Control of software prototyping process: change classification approach." Information & Software Technology **31**(2): 59-66.

McDermid, J. (1991). Software engineering's reference book. Butterworth-Heinemann Ltd.

Meyer, B. (1992). Eiffel the Language. New York, Prentice Hall.

Miller, D. C. (1991). Handbook of Research Design and Social Measurement. Academic Research.

Millington, D. and J. Stapleton (1995). "Developing a RAD Standard." IEEE Software **12**(5): 54 - 55.

Mistrik and Ivan (1984). Prototyping for Real-World Applicability Tests. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 232-243.

Mittermeir, R. T. (1982). "HIBOL, a language for fast prototyping in data processing environments." ACM SIGSOFT Software Engineering Notes **7**: 133-140.

Mock, M. T. and L. R. Hodge (1992). "An exercise to protoype the object-oriented development process." Software Engineering Journal **7**(2): 114-118.

Monckmeyer (1984). Concept and Experiences of Prototyping In a Software-Engineering-Environment With NATURAL. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 122-135.

Morrison, W. (1988). "Communicating witn users during systems development." Information & Software Technology **30**(5): 295-298.

Morton, M. S. S., Ed. (1990). The corporation of the 1990s: Information technology and organisational transformation. Oxford University Press.

Myers, B. (1995). "User Interface Software Tools." ACM Transactions on Computer-Human Interaction **2**(1): 60-103.

Myers, B., Ed. (1997). UIMSs, Toolkits, interface Builders. Handbook of User Interface Design.

Nelson, M. L. and R. B. Bymes (1992). "A Spiral Model of Object-Oriented Rapid Prototyping." TOOLS USA '92 (technology of Object-Oriented Languages and Systems). : 111-120.

Nelson, M. L. and A. David (1984). A Software Development Environment Emphasizing Rapid Prototyping. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 136-151.

Nosek, J. T. (1984). Organization Design Choices to Facilitate Evolutionary Development of Prototype Information Systems. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 341-355.

Osterweil, L. J. (1987). Software Processes are Software Too. 9th International Software Engineering Conference, IEEE Computer Society Press.

Ould, M. A. and C. Roberts (1986). Modelling iteration in the software process. 3rd International Software Process Workshop, Breckenridge, Colorado, USA,

Ould, M. A. and C. Roberts (1988). Defining formal models of the software development process. Software programming environments. Ellis-Horwood. 13-26.

Penedo, M. and C. Shu (1997). "Acquiring Experiences With the Modelling and Implementation of the Project Life-Cycle Process - the PMDB Work." Software Engineering Journal **6**(5): 259-274.

Pliskin, N., T. Romm, et al. (1993). "Presumed Versus Actual Organizational Culture: Managerial Implications for Implementation of Information Systems." The Computer Journal **36**(2): 143-152.

Pliskin, N. and P. Shoval (1989). "End user prototyping: sophisticated users suporting system development." Database **18**(4): 7-17.

Pliskin, N. and S. Shoval (1989). "Responsibility sharing between sophisticated users and professionals in structured prototyping." Information & Software Technology **31**(8): 438-448.

Pocock, J. N., Ed. (1991). VSF and its Relationship to Open Systems and Standard Repositories. Springer-Verlag.

Potts, C. (1993). "Software Engineering Research Revisited." IEEE

Software **10**(5): 19-28.

Pressman, R. S. (1992). Software engineering. A practitioners approach. McGraw-Hill.

Radice, R. A. and R. W. Phillips (1988). Software Engineering: An Industrial Approach. Prentice-Hall.

Redwine, S. T. (1988). Software reuse processes. 4th International Software Process Workshop, Moretonhampstead, England,

Reilly, J. P. and E. Carmel (1995). "Does RAD Live Up to the Hype?" IEEE Software **12**(5): 24 - 26.

Reisig, W. (1982). Petri-Nets. Springer Verlag.

Riddle, W. E. (1984). Advancing the State of the Art in Software Systems Prototyping. Approaches to Prototyping. 19-26.

Robinson, R. (1996). "Put the Rapid into RAD." Datamation **42**(4): 80.

Rombach, H. D. (1988). A specification framework for software processes: Formal specification and derivation of information base requirements. 4th International Software Process Workshop, Moretonhampstead, Devon, England,

Rombach, H. D. (1991). MVP-L: A language for process modeling in the large. University of Maryland.

Rossi, P. H., J. D. Wright, et al. (1983). Handbook of Survey Research. Academic Research.

Ruiz, F., F. v. Harmelen, et al. (1994). Evaluating a Formal Specification Language. Springer-Verlag.

Rumbaugh, J. (1991). Object-Oriented Modelling And Design. Prentice Hall International Editions.

Rzevski and George (1984). Prototypes versus Pilot Systems: Strategies for Evolutionary Information System Development. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 356-367.

Shlaer, S. and S. J. Mellor (1992). Object Lifecycles — Modelling the World in States. Yourdon Press Computing Series.

Shoval, P. and N. Pliskin (1988). "Structured prototyping : integrating prototyping into structured system development." Information Management **14**(1): 19-30.

Smith, M. F. (1991). Software Prototyping: Adoption, Practice and Management. England, McGraw-Hill.

Soong, X. and L. J. Osterweil (1991). Comparing design methodologies through process modelling. 1st IEEE International Conference on the Software Process, Redondo Beach, CA,

Spence, I. T. A. and B. N. Carey (1991). "Customers do not want frozen specifications." Software Engineering Journal **6**(4): 175-180.

Stahl, B. (1986). "The trouble with application generators." Datamation **32**(7): 93.

Steinbauer, D. (1991). A Repository and other tools in a commercial development center. Software Development Envrironments and CASE Technology, Konigwinter , Germany, Springer-Verlag.

Stephens, M. A. and P. E. Bate (1990). "Requirements engineering by prototyping: experiences in development of estimating system." Information & Software Technology **32**(4): 253-257.

Stroustrup, B. (1988). "What is Object-Oriented Programming." (5): 10-20.

Stroustrup, B. (1991). The C++ Programming Language. Addison Wesley.

Sugiyama, Y. and E. Horowitz (1991). "Building your own software development environment." Software Engineering Journal **6**(5): 317-330.

Tate, G. (1990). "Prototyping: helping to build the right software." Information & Software Technology **32**(4): 237-243.

Tate, G. and J. Verner (1990). "Case study of risk management, incremental development, and evolutionary prototyping." Information & Software Technology **32**(3): 207-214.

Tavolato, P. and K. Vincena (1984). A prototyping Methodology and its tools. Approaches to Prototyping, Berlin, Springer-Verlag.

Tozer, J. E. (1987). "Prototyping as a system development methodology: opportunities and pitfalls." Information & Software Technology **29**(5): 265-269.

Tracz, W., L. Coglianese, et al. (1993). "A Domain-Specific Software Architechture Engineering Process Outline." ACM SIGSOFT Software Engineering Notes **18**(2): 40-49.

Tully, C. J. (1986). Software process models and iteration. 3rd International Software Process Workshop, Breckenridge, Col, USA,

Turner, D. A. (1986). "An overview of Miranda." SIGPLAN Notices **21**(12): 158-166.

Van Hoeve, F. A. and R. Engmenn (1984). The TUBA-Project: a set of tools for application development and prototyping. Approaches to Prototyping, Berlin, Springer-Verlag.

Venken, R. and M. Bruynooghe (1984). Prolog as a Language for Prototyping of Information Systems. The Approaches to Prototyping. Namur, Springer-Verlag Berlin Heidelberg. 447-458.

Vonk, R. (1989). Prototyping: the effective use of CASE technology. London, Prentice Hall.

Walker, R. (1985). Applied Qualitative Research. Gower Publishing company.

Walsham, G. (1995). "Interpretive case studies in IS research: nature and method." European Journal of Information Systems 4(2): 74-81.

Whyte, W. F. (1984). Learning From the Field. Sage Publications, Inc.

Wileden, J. C. and M. Dowson (1986). "The software process and software environments." ACM SIGSOFT Software Engineering Notes **11**(4):

Winek, G. and V. Sriraman (1995). "Rapid rototyping : The state of the technology." Journal of Engineering Technology **12**(2): 34 - 43.

Yap, J. (1995). "Rapid Application Development in Small Unit Settings." International Journal of Bio-medical Computing **40**(2): 157 163.

Yeh, H. T. (1991). Re-engineering a software development process for fast delivery - approach and experiences. 1st IEEE International Conference on the Software Process, Redondo Beach, CA,

Yin, R. K. (1984). Case Study Research — Design and Methods. SAGE Publications.

APPENDICES

Appendix A Questionnaire on Software Prototyping

Questionnaire on Software Prototyping

This questionnaire aims to find out :

- whether your organisation makes use of software prototyping
- what, if any, problems you encounter with prototyping
- your opinion about where solutions or improvements are most needed
- your intention for further co-operation

Software prototyping has become a popular system development alternative to cope with the fast moving business needs. It is considered to be an iterative process which involves quickly constructing one or more working models(prototypes) of the whole or parts of the final system. Software prototyping is mainly used to serve at least one of the following purposes:

- exploring user requirements
- clarifying user requirements
- refining user requirements
- experimenting design alternatives and system requirements.

All information replies will be treated as completely confidential.

NAME: _____
JOB TITLE: _____
COMPANY:

_____

_____

1) Have you used software prototyping in your system development practice in last 5 years?

yes☐ no☐

(if not, could you please pass the questionnaire to somebody who has been involved)

2) For what application domains have you used (or are using) prototyping as part of the development process?

Database or information systems	yes☐ no☐
Networks and distributed systems	yes☐ no☐
Computer-integrated manufacture	yes☐ no☐
Real-time systems	yes☐ no☐
Digital telephone and switching	yes☐ no☐
Symbolic computation	yes☐ no☐
Safety critical systems	yes☐ no☐

3) Do you think that using prototyping within your organisation has resulted in?

Faster system development	yes☐ no☐
Improved reliability of delivered product	yes☐ no☐
Enhanced maintainability of delivered product	yes☐ no☐
Greater user satisfaction.	yes☐ no☐
Reduced development costs	yes☐ no☐
Increased productivity	yes☐ no☐
Better human computer interface	yes☐ no☐
Earlier user validation of the system	yes☐ no☐

Any other benefits?

4) Do you believe that there are benefits which have yet to be realised? If so, what?

5) Which, if any, of the following difficulties has your organisation experienced with software prototyping?

Lack of effective methods or guidance	yes☐ no☐
Inadequate tools and / or environments	yes☐ no☐
Lack of management support	yes☐ no☐
Prototyping process is hard to control	yes☐ no☐
Problems of communication and co-operation between users, prototypers, developers and managers	yes☐ no☐
Controlling product quality	yes☐ no☐

Any other problems ?

6) What do you consider to be the three most important problems that you have encountered with prototyping? Please indicate in decreasing order of severity.

1.

2.

3.

7) As part of my PhD research programme I am carrying out an investigation into software prototyping. Would you be

• interested in receiving results from this questionnaire / reports etc.
yes☐ no☐
• prepared to allow me to visit your site to meet with prototyping participants

yes☐ no☐

Contact address:

_____

_____

Phone: Fax:

E-Mail:

Thank you for your time and co-operation in completing this questionnaire. If you have any queries please do not hesitate to get in touch on (0202-595101 or fax 0202-595314).

Please return the questionnaire, if possible, by Friday 28st January, 1994 using the enclosed, stamped addressed envelope.

Liguang Chen
Bournemouth University
Poole, BH12 5BB

Appendix B Company Background Information

reference	business area	site size* of organization	main app. domain or project type	configuration of project participants	main purpose of prototyping	percentage of prototyping effort over whole project effort
process 1	international banking	large	info system	total about 8 participants: 3 to 5 development team(including manager(s) and other roles) + 2 to 3 customer and end-users	user's requirements	above 60%
process 2	airway service	medium	info system	total about30 participants: 3 to 5 development team(including manager(s) and other roles) + 20 to 30 customer and end-users	user's and/or system's requirements	above 60%
process 3	software house	small	info system	total about15 participants: 1 developer + 10 to 15 customer and end-users	user's and system's requirements	above 60%
process 4	hotel service	small	staff scheduling	total about5 participants: 1 developer + 4 customer and end-users	user's requirements	above 60%
process 5	university computer centre	small	network monitoring	total about4 participants: 1 developer + 3 customer and end-users	user's and system's requirements	above 60%
process 6	air traffic control	medium	air traffic control	unknown	risk analysis + system's requirements	5-10%

process 7	electronic engineering	medium	circuit testing	total about 15 participants: 1 prototyper + 10-15 others (managers, engineers and proving staffs) + 1 or 2 customer representatives	interface design	5-10%
process 8	tele communication	large	intelligent networking	total about 80 participants: 12 prototypers + about 65 others (managers, designers, engineers, proving and marketing staff) + 1 or 2 customer representatives	system design and/or implementation	10-30%
process 9	tele communication	large	intelligent networking	total about 30 participants: 9 prototyper + 20 others(managers, proving and commercial staff) + 1 or 2 customer representatives	risk analysis and/or system design	10-15%
process 10	publishing	large	info system	total about 10 participants: 1 project manager, 1 design manager, 3 developer; 2 customer + 3 users	system design	N/A

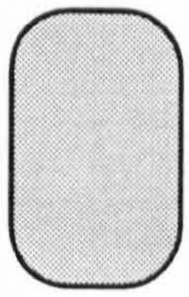
---

* size here is a rough estimate of the number of system development staff within the site of an organisation, which is categorised in three scales:

large: 150 above; medium: 50 — 150; small: under 50.

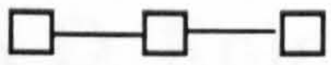
Appendix C Role Activity Diagrams (RADs) Notations

Role Activity Diagrams (RADs) Notations



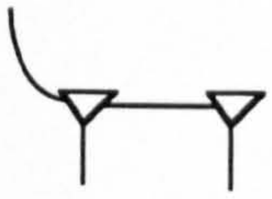
represents a role

indicates an interaction between two roles



indicates an interaction among three roles

represents the activity performed by a role



indicates an alternative path

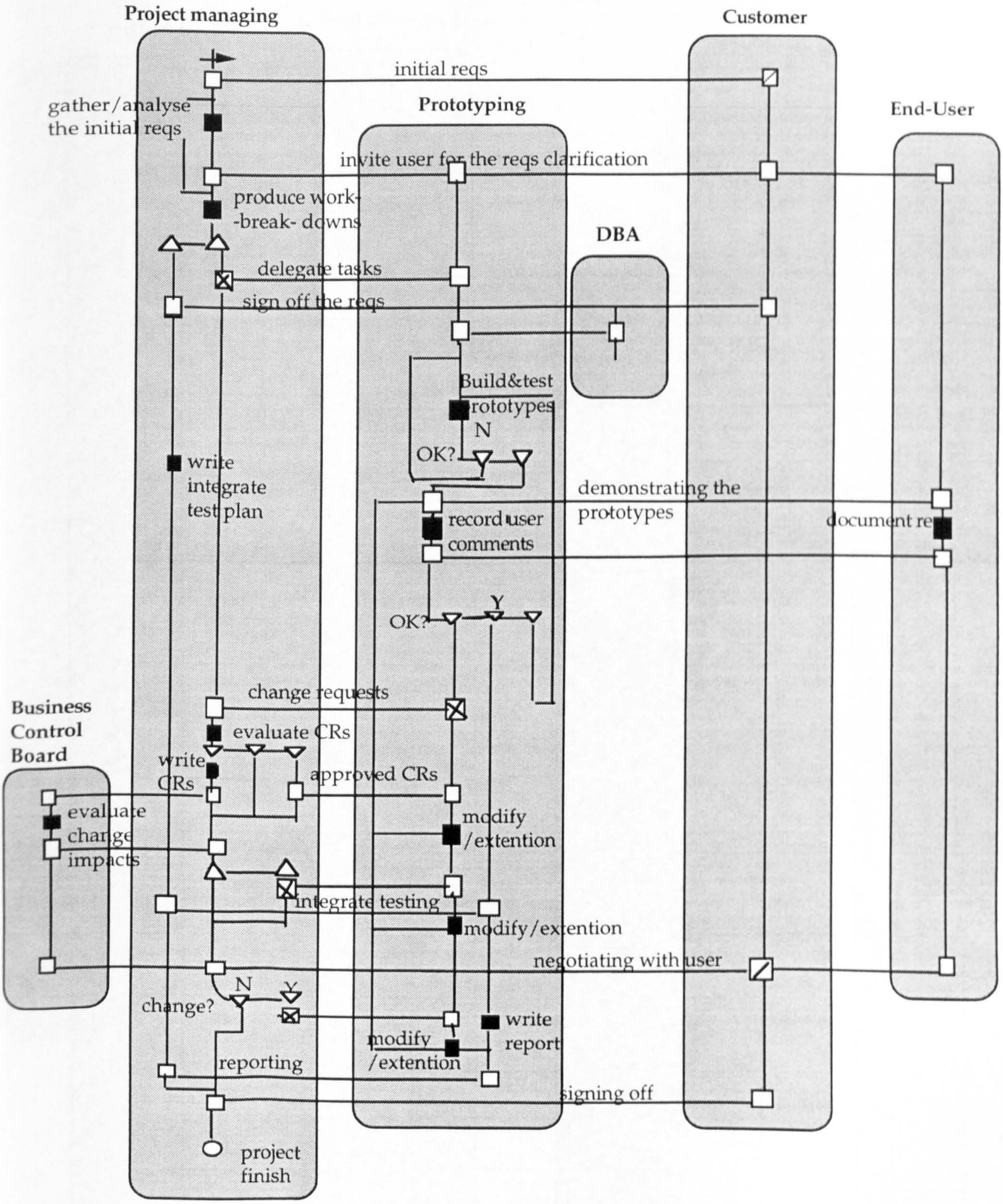
indicates a parellel path

indicates a state of the process

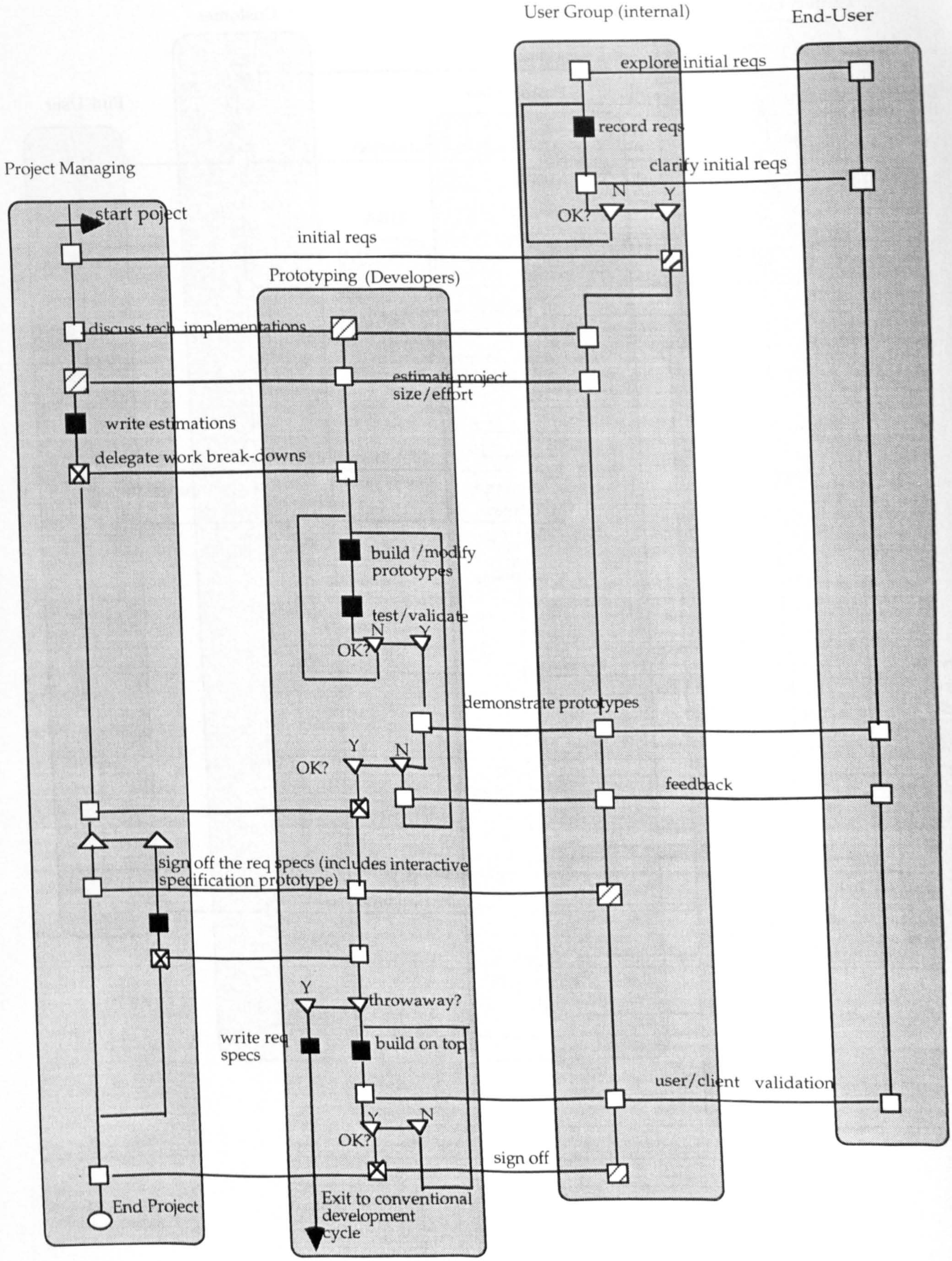
indicate a role switching

Appendix D Role Activity Diagrams of the 10 Processes

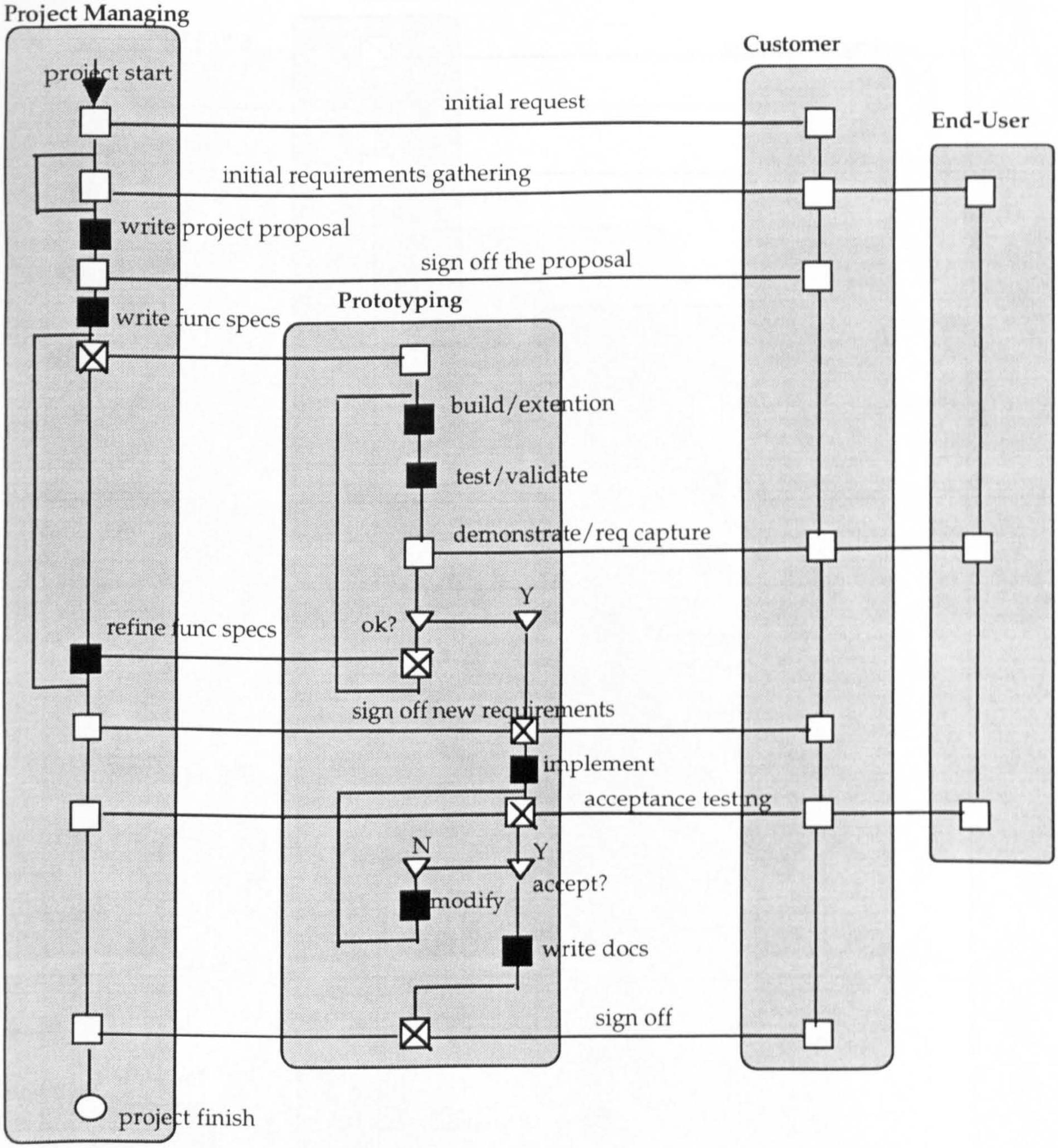
RADs of Process 1 (Company B)



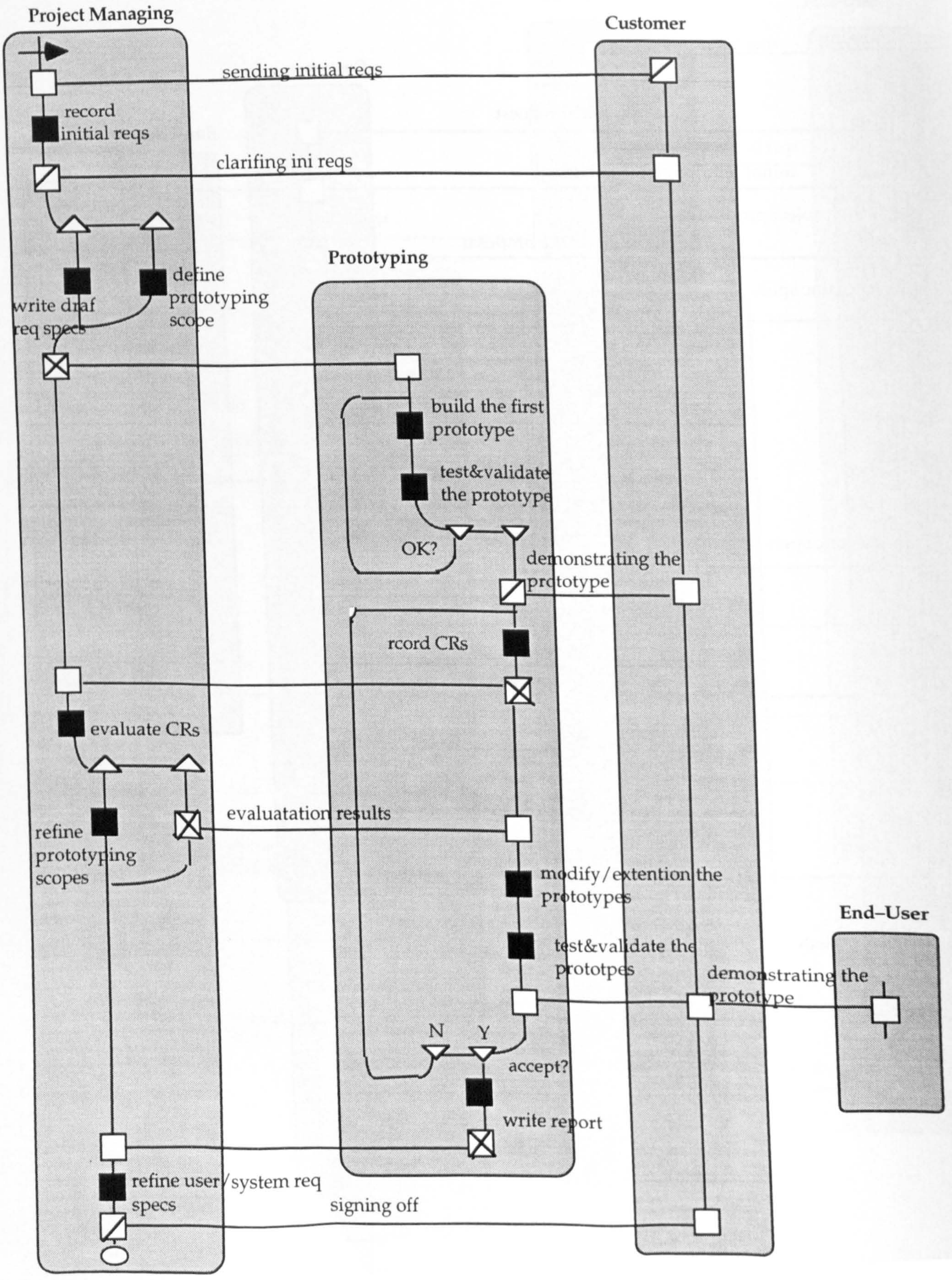
RADs of process 2 (Company A)



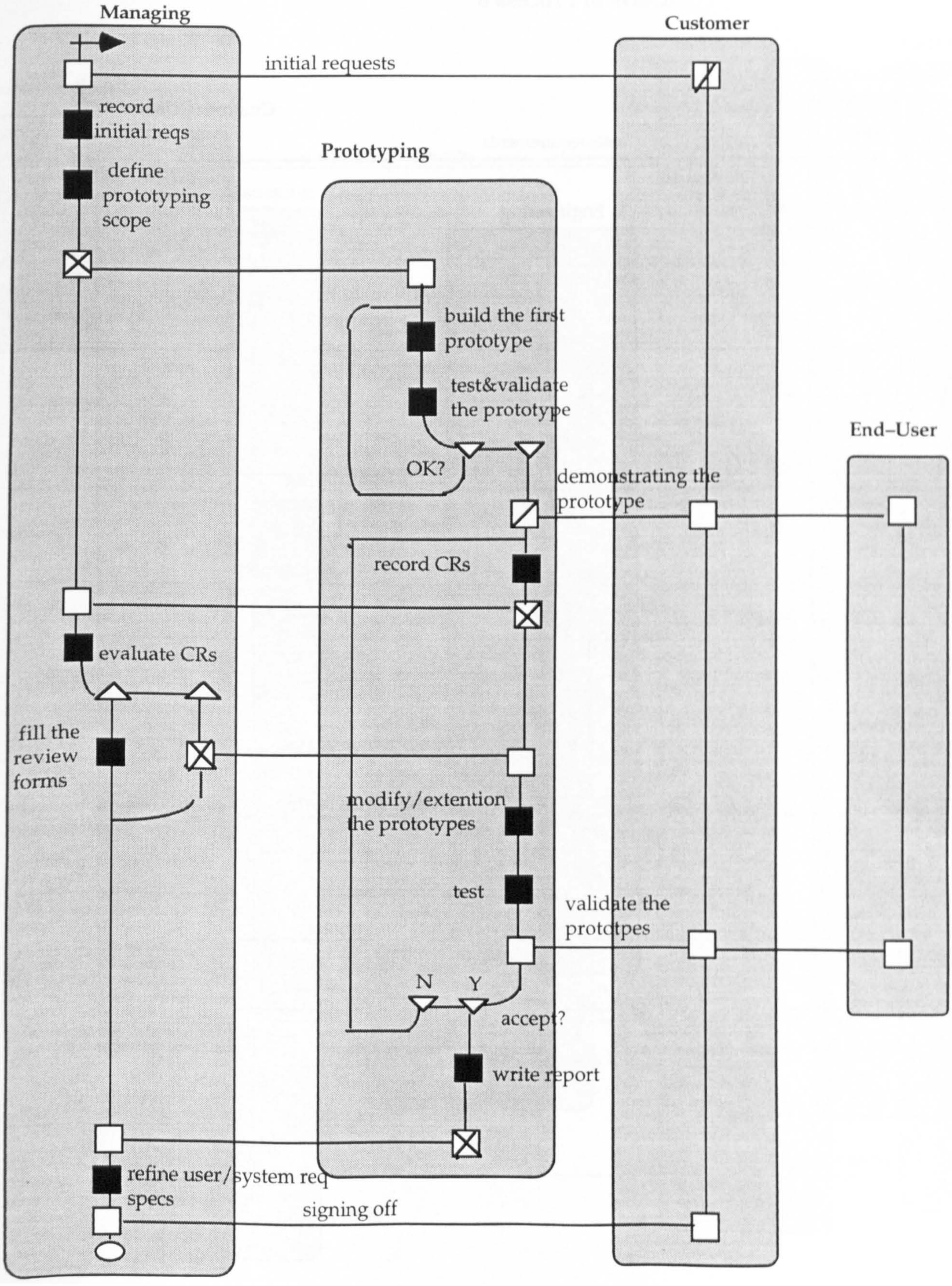
RADs of Process 3 (Company D)



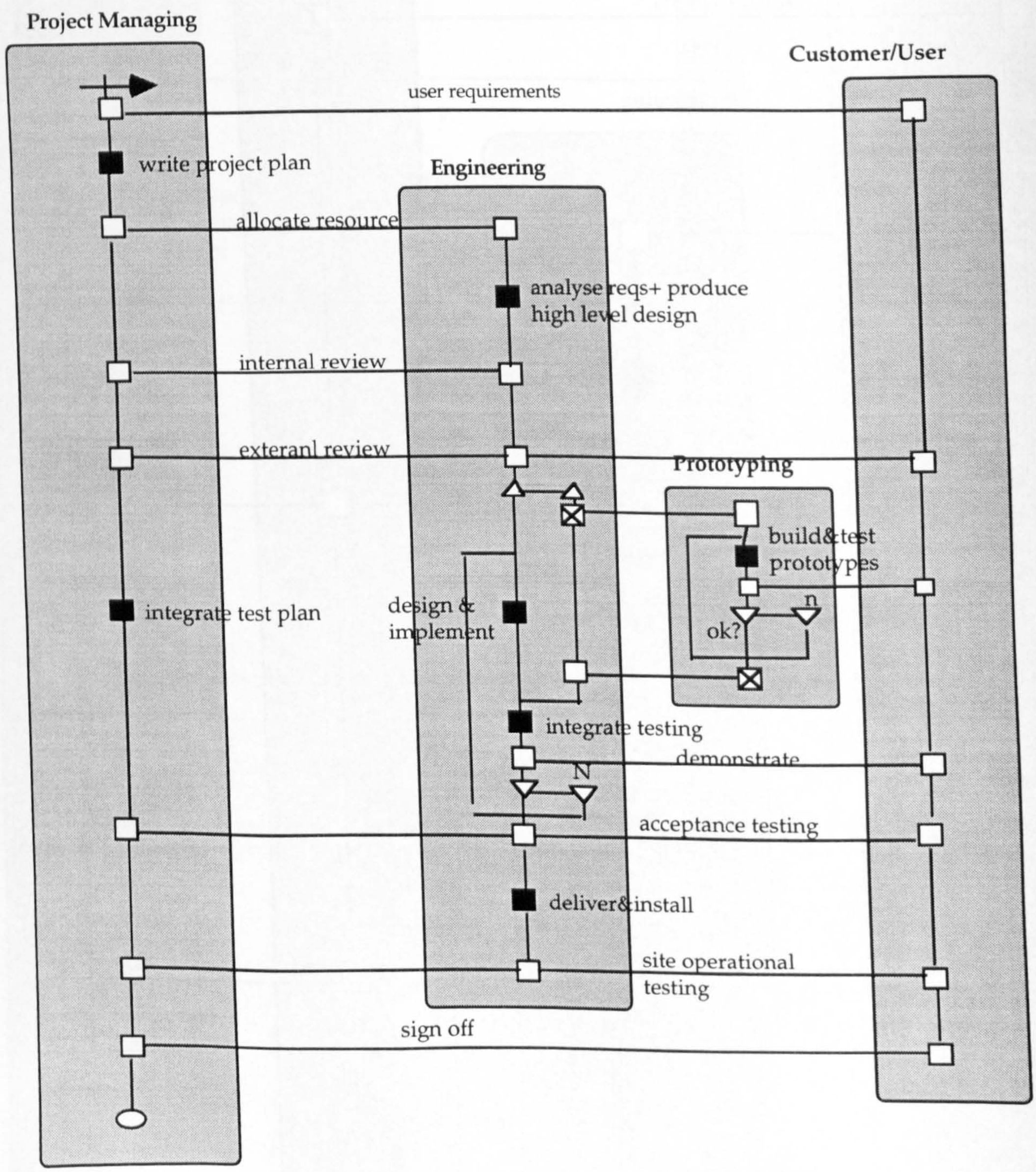
RADs of process 4



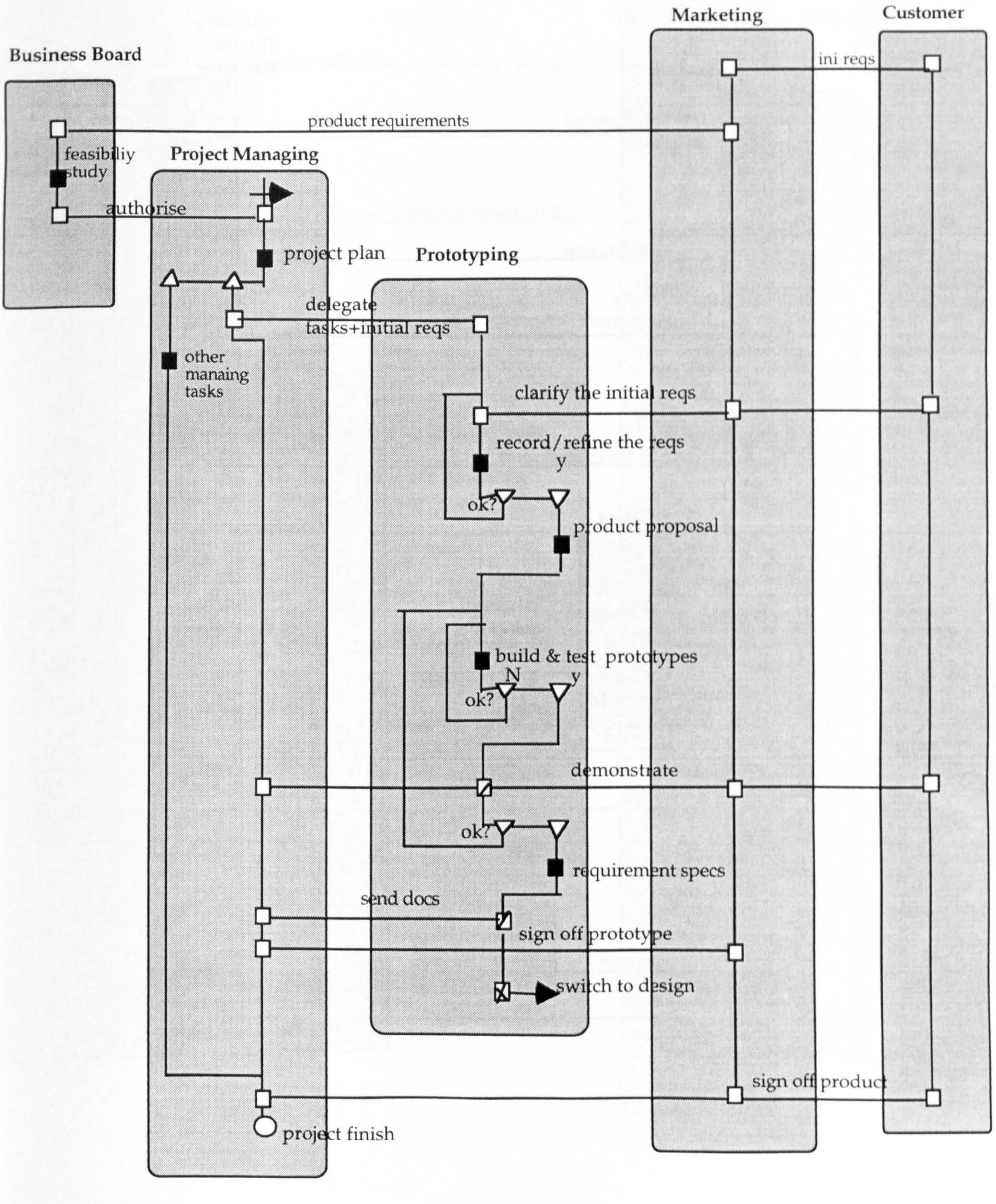
RADs of Process 5



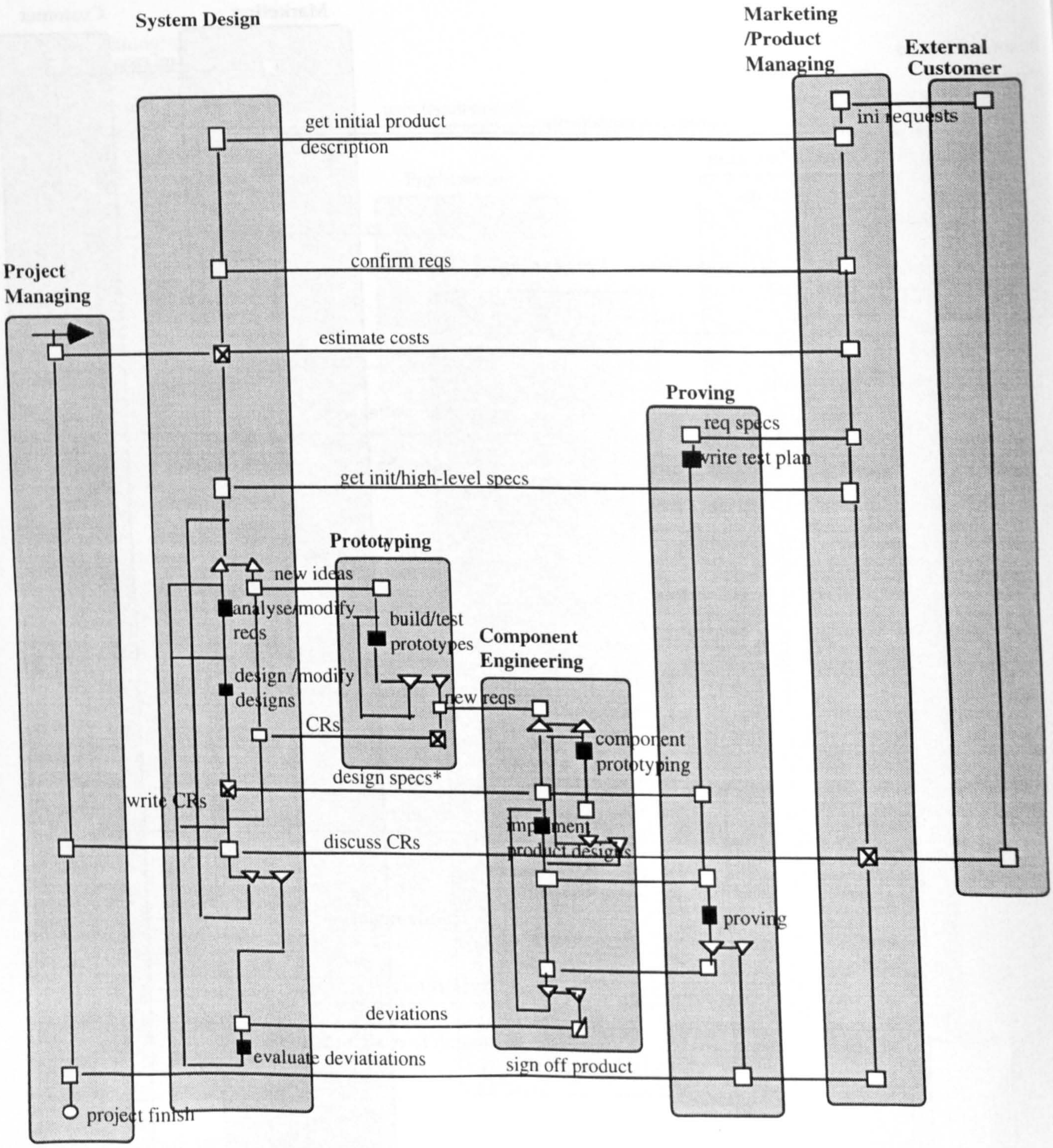
RADs of Process 6



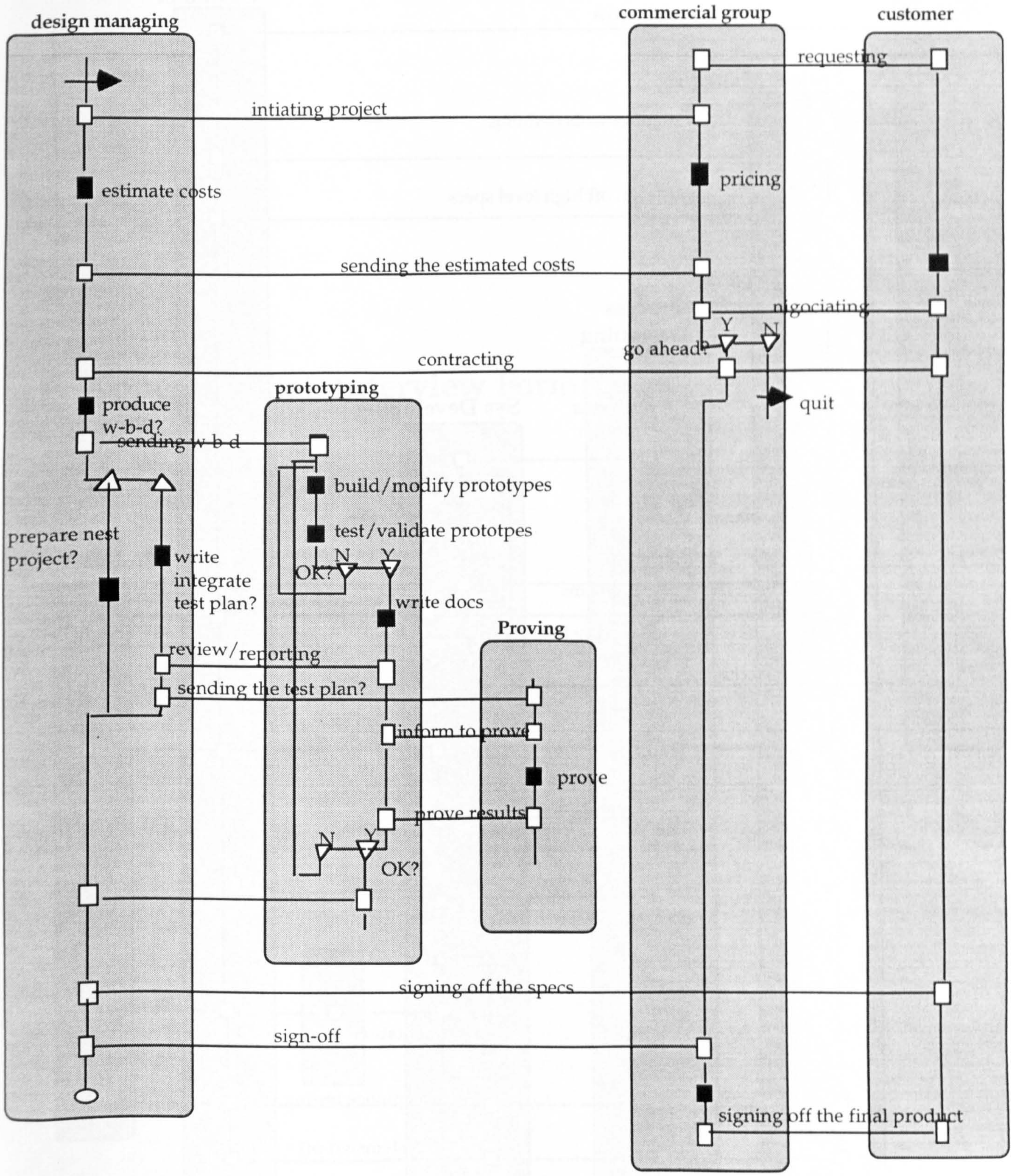
RADs of Process 7



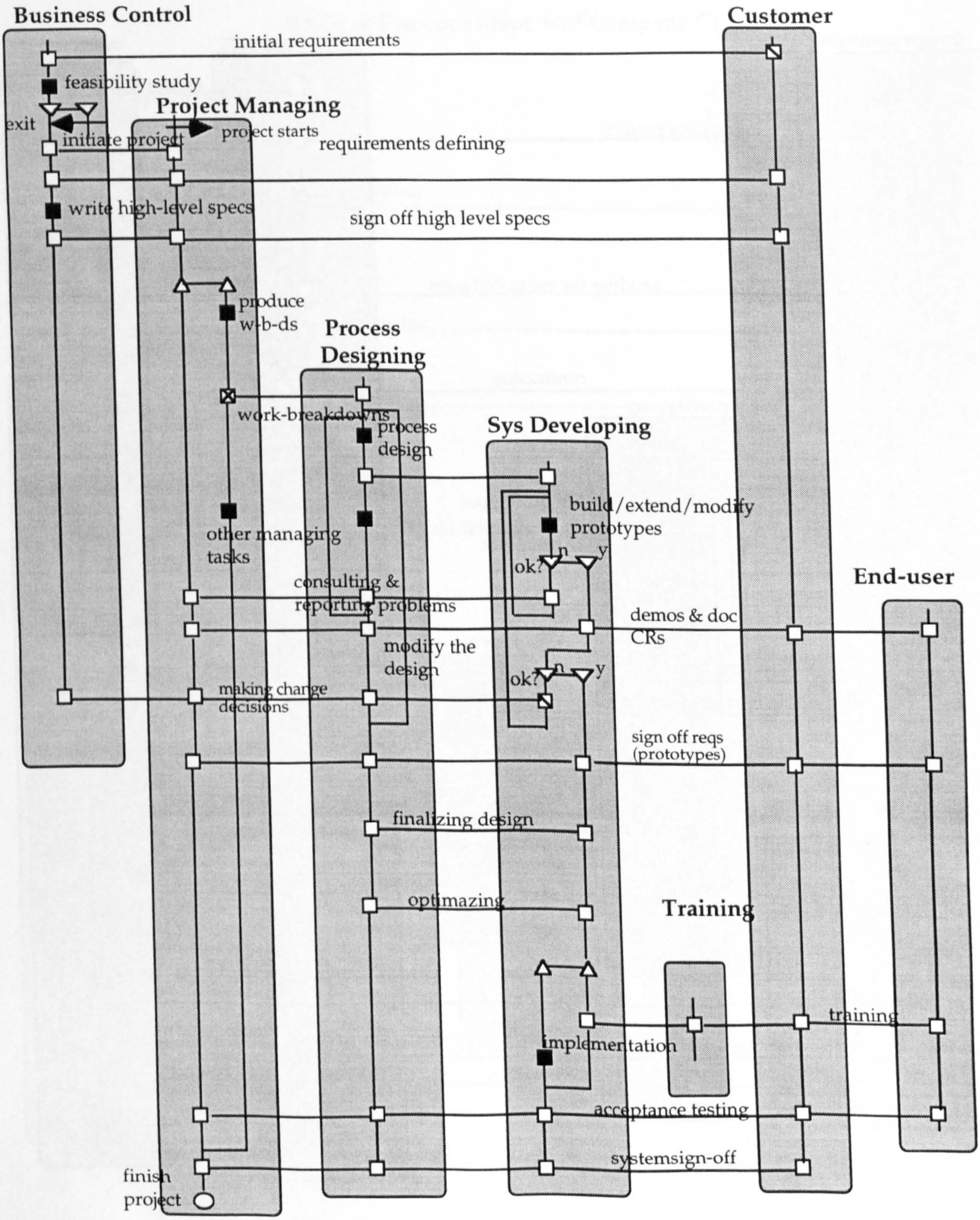
RADs of Process 8 (dept. b of Company C)



RADs of Process 9 (dept. a of Company C)



RADs of Process 10 (Company E)



Appendix E(a) Interview Form Template

Interview Form Template	
1 Meeting Details	
Date	
Duration	
Location	
Reference	
Department	
2 Interviewee Brief	
Name Reference	
Job Title	
Process Role	
Other Information	
3 Meeting Agenda	
3.1 Purpose of the Meeting	
3.2 Following-up Plan	
3.3 Data Gathering Methods	
4 Summary	
4.1 Information Gathered	
The information gathered is arranged in following:	
4.1.1 The RAD Model	
4.1.2 The Personality Test Results	
Reference	**Personality type**
4.1.3 The Key Questions asked and Answers replied	
Questions	**Answers**
4.1.4 Other Information	
4.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**
4.2 Follow-up Agreement	
4.3 Comments on the meeting	
4.3.1 About The RAD model	
4.3.2 About The Personality Test	
4.3.3 About The Questions and Answers	
4.3.4 About Other Information	
4.3.5 About The Supporting Documents	
4.4 Key Issues Raised from the meeting	

Appendix E(b) Interview Summaries

Interview Form A1

Some sections are left blank if no appropriate information to fill in.

1 Meeting Details

Date	24/5/1994
Duration	60 minutes
Company Reference	A
Department	Rapid Adaptive Development
Meeting Reference	A1

2 Interviewee Brief

Interviewee Reference	BC
Job Title	Rapid Adaptive Manager
Process Role	Project Managing, prototyping
Other Information	

3 Meeting Agenda and Methods

3.1 Purpose of the Meeting

a. Introducing my research

b. Discussing the co-operation

c. Get a draft RADs process model

3.2 Following-up Plan

a. Arrange a date for clarification

b. Explain the potential benefits

3.3 Data Gathering Methods

a. Interviewing with taking notes

b. RADs Modelling

4 Summary

4.1 Information Gathered

4.1.1 The RADs Model

A draft RADs model was derived by going through their normal practice — from the beginning to finish of a typical prototyping project, which indicates the main roles, activities and the interactions of the process. This was further tidied up using a drawing package and referred as the daft RADs. 4.1.3 shows the key questions asked and the answers in deriving the draft model.

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied

The following questions are asked in order to model the prototyping practice in term of role, activity and interaction of the process.

Questions	Answers
a. What are the distinct roles involved in a typical prototyping project?	There are four in our RAD unit, which are project manager, developers, user group and end-users.
b. What is the function of the user group ?	Here the user group is a department within the company, it function as if it were a external customer to us.
c. How is a project initiated?	The initial request normally come from the user group to me (the manager of the RAD unit) via e-mail or telephone or informal meeting.
d. How are initial requirements gathered?	Normally the user group will gather initial request from or explore the area for improvement with the end-users. The initial requirements will then pass to the RAD manager, who will subsequently allocate the work to a suitable person. We (the developer and customer and I) will further clarify the initial requirements and discuss the implementations.
e. What do you need to do before the prototyper starting to build the prototype(s)?	I need to write the project estimates, and find the right person to do the job. I will then pass the initial requirements to the developer if it is in written form otherwise build the contact between the developer and the customer/end-user.
f. What are involved in the first prototype building ?	It involves a construct the mock-ups, do a quick design make it work quickly.
g. What and who is involved in the prototype(s) evaluated?	Normally the prototypers, the customers and end-users.
h. Do you sign off the prototypes with customer as (or as part of) specifications?	Yes, we do.
i. Any change request come to you either from the prototyper or user group or end users?	The change requests normally come from the user representative or the end-users to the developers.

j. What and who involved in the decision making?	As I am often involved in the development, the decisions are informally made at the time by discuss with the developers.
k. Who involves in the final sign off?	The project manager and the user representative.
l. Could you indicate any other explicit controls on the diagram?	I, as the manager, need to do some estimation work (time/effort) just after the initial requirement gathering. The estimates is done by me and agreed by the user representative.
4.1.4 Other Information Mr BC is leaving the company. He recommend KR for future contact.	
4.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**
4.2 Follow-up Agreement **a.** I post him the first version (tidy-upped version of the draft) and he send me back the comments **b.** Mr KR is recommended for future contact[1] **c.** I will give them a summary report at the end of the field modelling	
4.3 Comments on the meeting The interview was satisfied in term of the purpose sought out, although it would be better if the draft could be checked and refined in person instead of posting the comments. The impression was they did not have any explicit CRs control method or guidelines and this was done often by informal discussions between the manager and prototyper. When asked quality control he said he and his team members did not like ticking box, also he felt there was not much point in doing so.	
4.3.1 About The RAD model	
4.3.2 About The Personality Test	
4.3.3 About The Questions and Answers	
4.3.4 About Other Information	
4.3.5 About The Supporting Documents	
4.4 Key Issues Raised from the meeting **a.** Are there any need for methods and or guidelines for CRs control, why? **b.** Is it practical possible for developing such a method/guidelines? if not what are the difficulties? **c.** In what way and in what sense the product quality is guaranteed?	

Interview Form A2

1 Meeting Details

Date	11/11/1994
Duration	90 minutes
Company Reference	A
Department	Rapid Application Development
Meeting Reference	A2

2 Interviewee Brief

Interviewee Reference	KR
Job Title	Senior Manager
Process Role	Project Managing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Review the RAD model

b. Get a more detailed view of the process

c. Ask pre-compiled questions

d. Gather some supporting documents

3.2 Follow-up Plan

Fix a date for next visit which will cover:

a. A brief interview of least two software developers

b. Carry out the personality test

c. Collect some documents

3.3 Data Gathering Methods

a. Interview with tape recording

b. Comments on the RAD model

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

Having walked-through and discussed the RAD model with me, he pointed out two unclear places on the diagrams:

(1) customer and/or users may be involve in prototype building;

(2) the involvement of the internal user group was not a must — some times managers or prototypers contact the customer and or user directly or vice versa.

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 <u>The Key Questions asked and Answers replied</u> Based on the RAD model which was derived from previous interviews, the following questions were asked in order to find out more about their process. The answers here are all from KR, which are partially rephrased and re-organised for the clarity.	
Questions	**Answers**
a. Could you please briefly tell me more about you prototyping practice? and first of all, how is your prototyping team made up?	Our RAD unit consists of highly professional staff, and end-users will have continuous involvement throughout a project. If you go back a number of years ago, we used to have a team of three to five developers working on same project in one application area. But, for various reasons, we have changed our project team; it primarily has comprise of a one person team for our business need; occasionally we have a team of two but not more than two developers. Sometimes one person may be involved in more than one project at one time. If a project comes up, I will chose the person with the particularly skill required for that project. In fact, we have one or two people work on each of the business area for least two or three years, so it will almost always the same people take up the project in that area during this period. Of course, every two or three years we'll move them into different area from career development point view.
b. What, if any, differences are there compared with traditional development?	Yes, I think, there are. Our main concern is users' satisfaction — put a system in use and make it usable, not to write nice piece of design or code — whereas in IT department their satisfaction comes very much from solving the technical problem in hand.

	Another difference is that we have much more time and cost constraints than the IT department do. Our staff are more commercially oriented, they talk business language with users rather than IT language.
c. What about the procedures and guidelines for prototyping?	We are still working on this. In fact, if you ask any our staff here they won't know much about it, because we don't have any formal procedures or guidelines. But our company is a member of the consortium of companies. The consortium is trying to put together industry-wide methods and standards for rapid development. We are actively involved in the work, for example, the DSDM consortium has just published a draft version of the RAD principles (for detail see the attached sheets), I have given this to my staff, but we haven't worked out how it will fit in here for us. And here I also have the daft framework of the methods (unfortunately confidential).
d. It looks like a set of very basic principles, do you find it is useful?	Yes, it is very skeleton at the moment, more details will go into the future methods. We do find it is useful.
e. What about the method for conventional development?	We actually have a company-wide method for conventional development called DIP—Development and Improvement Programme. Most of the main development groups are the DIP compliance, who have to work by the method. This was considered as a massive burden and not welcome by the compliance although that has been changed over last six month, they now recognise some the value by using the method. However, you will still find people talk about it as an imposition rather than something they think as practical useful

	and willing to grasp. So it is still a long way to go to be mature. And I have made it clear that because the tight time scale that RAD has, it is impractical to impose upon our RAD team such a methods and standards like the DIP, which will unduly affect its ability to response rapidly, and also the fact is that our developers are driven by the passion of working for customers. Therefore the principles of our methods or standards has to be a liberating and energising force rather than an imposition that generates any unnecessary over-heads upon them, in other words, what is the best practice, what really works around here, what actually works for customer.
f. What, if any, clearly defined responsibilities for managers, developers and customers or users?	We are not as explicit as we ought to be in doing that. But, it has to be said that all our staff here are very experienced, they have very good intuition and they know what they are doing, and have a good grasp of what sort of thing should come back to me. Again, with developing our own RAD standards and methods, those responsibilities will be included in.
g. Could you tell me more about the RAD unit?	The RAD was originally built, about two years ago, out of mixed disciplines: some of a industrial engineering group; some from an artificial intelligence group; and some from operational research group, and is established formally less than one year old. At the moment we have a dozen developers.

h. How are the initial requirements gathered?	Typically an initial request will arrive by telephone or e-mail, and we will talk to the potential customer who can be a colleague in information management or a customer in his own right. We will then, in general, gather the requirements in order to make an estimate of how important the work is and therefore where it should sit in our portfolio. We have to make a couple of assessments: one is to decide the importance of the work and another is the technical requirement, i.e. who can do it in term of the technical skill and his availability. Then, make an estimate of the scope and size of the work, i.e. how much commitment and therefore how much the cost and time will be needed. Normally the initial enquiry is verbal and informal which might come through me or any one of the staff. Sometimes an customer submits a written request, and other times we write an initial requirements according to what being asked, or sometimes even produce a mock-up of the requested system to show them what we would be able to do. Of course, this all depends very much on what kind of relationship that we've built already. For example, there are a few cases, where we haven't had any historical relationship with them who come to commission their work;
	we started to talk to them the RAD approach that would be taken, then they said they don't know what we were talking about. So we invite them to join a mock-up session, and told them that's what we think you want to us to build, and asked if we were along the right direction.

	So, in those case, a couple of my staff sitting in front of the screen with the customers (user group), and in the background were some potential users. This often happens for those work involving much novelty, and normally we go around a couple of times. [refer to the RAD model, the very first loop].
	If the initial requirements are pretty clear and the work is important, and we have right person for the job, we will straight go on into the prototype building stage [refer to the second loop on the RAD model] and the analysis and design will be within the loop.
i. Any guidelines at this stage?	As to the guidelines, as I said before, we don't have a standard procedures yet, if you talk to my staff you will find that different people do it differently, partly because the technology they use, partly because the different background.
j. What about the prototype building phase?	Well, there are two way look at it. One way is just as if you were building a system traditionally but with fast speed, I call this prototype a minimum useful working subset of the total system. So you do ENOUGH work up here in requirement analysis to show what is the total system you are going to build, and what that system will do for customer; you then work out which part you need to build first—the bit which is the most useful and fast to build. You then build that and give it to the customer, something which they can use and
	benefit from it, and from the result of using it we can learn a great deal about what they really want, you can then update your idea about the system in the longer term and decide the next module to build.

	Another way of doing it is by using the concept called "time-boxing". That is to say first choose a relatively arbitrary time and then build whatever you can build during that period time. The difference with the other way is that you work out the 'minimum useful working set' based on the pre-fixed time scale. This requires, I think, a very different mind set from the mind set which says: do the requirement analysis, do the functional design, and do the coding and testing, as the other way I mentioned earlier on. In another word, the time-boxing approach is more product focused, and that the problem a lot easier to solve. I very much prefer the latter, but you'll find most our staff are using the former approach.
k. Do you think the analysis and design are two distinct activities within your prototyping construction?	Well, our consortium do regard them as distinct activities, but if you work in a time-boxed way the stage line will definitely blur. I have to make it clear that, most of the time, our works to do is more or less half-done: the staff working on operational system which generates a lot of data typically used for decision support systems which are the bread and butter work of the RAD unit. We, in general, interface into large databases, so we don't have to do much design work from that point of view, we take a lot on board already.
l. Do you have any controls over quality?	Not really. Because our prototypes are by no means throwaways, I do think the quality controls on things like the validity and the maintainability of the built system are important which we haven't address yet. And also that is the part have so far least addressed in the methods our consortium working on. Of course, all our staff know that somebody else might take over his work or vice visa, so

	they do sort of quality concerns and actually do something about it. But we do often have the intuitions where somebody take over a project and say: "gosh, I don't understand this; I want to rewrite the whole thing".
m. How about the DBA roles? How are they involved?	We have a co-operative data group who are very actively involved in our RAD unit. They are really involved in laying down the standards by which we work. They are not directly involving in prototyping project instead they have a regular meeting with us about every six weeks.
	And their role is not just data administration but also in supporting us doing library management, so in the meeting we can talk about what library developments we've done in using the data they provided and they can to us about the improvements in the coverage of the co-operative databases which we have access to.
n. What are the typical changes that occurred?	There are two classes change requests I get: one is that the customer want change their systems because of the business changes: it may be either the external ones in that they've changed the way they work with BA or the internal changes; another is more domestic in nature: they want to change the way the data is organised and represented on screen.
o. What about the CRs controls?	Normally the change decisions are made by the developers. There are a lot of feedback from customer and/or users, because we regard them as part of the team, they are heavily involved in the actual prototype building process [see the RAD model: the activity 'build prototypes']. Because we work in a very short time period and having very close relationship with customer, there is rarely any point in

	making decisions formally. Each decisions is rather small and almost invisible to an organisation because of the incremental nature of the process whereas in traditional way of decision making is always seemingly large ones involving lot of money. But I have to say the quality of the work and the decision making is bit of a unknown.
p. How does the CR controls differ from your conventional project?	For the large project that I know they have a monthly report to the managing director of the company reporting their progress on that. I don't know if a RAD project have ever figured on it, because it often is a invisible part of a big system. The difference with traditional developments is we don't go through the committee meeting which happens monthly or quarterly. It's OK if you have a project that lasts a couple of years and don't have to deliver visible values quickly.

4.1.4 Other Information

KR: there are about two thousand computing staff in the company, and probably just under one thousand of them are software developers who mainly work on building large mainframe database applications for our customers, particularly operational systems such as RC reservations computing, ticket processing system, revenue processing system, our engineering system and operation control system. All of those are big production systems built with 3GLs or even more primitive. They generate large amounts of data for decision support, and that's where RAD comes in.

KR: when I say customers I mean internal client. But we still charge our clients as if they were external.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**
a. The RAD principles (draft)	13 principles for the Rapid Application Development

4.2 Follow-up Agreement

a. Arrange a meeting for me to see some developers

b. Do the personality test

c. Try to find some relevant documents for me

4.3 Comments on the meeting This was the second visit to the Rapid Application development of the company and the first time with KR. We started by checking and discussing the first version RADs model which was derived from the draft version with BC and his feedback comments.
4.3.1 About The RAD model Apart from the problems he recognised as in 4.1.1 he agreed that the RAD model is a fair representation of their prototyping practice.
4.3.2 About The Personality Test
4.3.3 About The Questions and Answers
4.3.4 About Other Information
4.3.5 About The Supporting Documents
4.5 Key Issues Raised from the meeting **a.** It appears the IT department has a very different work emphasis with the RAD unit. Is it true elsewhere? i.e. does the IT department generally have very different way of working compared with commercial settings?, and therefore different treatment?
b. The RAD unit has an interface with their conventional development team. How do they work together?(any defined roles and responsibility?) what is criteria for determining the part of a system to prototype?

Interview Form A3

1 Meeting Details

Date	25/11/94
Duration	120 minutes
Company Reference	A
Department	Rapid Adaptive Development
Meeting Reference	A3

2 Interviewee Brief

Interviewee Reference	KR
Job Title	Senior Manager
Process Role	Project managing
Other Information	
Interviewee Reference	JL
Job Title	Knowledge engineer
Process Role	Project managing, prototyping
Other Information	
Interviewee Reference	BT
Job Title	Knowledge engineer
Process Role	Project managing, prototyping
Other Information	
Interviewee Reference	RK
Job Title	Business Consultant
Process Role	Project managing, prototyping
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. make an enquiry based on the pre-compiled question guidelines for further investigation

b. carry out the personality test

c. gather relevant documents

3.2 Following-up Plan

This is the last visit intended, no plan for further visit.

3.3 Data Gathering Methods

a. Record the conversation

b. complete the personality test results form

c. Collect the supporting documents

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	Personality type
KR	ENFP
JL	ENTJ
BC	ENTP
RK	ENTP

4.1.3 The Key Questions asked and Answers replied

Based on the RAD model which derived from previous interviews, the following questions were asked in order to find out more their process. All the answers from the interviewee are partially rephrased and re-organised for the clarity.

Questions	Answers
a. From what I gathered with KR, I understood that you haven't got very clearly defined methods and nor any procedures and guideline in place. How do you cope with this situation?	JL: We are beginning to have a method now, I've got a second draft of it and I can give you a copy later. In my opinion, our general practice here is that the analysis and design dose not happen beforehand, we use prototyping as a way of demonstrating, testing and proving the concept we got on our mind, it's certainly the case in the current project that BC and RK undertaking. RK: it is trust really. If the customer and the project manager trust you, then you can cut the corners; for example, in JL's case, if the manager give the trust and do a bit more reorganise, then your work can be done at that time.
	JL: that is why in RAD we prefer to do a stand-alone project.
b. Could you have a brief description of your typical practice?	JL: I begin with some contacts with customer where the ideas normally come from, the ideas, of course, might come from us or other business area. We just generally talk about ideas—what they need or what I can do for them.

	Then, I think the first and very important thing is to get something together quick, which looks good and will effectively get your foot in the door, because you want to show them something which is really impressive, does something, even better if you can present them with real data.
	Now you can show them and tell them that you can use it now, and it will help straight away. It's like a sales call, a demonstration of what you can do with it in future. You might not be able to do that straight away, but you might be able to show them some previous similar examples. What I then do is to tell them at each stage we agree that before we meet again, say two weeks time, what I'll do in between.
	I have some principles in mind: (1) keep the time short [the session with customer]; (2) keep the task small and manageable; (3) do not be afraid to ring up customer or end-user if something is unclear during the prototype building, and (4) meet regularly, I tend to meet once or even twice in a couple of days.
	So the analysis and design is done in a very small and manageable chunks.
c. What, in your opinion, are the most important things in the process?	JL: I think, at first when I start doing this sort of work, to build something quickly was the key. Now I begin to think that if the system involves lots of data or variables, I'll then try to design some future instances so I won't need to change them often in future. You can do thing very quick and dirty, and you can get away with it, but you'll cause problems later. So to avoid it you ought to put some thought earlier on, i.e. do some analysis and design about how things functioning inside. So if you are in a project which is not going very far, you can do it quick and dirty.

c1. It is interesting that you said "you can get away with it", how is that?	Because we haven't have any quality audit yet. but I'm sure it will be.
	RK: I think what JL has said is more or less the way we work here. Just a few points I want to make. One thing is you have to bear future in mind. i.e. think about the interaction with other systems and the technology changes. Another important thing is the data modelling, for instance, we do it in our current project though I don't think it is part of RAD, but we have to do it, no one else takes the responsibility for it. So I think the RAD people not only the RAD people but also the traditional development people. BC: Something else JL said specifying a date and sticking to that date is very important. When you first meet the customer, he gives you a wish-list he would like to see at the end of it,
	but all the due agree to is to have a best version by a certain date, and that best version will include that wish-list defined at beginning. Then at least the customer will happy in his mind, that something is happening and will contain his wish-list. At that point he may decide that there are some other wishes are also important.
c2. so, is it very much the time-boxing way as KR described?	BC and RK: it is very much so. And we do think it is good to always deliver something on time as KR always emphasised. JL: That's one thing KR has been right about!

	RK: Another important thing JL said about the visual thing. Sometimes you spend a lot time building something behind the scenes they don't give you credit, while if you put on some snazzy box, they will say: 'wow!' But it's often only taken a few minutes to do it. It's a selling point, it buys you in.
	JL: It is a very important aspect to make it visually impressive, because if you have a impressive start you'll be impressive all the way through. We are **enthusiasts**, we believe in the RAD approach, we like the work we do: it's fast-moving and changing , it is exciting. And also we have to be a salesman all the time, for example, we tick the box of the wish-list at the end of each time-box and we may add on some new ones and re-prioritise for next time-box.
d. How are you actually conducting the initial requirement gathering?	RK: It's depends on customers really, normally customer will get very annoyed if you go on and on abut the requirements, so it is better to
	get a bit of the requirements and do it quickly, and then show them; obviously you might miss some points; they might say they don't want this bit or that bit; you then, carry on, change them and/or build more. This is always best to give something and show something quickly.
	BC: Just like our current project, at the beginning he seemed very sure what he wanted, but when we showed the first prototype, he then said: 'no I don't want that, what I meant is actually this...'
	RK: We normally sit down with our customer in front of the screen for an hour or two session.

d1. Do you take notes or just do a mock-up and change it as you go along?	RK: If it is minor changes we'll just change it. Of course, there is danger in doing that because something that seems trivial might cause problem later. So it's best to take notes and show them later if you want play safe.
e. Do you think there is the need for some sort of formal controls on the CRs [change requests]?	BC: Yes, I think there is need for certain level of bureaucracy from the control point of view.
	JL: Yes, if it is something simple like getting clear the responsibilities and doing some sort of signing off at some points, rather than long-winded procedures. And we think it's very essential indeed. I know that we are doing something like that in our RAD methods, and it will go into the Information Management Standard Book. I tell you that the IMS book is a very very boring book.
	RK: I think it is nice to have that sort of simple procedures or guidelines as long as it is flexible. That is to say we also need procedures on the use of the procedures. The DSDM is a very general document about the RAD methodology, I think what should happen is that we have a local methods like our DIP process, something is workable with us, and we shouldn't stick to a general method, it's just too boring, too large, and takes time to read it. Because the RAD project is relatively small, what we really need is some guidelines that protect you, bring you back now and then the way you should do it.

f. Could you provide me any documents which either come from or are produced for the management for the controlling purpose?	All of them: Not really, we haven't got much to show you. The moment things are done really up to us. But we can give you the draft RAD methodology which might help.

4.1.4 Other Information

Here are some more points made by them when I asked them before end of the meeting.

About RADs model

When I ask them (all the three developers in the meeting) to make comments about the RAD model, BC and JL thought it was fairly representative of their practice, but RK argued that the user group should not be there because he almost always has direct contact with end-users, meanwhile JL argued that in his case he always goes through the user group (or user management).

About the terminology

RK: It is dangerous to use the term of prototyping, because it often causes confusion.

JL: Yes, we tend not to use the term prototyping here, because most of us think that prototypes have to be throwaway, while we don't.

About customer 're-education'

RK: Another thing I want to mention is that we need to re-educate the customer to fit in to RAD thinking as well. I often encounter customers saying that you have to fill this or that form etc. when I try to sell a concept to them.

BC: It is easier to work afresh with somebody like the customer we [RK and BC] have. Because he hasn't had involve any IT project before, he don't have the pre-conceptions toward what is supposed to be involved, and that is the way we prefer to work.

About team makeup

RK and BC: We work on same project because our particular experience for the job, and also we have much freedom to choose our partners.

JL: Often the customer is not happy to see you all the time thinking that is waste of time. They would say: I will see the system in such a date, and they are not expect to see you in between.

About the bureaucracy

RK: Within our company there are a lot bureaucracy, the infrastructure such as the field services, the engineering group, and the tele-communication group they are sometimes a stumbling block. For instance, the black box here "estimate project", sometimes involving in or relying on the data provided by the field service group. And because the crucial factor of RAD is time. So I mean the infrastructure can really be the pain in the neck. Another example is we are currently working on a RAD project but are held up by the database people because they haven't loaded the database yet. Which in turn affects another related project—the ripple effect.

JL: I had similar problem. Once I had project which required two mainframes speaking to each other with very high speed, the link to one direction was in place, but the link in other end was not installed; As this was part of large project to be installed at later time, even we tried to bring it forward and also we had the tools and skill to do the job but we was not allowed to, and we don't have that muscle to bring it forward.

**** When asked where did he think the problems lie?

JL: In my case it's mainly because the priority of my work was lower than the other larger project and we are both under control of the business centre manager.

RK: It is also matter of risk. Because our work is relatively small RAD type, and we often rely on the data provided by some large project which involving many individuals or group, and having strict time schedules, so we suffer from their reluctance of taking risk.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**
a. Information Management Standards: Section 12 — Rapid Application Development (RAD) (Unreviewed second draft)	Introduction When should RAD be used When should RAD not be used What are the main elements of RAD The benefits of RAD Where does a RAD differ Major considerations in conducting a RAD How to perform a RAD The use of fast-track The life cycle The principles of RAD
b. DSDM principles	The 13 RAD principles with brief comments

4.2 Follow-up Agreement

They all agreed that I can make further contacts when needed.

4.3 Comments on the meeting

4.3.1 About The RAD model

4.3.2 About the personality test results

Two of them (RK and BC) have identical type: ENTP; one (JL) have the type: ENTJ, which only has one different attribute with those two; and all the four testers have attributes of E and N.

4.3.3 About The Questions and Answers

a. About the conversation

RK's comments on the process model is very valuable. He speaks out his experience inside some the black box (field services, and databases people), and a related, more general problem of large organisation — inflexible infrastructure and bureaucracy.

RK and JL also brought about the problems of communication and bureaucracy. Their comments no doubt once again to bring about the issue of process diversity owing to the organisation infrastructure.

RK did not agree the use of the term prototyping (refer to 5.1.3) I think he means by prototyping is simply the throwaways. This is proof of how differently people understand it. And it support my suggestions, in my literature search report, of a more clearly uniformed definition is need in the community in order to have a better communication.

They also raised an important issue of customer re-education to fit in the new development paradigm.

It is interesting to note that they have very loose control. This is evident from both their comments and the RAD model the only explicit control is at the end of project (the sign off : refer to the Diagram — Appendix D).

4.3.4 About Other Information

4.3.5 About The Supporting Documents

4.4 Key Issues Raised from the meeting

a. How much significance the organisation infrastructure and culture have over the prototyping process across different organisations?

b. Are there any the underlying patterns in term of size and business nature of organisations?

c. Can some of the problems be solved or alleviated by increasing all levels of communications?

d. Is customer education or re-education a general problem? What is the importance of this toward a better process?

Interview Form B1

1 Meeting Details

Date	13/4/1994
Duration	60 minutes
Company Reference	B
Department	
Meeting Reference	B1

2 Interviewee Brief

Interviewee Reference	DR
Job Title	Project Manager
Process Role	Project managing, developing
Other Information	

3 Meeting Agenda and Methods

3.1 Purpose of the Meeting

a. Introducing my research

b. Discussing the co-operation

c. Get a draft RADs process model

3.2 Following-up Plan

a. Arrange a date for clarification

b. Explain the potential benefits

3.3 Data Gathering Methods

a. Interviewing

b. RADs Modelling

4 Summary

4.1 Information Gathered

4.1.1 <u>The RADs Model</u>

Having explained the RADs notation, I tried to draw the RADs model by going through their normal practice — from the beginning to finish of a typical prototyping project. The results was a draft (hand-drawing) diagram which indicating the main roles, activities and the interactions. This was further tidied up using a drawing package and referred as the first version RADs. 4.1.3 shows the key questions asked and the answers.

4.1.2 <u>The Personality Test Results</u>

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied

The following questions are asked in order to model the prototyping practice in term of role, activity and interaction of the process.

Questions	**Answers**
a. What are the distinct roles involved in a typical prototyping project?	There are five roles. They are: business control board, managing (project manager, project leader), developer, customer and end-user. The customer and end users are internal to the company.
b. What is the function of the business control? and what is the difference between the project manager and the project leader?	The business control board is made up temporally for the project. It includes my boss, me (the project manager, the project leader and a couple of other people from other departments. The main function is to make important decisions on the system boundary, its business impacts and resource allocations. The project manager has the overall control of the project. The main responsibility is to co-ordinating the project team and the customer/end-users, and allocating adequate resources. Whereas the project leader is responsible for the technical side of the project, e.g. in charge of the database design, implementation, maintenance etc.
c. How is a project initiated?	Normally a project is initiated either by us or by the customer, and they will do some sort of feasibility study to see if the project goes ahead.
d. How initial requirements are gathered?	We —me and the DBA person— normally gather the requirements by having two or three meetings with the customer and end-users.
e. What you give to the prototyper to start to build the prototype(s)?	A set of functional specifications or high level designs.
f. What the prototypers do on their own? who they contact to?	The prototypers do the coding and testing according the functional specifications. They'll have direct contact with us if any problems occurs during the prototypes building. They involve in demonstrating the prototypes to the customers and users.

g. What and who is involved in the prototype(s) evaluated?	The project managers, the prototypers, customer and end-users.
h. Do you sign off the prototypes with customer as (or as part of) specifications?	No, we intended but not actually happened.
i. Any change request come to you either from the prototyper or user group or end users?	Yes, the change requests mainly come from the end-users to either the design manager or the prototyper or both.
j. What and who involved in the decision making?	The project manager and the design manager, and some time involve the business control board.
k. Who involves in the final sign off?	The business control board and the customer.
l. Could you indicate any other explicit controls on the diagram?	It looks OK, I can't see now if something is missed from the draft.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

4.3 Comments on the meeting

4.3.1 About The RAD model

4.3.2 About The Personality Test

4.3.3 About The Questions and Answers

4.3.4 About Other Information

4.3.5 About The Supporting Documents

4.4 Key Issues Raised from the meeting

Interview Form B2

1 Meeting Details

Date	13/4/94
Duration	60 minutes
Company Reference	B
Department	
Meeting Reference	B2

2 Interviewee Brief

Interviewee Reference	DR
Job Title	Project Manager
Process Role	Project managing, developing
Other Information	In charge of production support both in application and technical aspect. Managing about 30 staff.
Interviewee Reference	ML
Job Title	Project Leader
Process Role	Project managing, developing
Other Information	Responsibility includes specifying project team member requirements, data modelling and some other technical aspect of a project.

3 Meeting Agenda

3.1 Purpose of the Meeting

a. To clarify and refine the RADs model.

3.2 Following-up Plan

a. Arrange a meeting to ask more questions

3.3 Data Gathering Methods

a. Making comments on the diagram

4 Summary

The first version of RADs model is clarified and refined in this meeting with DR and ML.

4.1 Information Gathered

4.1.1 The RAD Model

The following are the changes need to be made on the first version of the RADs model:

a. Separate the single role — Client/user — into two distinct roles as Client and End-user;

b. DBA role is not take part in the requirement gathering stage;

c. Some of the activities either inadequate or misplaced;

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied

Questions	**Answers**

4.1.4 Other Information

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

4.3 Comments on the meeting

4.3.1 About The RAD model

4.3.2 About The Personality Test

4.3.3 About The Questions and Answers

4.3.4 About Other Information

4.3.5 About The Supporting Documents

4.4 Key Issues Raised from the meeting

Interview Form B3

1 Meeting Details

Date	2/2/95
Duration	90 minutes
Company Reference	B
Department	European System Development (120 staff)
Meeting Reference	B3

2 Interviewee Brief

Interviewee Reference	DR
Job Title	Systems Manager
Process Role	Project managing
Other Information	In charge of production support both in application and technical aspect. Managing about 30 staff.
Interviewee Reference	ML
Job Title	Project Leader
Process Role	Project Managing, prototyping
Other Information	Responsibility includes specifying project team member requirements, data modelling and some other technical aspects of a project.

3 Meeting Agenda

3.1 Purpose of the Meeting

a. To ask the questions based on the further investigation guidelines [see appendix E(c)]

b. To do the personality test

c. To get some relevant documents

d. To refine the RAD process model

3.2 Following-up Plan

This was the last visit intended.

3.3 Data Gathering Methods

a. Interview with tape recording

b. Run the Personality Test

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	Personality type

DR	INTJ
ML	ENFJ
4.1.3 The Key Questions asked and Answers replied	
Questions	**Answers**
a. You may remember that I asked you for some documents. I wonder if you have them with you to show me? And just before your colleague join us, please tell me a bit more about the RAD project you mentioned before?	DR: You asked me last time about the relevant documents, the trouble is that they're all archived, so take a while to get it back. What I've got here is the life cycle which includes how our company go about the RAD. Because we recognised that the way we worked previously and various documents we produced did not fit in the rapid development as the project we did before.
	Another a couple of documents I brought through are the way we operated the main part of the project, where ML, he will join us in a while, performed the analysis with the users to determine their requirements for specific applications.
	Normally, the next stage for him will be to write the applications, show users and go through the prototyping until everything is finished, you might remember that we have a lot of junior people in our team, and the idea was they need to be introduced into the new
	languages and new techniques. But, because they don't know the business they won't be able to go through the full prototyping cycle. So, out of the initial discussion, ML produced the high level views [refer to the RAD model: work-break-downs] which people code from.

	So, it was like adding an extra phase because ML was not going to do the coding , he is only there to find out the requirements and document it. The programmers then coded from that, and demonstrating that to the users. So we really start doing prototyping at this level. As I mentioned earlier, because they lack of experience and business knowledge they was not able to do it all the through.
b. Do you think this project is successful?	DR: Yes, I think so. It is recognised, and we have documents to prove it, that the productivity for this project—we did function point analysis—was 3 to 1 comparing with other projects we developed here using 4GLs. And this was done with a team that most people who were new to the data processing and were novice programmers (less than two years experience), just ML has got a lot of experiences and I was project leader. The customer was happy as well because it provided more functionality than they had before and even better than similar systems other banks have.
c. Do you have some carrying on work to do since the system was up running?	DR: No, unfortunately, the decision had been made not to continue to develop it. This is because the product did not fit in the company B infrastructure, we primarily are Natural [the language used] and Adverse [the file the data kept in company B] shop, if you like, and the tool we used for the RAD—Huron which is promoted by Amdahl—doesn't directly interface with the Adverse files. It works fine now as stand-alone system, which was originally sought out.

d. So why was this interface issue not considered beforehand?	DR: The idea was to speed up the deliveries to the business users. In the banking world there is a great demand for new products, customer always ask something different, we have to see what other banks are offering, and keep the fore for it. Huron is good at RAD, when Amdahl made the presentation to top management, they decided to have a go with it, so the Agency system was the trial. The system used to run on Wang machine and written in COBOL, and company B want to get rid of those Wang machines and aim to re-write the whole system.
	Because the Agency system is self-contained, they decide to use Huron to build from scratch. We didn't look into too much about the existing system, we talked to users about the functionality they wanted, and documented all that, then identified a pilot system to run over two month to deliver a subset of the whole application. After we evaluated the pilot system, we developed the rest of the system in seven months time with 70 applications. In the pilot we did six applications in two months.
	So, we used the tool, the data modelling and the prototyping, we didn't have all the team involved in the whole prototyping process because they are not experienced enough. We have the experienced ones dealing with users and less experienced doing the prototype building. So we adapt our methodology all the way through.
e. Do you have the method which you just gave me before you started the project?	DR: Not before we started. We started the project at July 1992 and finished at August 1993, and this Huron methods was not available until May 1993. The method was
	developed based on our project, as I said before, it was that the existing life cycle methods within company B is inappropriate for the RAD project.

f. How many applications were identified before the prototype building start?	DR: We started with a meeting with about 20 people —some were from the user department, some were the system development team and some were the management from different business area— and identified about 30 main applications (I have to say I thought we got almost all of them because the initial meeting went over 3 days, and we went through all the business processes and documented all the requirements.)
g. How did you decide which to develop first?	DR: We then discussed with the users about building a pilot system which would form the basis of the final system, and determined six core applications for the pilot system.
h. How many applications were built at the end?	DR: After the pilot we back to users and put on about another 20 applications, and finished with the third round at 70 applications.
i. How about the database design?	DR: Having gathered the initial requirements and all the data that they required, first thing we did was to build our data model, we had help from Amdahl consultant and came up with the data model for the Agency system. We talked the data model through with our users, so that they were happy with the data relationships.
j. What is the difference between the way of doing a non-prototyping project and the RAD project?	ML [he came in the half way through]: With the conventional system development, typically we would, first of all, identify all the requirements and document them, and confirm them with the users. Then we build the system

	as conventionally. But we do have the users come in during the development to clarify and validate the requirements. The difference is, with the RAD project, we were easily led to the users way: build what they want. But by the end of it, when we recognised some problems, e.g. from the security or audit point of view, say if the system crashes half way through, or if somebody can do something with it, and you ask them what happened if... they'll turn around say: that is you problem! So we always end up adding lots more on. Whereas traditionally we'll take all this into consideration into design before the implementation. So, at least we know what we are doing.
k. So you really ought to have some of these design considerations to start with?	DR and ML: Yes you should bear that in mind all the way through, but it is very difficult to do because, in a way, we are really designing the system with the users.
l. Do you have the standards and methodology used for the conventional development?	DR and ML: Company B has its own methodology, which is phased-oriented.
m. Do you use them? How?	DR and ML: We do refer to it sometimes, especially at the beginning of a project, and at least we use part of it for control purposes. The SDS —system design specification—is such a document that needs a user's sign off. For a large project, we normally do a SDP(system definition plan) first, then the SDS and the SDIP (system design implementation plan). For small project we might just need one of the three document standards.
n. What is the size of the RAD project we have been talking about?	DR and ML: We would say it's probably a medium size. 5 people for about 9 months.

o. So, on average, how many iterations did you go through?	DR: I would say about 3 or 4 times, but for some applications it went up to double figures. What happened was that the different user wanted different thing, and they often conflict, so we end up with no real progress.
p. How do you perform the work breakdowns?	ML: You just go out there talk to user what they really want and you just do it. I mean you have a main function for each application, and you may have anything from one screen up to 20 screens — for inputs and outputs— for each application.
q. What is the criteria?	DR: As ML said, we first talk to users to find out what sort of thing they want to do. Then it is down to us to decide if the requirements fit in one application, if not we then split it down. It is really down to the experience.
r. How did you set up such a project team? What was the criteria?	DR: It is really my boss's decision. He chose me to take charge of the project, ML as a experienced software developer and another team member [software developer] with one year experience, and also an additional four inexperienced programmers involved in the later building the pilot system. The reason for such a team is to see how we — different ranges skills and experience— react to the new development environment and new approach. If it's all very experienced people involved, we could not learn much from it. But, I do think it was luck we have ML in our team, because he got a good knowledge and experience in both technical and business area, and it was, I think, very important to the success of the project.
s. Could you provide some more documents such as the initial requirement specification, meeting minutes and the sign-offs?	DR: We had some this kind of documents, it is somewhere, but I am not sure that I would be able to find it. We started to write up the specification when we got about 50 applications, and the specification had been constantly changing, probably about once a week.

	ML: The project started at June 1992, and we didn't get the specifications signed off until August 1993. So you can see the requirements were not agreed at the beginning but at the end of it. In a way, we don't know what we were doing in between.
t. Did you do the analysis and design for the project?	ML: Well, there is no analysis because we were prototyping, it's just all designing. we did the data modelling, and produced some kind of high level design, that's it.
	DR: one thing we did not feel comfort about was that we were not very clear, at outset, what we were doing and where it leads.
u. Did you have some quality check over the project?	DR and ML: No, we didn't have any.
v. How about the CRs control?	DR: Eventually we had a committee consisting of the systems manager, the user manager and the auditor, and they would decide on changes. Basically we got a large number of changes all the time, and if we kept going like that the delivery date would keep backing. The committee met every week.
	ML: If we think the changes were small we would just change it, otherwise we would let the committee to decide. But if something was urgent so that we can't wait until next meeting, we'll still do it. DR: We tented to resist the changes that involve lots of effort. One classic example was
	that if two applications trying to update one form at same time, who ever hit the key first would succeed, and the other one would fall over. We suggested to them not update at same time, but the fault is still there. At the moment we can't afford change it because it related to the basis that we built the system.

w. What about the responsibilities to changes made?	ML: Apart from the committee, me and DR would have the responsibilities, the prototypers would report whatever the changes to us.
	DR: Ideally, I think the prototyper may take more responsibilities with some guidelines.
x. What did you use for customer end?	DR: We had two sign-offs. One was for the specifications; the other was for the user acceptance before the implementation (go live to production).
	ML: The initial requirement sign off that appears on the RAD model is not actually happening, that's what should have happened.
y. How about the cost?	ML: Every department has their own budget. So every project we do has to be funded. We work just as if they were external.
4.1.4 Other Information	
ML: One thing I want to say about prototyping is that, because of our approach, the users tend to not think hard enough to try to get all the requirements at first place. So by signing off the document like SDS, they know that is what they are going to get, and therefore they will take it more seriously. And we will more clear what we're going to do.	
DR: Another thing I would like to say is that, after we implemented the project, if we had spent more time with users in the beginning to try to identify all the applications before starting the building, say if we identified the 70 applications at the beginning, you can then break down more structurally. For example, you can say these 10 are related and focusing on that until users satisfied with it, and so on so forth. What happened was that, after the pilot system, we tend to jump from one application to another, and therefore less efficiency.	
Questions from the researcher: But, is it worthwhile to spend all that time on specifying rather than prototyping?	
DR: Well, I think we at least can spend one more week on specifying, in doing so we might get another 20 more applications specified, so we can have more reasonable work-breakdowns to start with the prototyping. This will avoid too many iterations. We want set our limit to three iterations.	
4.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**

a. Huron System Development/Enhancement Life Cycle (section 1)	This documents contains the following: Section 1.1 Definition Phase Section 1.2 Analysis Phase Section 1.3 Prototyping Phase Section 1.4 System Control Requirements Section 1.5 Implementation Phase Section 1.6 Acceptance Into Production Status
b. A sample document of the functional specifications of a RAD project	This contains the functional specifications of 17 sub-applications
c. Additional documents	c1. Huron Naming Standards c2. Frankfurt Integration Project (a conventional development)

4.2 Follow-up Agreement

DR suggested that I could contact one of his colleague who is going to take charge of the RAD development.

4.3 Comments on the meeting

4.3.1 About The RAD model

There are a few places need to change:

a. The DBA role should be Data Modelling.

b. The first sign-off was not actually happening until the project finished—almost same time with the final acceptance sign off—which is not clearly shown on the diagram.

c. The activity document requirements were not actually happening.

It is interesting to note that these points were neither recognised during the modelling, nor reported from the feedback comments, but from the conversation summarised above.

On the one hand, obviously, it is because that the more you know the process the closer you match the model to the process. On the other hand, partly it is because of somewhat their sometime confusion between what things are supposed to be and what it is; partly it is, I think, the "tried too hard" effect —too much wanted something to get it. In the case of my field modelling, the emphasis was always 'the model'. Saying and implying: I came here for 'the model', can you help me get 'the model', can you comment on 'the model' etc. in stead of making them more aware that how things get done and what is going on here.

In contrast, because the emphasis of this meeting was on how and why things get done, the things need to be modified on the diagram come out naturally.
This suggests that the field modelling can be done more effectively by having a good conversation with the practitioners before attempting to model the process using any of the modelling notations.
4.3.2 About The Personality Test From 4.1.2 [DR: INTJ; ML: ENFJ] we can see that Both ML and DR have two characters in common: N (intuition) and J (Judging). Has this anything to do with the fact that they are both managers? Comparing this results with the other companies might indicate something more general.
4.3.3 About The Questions and Answers
4.3.4 About Other Information
4.3.5 About The Supporting Documents
4.4 Key Issues Raised from the meeting

Interview Form Ca1

1 Meeting Details	
Date	22/6/94
Duration	60 minutes
Company Reference	C
Department Reference	a
Meeting Reference	Ca1

2 Interviewee Brief	
Interviewee Reference	PM
Job Title	Design Manager
Process Role	Managing, System Designing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Introduce my research

b. Get a draft RADs model

3.2 Following-up Plan

a. make a date for further clarification and refinement of the RADs model.

3.3 Data Gathering Methods

a. Interviewing with taking notes

b. Diagramming using RADs

4 Summary

4.1 Information Gathered

The information gathered is arranged in following:

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	Personality type

4.1.3 The Key Questions asked and Answers replied

The following questions are asked in order to model the prototyping practice in term of role, activity and interaction of the process.

Questions	Answers
a. What are the distinct roles involved in a typical prototyping project?	Project managing, System Design, developer/prototyper, system proving, component engineering, internal marketing group and external customer.

b. What are the main functions of each of the roles?	The project managing, designing, marketing and customer are just as the name suggested having their conventional interpretations. The prototyping role is actually part of the system design and performed by the designers, it is separated out because we want to emphasise the prototyping effort within the design group. The component engineering is equivalent to implementing.
c. How is a project initiated?	A project is initiated by the product marketing group.
d. How initial requirements are gathered?	The initial requirements are normally presented in the product description from the marketing group.
e. What you give to the prototyper to start to build the prototype(s)?	As mentioned earlier, prototyping is part of designers' job, it normally takes on the new concepts/ideas for experimenting.
f. What the prototypers do on their own? who they contact to?	The prototypers do the design, coding and testing according the new ideas and/or initial requirements. They both contact me for any problems occurs during the prototypes building, and the implementing group for technical viability.
g. What and who is involved in the prototype(s) evaluated?	Me (design manager), project manager, the marketing group.
h. Do you sign off the prototypes with customer as (or as part of) specifications?	No.
i. Any change request come to you?	Yes, the change requests mainly come from the prototyper and the component engineering group.
j. What and who involved in the decision making?	Me, the project manager and the marketing group.
k. Who involves in the final sign off?	The marketing group, the project manager and the customer.

1. Could you indicate any other explicit controls on the diagram?	No.
4.1.4 Other Information	
4.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**
4.2 Follow-up Agreement	
4.3 Comments on the meeting	
4.3.1 About The RAD model	
4.3.2 About The Personality Test	
4.3.3 About The Questions and Answers	
4.3.4 About Other Information	
4.3.5 About The Supporting Documents	
4.4 Key Issues Raised from the meeting	

Interview Form Ca2[2]

1 Meeting Details

Date	6/7/94
Duration	45 minutes
Company Reference	C
Department Reference	a
Meeting Reference	Ca2

2 Interviewee Brief

Interviewee Reference	PM
Job Title	Design Manager
Process Role	Managing, System Designing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. To clarify and refine the RADs model.

3.2 Following-up Plan

b. Arrange an interview with PM (and his colleagues if possible) aiming to asked more specific questions (see appendix E(c): the questions for further investigation) based on the RADs model.

3.3 Data Gathering Methods

a. Make comments and corrections on the diagram sheet.

4 Summary

4.1 Information Gathered

The information gathered is arranged in following:

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied

Questions	**Answers**

4.1.4 Other Information

4.1.5 Supporting Documents from the Interviewee and their Contents

---

[2] As for the sole purpose of this visit was to refine the RAD model and only minor changes were made, so most of part of the form are left blank. It is presented here just for completion.

Documents	Contents
4.2 Follow-up Agreement	
4.3 Comments on the meeting	
4.3.1 About The RAD model	
4.3.2 About The Personality Test	
4.3.3 About The Questions and Answers	
4.3.4 About Other Information	
4.3.5 About The Supporting Documents	
4.4 Key Issues Raised from the meeting	

Interview Form Ca3

1 Meeting Details

Date	21/10/94
Duration	90 minutes
Company Reference	C
Department Reference	a
Meeting Reference	Ca3

2 Interviewee Brief

Interviewee Reference	PM
Job Title	Design Manger
Process Role	Managing, System Designing
Other Information	
Interviewee Reference	AB
Job Title	Development manager
Process Role	Managing, component engineering
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. To get more feedback about the RADs model;

b. To find out more detailed information about the process based on the RADs model.

3.2 Following-up Plan

a. To conduct the Personality Test

b. To contact ST [development manager] and GB [process manager] for some supporting documents.

3.3 Data Gathering Methods

a. Record the comments and make notes about the RADs model;

b. Record the interview based on the question guidelines on the further investigation [see Appendix E(c)]

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

PM: this RADs model is centred around design. Actually, our use of prototyping is more than just in design, it has been used in requirement capture, cost estimation, even to get the initial product descriptions.

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied	
Questions	**Answers**
a. How your group practice prototyping?	AB: We have a dozen of developers in our group. We normally get the requirement specification or new ideas from the design team, we then have interactive coding to explore either the feasibility of the design or to optimise the design implementation.
	PM: In our design team, we use prototyping to uncover and experiment new ideas. Because most of our systems are built concurrently, so does the prototyping. Our practice now tends to go like this: you have whole bag of requirements, and you choose or determine the framework of the system required, quick design it and pass to the component engineering to build it. At same time, some other part is designed and implemented in a similar fashion.
b. Do you have separate people as prototyper(s) in your group?	PM: Not normally, as I said earlier, prototyping is just a technique we used here, most designers use it. And the results of the prototyping will normally goes into the requirement specifications without much trace. AB: Yes, I agree what PM said and we do it very much the same way. As I said earlier, we'll explore the design ideas further or adapt it, and do our own prototyping to make the code more modular and more efficient.

c. Do you have any guidelines or procedures made for prototyping?	PM: I am not aware if there is any in our group. AB: Well, because the prototyping is often very short and sharp sessions, so it is very much down to individuals and not fully documented. But they do have to abide by the company's standards such as coding standards.
d. Have the developers have the right to do any changes without control?	AB: No, they can't. First of all, they need to do it according the specification, and if there are problems in implementing they has to report the team leader and therefore making decisions accordingly. Of course, each individual have the right to decide the way they want go about it.
e. How you two groups co-operate?	AB: We work very closely indeed, sometimes the boundary is rather blurred. If we find out something problems in the design or the requirement specifications, we'll discuss them with the design people.
f. How about the change request control?	PM: Changes and problems that are identified by component engineering group is feedback to system design group by several reviews, the system design review documents are reviewed many time during the project life cycle. We will then do some changes accordingly, or if there some fundamental changes on the requirements we have to go back to negotiate with the customer. So we do have overall control standards and procedures for life cycle, but not for prototyping.
g. Could you provide me some control documents used in your projects?	PM and AB: We are not sure if we are able to do so, we'll let you know later.

h. You mentioned you use prototyping elsewhere during the system development, not just as showed in the diagram, could tell me a bit more about it?	PM: For example, we used for confirming initial requirements. But just used internally within the system design people, as I said before as a technique. No customer or user validation. Also how we use it varies from project to project. There was a case we did some prototyping with the marking people and the customer, we got a mock-up very quick and ask if that's what they want, while in another case we were not sure about the requirement, so we did some prototyping and found out it not feasible. In this case we just told the customer that their request did not make sense or not possible instead of demonstrating the prototype.
i. Is it true that you normally have fairly well-understand requirement and therefore straight go to design?	PM: That's not true, there normally a loop there, probably go through two or three times. A9AB: That was true probably 10 or 15 years ago with relatively small project, but not with nowadays large project with large companies or organisations.
j. How about the initial requirements gathering? how many interactions? how much effort?	PM: The methods for this stage could be anything from interviewing, questionnaire to prototyping and simulation. The requirement gathering stage may run through the project or even longer. and the iterations may from 1 to n. I've seen the list questions about the statistics on that paper [see the attached sheet], but I'm not the right person to ask, you might contact the development manager see if she can help.
k. Is the system design a rather distinct stage?	PM: Yes, it is in our case, it would be rather blurred if we were doing small project.

l. Do use or company wide standard and procedures in your work?	PM: Yes we do. Because our projects normally are very large, which may involve many companies across countries and produce millions lines of codes, we need follow rigorous methods and standards. Although it is difficult to follow, it is, in my opinion, essential to make the process manageable. AB: But we not followed it in a sequential manner. Because of the time scales and concurrent processes we also can't afford to follow it rigidly.
m. Any procedures or guidelines on CRs controls?	AB: Yes, we do have such control procedures in place. PM: Yes, we may have some, obviously we don't slavishly follow. Our control is basically done by the reviews. I think it is difficult and unnecessary to classify the changes, it should be down to the designer or the component engineer to decide what should be done internally or externally because the situations varies from individual to individual and from project to project.
n. Who make the change decisions?	PM: Some changes settled internal to the team [design or component engineering] as the problem occurs, some requests from the component engineering are solved either by the design team review or individual designer or both.
o. Do you think a classification of the common accrued problems or requests would help to control?	PM: Well, I think it's difficult to do that, and you have to trust the manager and/or the developer to make their own judgement, they have the ability to look over their own desk. AB: Sometimes things do go wrong and problems not discovered until proving stage, hopefully not when customer using it.

p. Have you had such experience: some problems occurred because the component engineer did some changes without consulting the design team?	PM: Yes, sure.
q. You mentioned you don't follow the procedures slavishly, why?	PM: Because they are too rigorous, too detailed to follow.
r. How quality standards and control?	PM and AB: We have many company wide standards, we also comply with ISO 9000. We have a group who guide and enforce these standards. You may try to talk to the process control people about these.
s. Who you choose to do the requirement gathering?	PM: The people are usually chosen from the design team who have expertise in requirement eliciting and the domain knowledge.

4.1.4 Other Information

PM:

a. It looks to me that you treat prototyping as a method or methodology, but, in my an opinion, it is a technique that can be used in various areas in software development.

b. Prototyping [GUI builder knock up some thing quickly, i.e. a mock-up] often used at requirement stage.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

They agree that I may contact them when needed.

4.3 Comments on the meeting

4.3.1 About The RAD model

4.3.2 About The Personality Test

4.3.3 About The Questions and Answers

4.3.4 About Other Information

From what PM and AB said there is no prototyping team as such, so most of the questions of "prototyping initiation" of the question guideline is seen not relevant and not asked.

4.3.5 About The Supporting Documents

When asked about providing some supporting documents, they said they cannot give any to me for (1) all the company standards and procedures, and all the project documents are managed and kept by a different group; (2) They are not authorised to distribute any of these documents.

4.4 Key Issues Raised from the meeting

a. PM repeatedly emphasised that the prototyping is used as a technique within their conventional development life cycle rather than a development approach on its own right. Is it therefore, as he suggested, not much need for any formal control or guidelines in their practice?

b. One impression through the conversation was they have lots standards and methods that has been used more of conforming some standards than of providing guidance and/or controlling the quality. Often, when asked about the standards and procedures, they say: I am not sure about this you better ask somebody else (e.g. the ISO 9000 guy etc.). So I wonder how much real benefits they can get from having those procedures and standards when not many people who actually doing the development work use, or even know about it?

Interview Form Cb1

1 Meeting Details

Date	6/6/94
Duration	60 minutes
Company Reference	C
Department	b
Meeting Reference	Cb1

2 Interviewee Brief

Interviewee Reference	MF
Job Title	System Design Manager
Process Role	Design Managing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

To get a draft RADs Model

3.2 Following-up Plan

a. To clarify and refine the draft RADs model

b. To obtain some more detailed knowledge about the process based on the RADs model

3.3 Data Gathering Methods

a. Interviewing

b. RAD diagramming

4 Summary

4.1 Information Gathered

The information gathered is arranged in following:

4.1.1 The RAD Model

This first meeting with MF aims to get a draft RADs model of the prototyping practice. Five distinct roles and their interactions along with some key activities were identified.

4.1.2 The Personality Test Results

Reference	Personality type

4.1.3 The Key Questions asked and Answers replied

The following questions are asked in order to model the prototyping practice in term of role, activity and interaction of the process.

Questions	Answers

i. Any change request come to you?	Yes, the change requests mainly come from the prototyper.
j. What and who involved in the decision making?	Me, the commercial group, customer and some time the prototyper(s).
k. Who involves in the final sign off?	The commercial group and the customer.
l. Could you indicate any other explicit controls on the diagram?	No.

4.1.4 Other Information

The department where the interviewee works has about 10 people in the development team and 15 in the proving team. The main work here is to install a certain IN (Intelligent Networking) platform and develop applications within the environment on customer's demand.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

a. I will come back to get some feedback of the first version that derived from the draft RADs and the conversation.

b. I will ask a few questions based on the RADs model.

4.3 Comments on the meeting

4.3.1 About The RAD model

It is surprising that there is no users and/or customer involvement in the whole prototyping building stage.

4.3.2 About The Personality Test

4.3.3 About The Questions and Answers

4.3.4 About Other Information

4.3.5 About The Supporting Documents

4.4 Key Issues Raised from the meeting

a. Why there are little users and/or customer involvement during the process?

Interview Form Cb2

1 Meeting Details

Date	22/6/94
Duration	45 minutes
Company Reference	C
Meeting Reference	Cb2
Department	b

2 Interviewee Brief

Interviewee Reference	MF
Job Title	System Design manager
Process Role	Managing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. To clarify and refine the first version of the RADs model

b. To ask some questions based on the RADs model

3.2 Following-up Plan

None

3.3 Data Gathering Methods

a. Interviewing

b. Making corrections to the diagram (the RADs model)

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

[refer to the first version which is revised from the draft based on this interview]

4.1.2 The Personality Test Results

Reference	Personality type

4.1.3 The Key Questions asked and Answers replied

Questions	Answers
a. From the draft RADs model we did last time, there is no end-user on the diagram. is this true?	MF: This normally is true. Because the direct end-user, in most of the cases, is a machine rather than a human user.

b. Why not directly involve the customer in the prototyping stage?	MF: Well, that is all to do with money. If we let the customer evaluate the prototypes they may bargain with us saying that the system does not seem that difficult to do, they could find somebody else to it much cheaper. This situation has happened before, especially a company like BT. So we try to keep the prototype out their sight, and just ask what they want, if we find out it's worthwhile doing we'll sign off the specification, and deliver the product specified at the end.
c. What is the use of prototyping in your practice?	MF: It has been used for risk analysis, cost estimation and design evaluation. If I can have a prototype easily, either from a previous similar project or build from scratch, I may used the prototype for cost estimation or risk analysis. But its main use here is for design evaluation — validating and proving the design.
d. What is the average size in terms of man-months of effort for the projects?	MF: It is difficulty to say, probably a normal project take about six month and three person, so about 18 man-month's effort. This does not include the technical implementation and acceptance testing.
e. How much is the average effort for the prototype building comparing with the whole project?	MF: As I said before, our department is mainly responsible for the design implementation and its proving, it normally involves a few different departments. So I cannot give you a definite answer. What I can say is that it roughly take 10 percent of the whole effort within our department boundary—the design and proving.

4.1.4 Other Information

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

4.3 Comments on the meeting
4.3.1 About The RAD model
4.3.2 About The Personality Test
4.3.3 About The Questions and Answers It seems that the prototype he meant is the first working version of the final system. [Reference to question e]
4.3.4 About Other Information
4.3.5 About The Supporting Documents
4.4 Key Issues Raised from the meeting
a. Are there any needs for customer/users involvement in their practice? **b.** If yes, how can we avoid or reduce those disadvantages such as customer's and/or user's inadequate expectation, mistaken effort needed, and therefore unfair (not always) bargaining?

Interview Form D1

1 Meeting Details

Date	3/21/94
Duration	60 minutes
Company Reference	D
Department	
Meeting Reference	D1

2 Interviewee Brief

Interviewee Reference	RM
Job Title	Technical Director
Process Role	Project Managing, developing
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Introduce my research

b. Get a draft RADs model of their prototyping practice

3.2 Following-up Plan

a. Refine the RADs model

b. Ask RM to invite least one of his colleague who is closely involve in their prototyping practice.

3.3 Data Gathering Methods

a. Interview with taking notes

b. RADs diagramming

4 Summary

4.1 Information Gathered

The information gathered is arranged in following:

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	**Personality type**

4.1.3 The Key Questions asked and Answers replied

The following questions are asked in order to model the prototyping practice in term of role, activity and interaction of the process.

Questions	**Answers**

a. What are the distinct roles involved in a typical prototyping project?	Managing, developing, customer and end-user.
b. What are the main functions of each of the roles?	Managing: Dealing with customer, i.e. initiating project, gathering requirement, change requests, and signing off. Developing: System development including coding and testing. Customer : initiate project; evaluate prototypes and sign offs; End-user: requirements gathering; prototypes and system evaluation;
c. How is a project initiated?	Normally the initial request comes to me via telephone or Email by customer.
d. How initial requirements are gathered?	I normally have a couple of time meeting and at same time do some prototyping [screen mock ups] with the customer and/or end-user.
e. What you give to the prototyper to start to build the prototype(s)?	In most my projects, I'm the prototyper myself, and I often have a functional requirement to start with.
f. What the prototypers do on their own? who they contact to?	Coding and testing the prototypes. They'll normally have direct contact with customer and/or end-user for gathering and clarifying requirements, and with their manager mostly for technical problems.
g. What and who is involved in the prototype(s) evaluated?	Normally the customer and user(s), manager and prototyper(s).
h. Do you sign off the prototypes with customer as (or as part of) specifications?	Yes, I do.
i. Any change request come to you either from the prototyper or user group or end users?	Mostly from customer.

j. What and who involved in the decision making?	The manager and customer, and sometime the sales manager;
k. Who involves in the final sign off?	The manager and customer, and sometime the sales manager;
l. Could you indicate any other explicit controls on the diagram?	No.
4.1.4 Other Information	
4.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**
4.2 Follow-up Agreement RM agreed to arrange an further interview	
4.3 Comments on the meeting The meeting went as planed, and as a result a draft RADs Model was drawn. Although being made it clear that the RADs meant to represent their general practice, the impression was that the daft model is still more of his own than their general practice. So it is important to get other prototypers view about the model.	
4.3.1 About The RAD model	
4.3.2 About The Personality Test	
4.3.3 About The Questions and Answers	
4.3.4 About Other Information	
4.3.5 About The Supporting Documents	
4.4 Key Issues Raised from the meeting	

Interview Form D2

1 Meeting Details

Date	18/11/94
Duration	90 minutes
Company Reference	D
Department	
Meeting Reference	D2

2 Interviewee Brief

Interviewee Reference	RM
Job Title	Technical Director
Process Role	Managing, prototyping
Other Information	Many years software development experience with the company. His main responsibility includes giving technical advice to both the company's managing director and system developers.
Interviewee Reference	SS
Job Title	Sales Manager
Process Role	Managing
Other Information	12 years project managing experience before joining this company. His main work involves selling products, doing initial proposal to clients and writing up specifications.

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Modify and refine the first version RADs model

b. Ask more specific questions based on the RAD model and the question template for further investigation

3.2 Following-up Plan

a. Carry out Personality Test

b. Get some supporting standards/procedures and project documents

c. Ask a few questions of the developers while doing the personality test

3.3 Data Gathering Methods

a. Tape Recording

b. RADs Modelling

c. Question template for further investigation (see appendix E(a))

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

According to the discussion with RM and SS, a few changes were made on the first version RADs which was derived from the draft version and notes from previous interview. The changes were:

(1) end-users were involved earlier at requirements gathering stage;
(2) there was a sign off before prototype building;
(3) there was a sign off of the requirement specification at the end of building prototypes;
(4) activity "write functional specifications" and "refine functional specifications are rarely happened in reality, so it may more true to not show them on the RADs, which mean to represent the actual process.
(5) the sale manger is also a significant role in co-ordinating customer and prototypers, and decision-making in the process.

4.1.2 The Personality Test Results

Reference	Personality type

4.1.3 The Key Questions asked and Answers replied

Questions	Answers
a. Could you provide me some procedures or guidelines or any standards that are used for control purpose?	RM: That is tricky because we are just started to try to put those sort of standards into places, but I can show you are some of the documents that we sent to clients saying that's what we going to do with the projects. We are going to prepare for the TickIT over next six month starts from January 1995... So we may show you something then. The moment we are really busy, and we haven't really thought about what exactly we are going to put in those standards and procedures.
b. Do you use the 'time box' method to control?	RM and SS: No we don't! Because if we can't deliver on the fixed date they may send us to court. We do some time have such request from clients, but often the deadline given was not realistic. In another word, we don't want to pay the price of their bad planning.

c. How is a prototyping project team made up? Especially how and why prototypers are chosen?	SS: One project is normally undertaken by one developer, and often there is another person who take some managing responsibility. This mainly due to the financial constraints, and also because the very interactive nature of most project we find it is easier to manage this way.
	RM: In some cases one person may involve getting a draft specification, and someone else then may go on to build the prototypes with the users. We always have some sort of specifications even very simple verbal description, and we are trying do it more formally and document it.
	RM and SS: As to the prototyper selection criteria, we don't normally have the luxury to have many people to chose from, we use just the next available, of course we do consider the developer's expertise in using particular development environment and their experience, for example, if the project is an application using Lotus Notes, we will chose someone have more experience in using it than others do.
d. Are there any differences between conventional and prototyping projects in term of selection criteria?	RM: No, not much differences. SS: If we have more than one person to chose from, and given they all have similar experience, we will consider their ability to work with customer, i.e. the communication skill.
e. What, if any, guidelines or principles are made for managers and prototypers to carry out each project?	RM and SS: We have a project proposal which contains the project objectives and responsibilities for each party. So the proposal in fact implies the prototypers' responsibility, although we don't have this sort of guidelines explicitly.

f. How are the initial requirements gathered? How many iterations with customer?	RM: This has been done by using normal requirement gathering methods such as interviewing customer, viewing relevant customer documents. The result of this stage is the project proposal, which, in most case, gets refined once before signing off, whatever no more than twice.
	SS: We some times also demonstrate our past similar projects to customers. This has often led them to say "oh, yes I like that...". So we do some time used ready prototypes for requirements gathering, but probably it is used more as part of selling.
g. How much effort the prototypes building compare with the whole project effort? are these figures stable across projects?	SS and RM: We would say the prototyping effort is about 10 percent of the whole project. Yes, they are relatively stable. ... the prototyping effort here means the prototypes building before signing off the requirements specifications.
h. How prototypes are built? how many iterations in term of user validation or demonstration?	RM: Normally the developer will build the prototype with the customer if they just want see what the screen look like, or if they want a pilot system with some functionality some coding has to be carried out separately. A couple of times iterations on average.
i. Do you do any analysis and design at this stage?	RM and SS: No, we don't normally have the need for that because the applications tend to be rather small and mostly just a mock-up rather than a pilot system.

j. What, if any, quality control activities? any standards and guidelines in use?	RM: Obviously the prototype building phase is rather straight forward and not many real functionality to build, so there seems not much quality issues to control about. As to the whole project our control is done by sign-offs and reviews. We don't have any standards or guidelines in use. But, as I said before, we are working on it.
k. How about change request control? any procedures or guidelines on both managers, developers and customer/user?	RM: If it is functional changes the manager will make the decisions, often with customer because it normally has cost implications. Once the prototypes are signed off as part of the specification, we freeze the specification, and then go through the hard analysis and design etc. i.e. conventional life cycle. Although we still have regular meeting with users, the purpose is purely for quality checks.
	SS: If some change requests from customer or user come in the middle of the development, then we have to have agreement on cost. We don't have any guidelines or procedures on the change request control, we do it on project basis. One guideline I would like to give developers is have a time limit for the prototypes building, the reason for this is to limit customer's tendency of constantly changing mind.
l. Do you still have customer/user's involvement after signing off the requirement specification (mainly the prototype(s)) ?	RM and SS: Yes, the customer and user are still frequently involved in the implementation and, in fact, right through the project.

m. If yes, how and why?	RM: They are involved in several ways: via telephone, e-mail, and meetings. The main purpose is to clarify the requirements.
n. What if the they want have fundamental changes or do some add-ons?	SS: Well, it's not normally the case. As RM said, they are involved in the implementation stage mainly for the requirement clarification. However, if such things happen — there were a couple of instances — we would treat those changes as new requirements and do it as if it were a new project, and prototyping if needed. Of course our main concern is the cost.

4.1.4 Other Information

RM:

It is interesting to notice that you put the activity "write functional specification" before the prototype building starts, and refine it around the loops which is what we suppose to do...

SS:

We find that to take full advantage of prototyping the sooner the user is involved the better the results is... while one advantage of using prototyping is to be able to show user what is possible to achieve, another, probably more important, is to have them buy into the big project.

There was a case which I experienced that we put a system into a company, where prototyping was not used. We went the company with the system we built based on their request. The system was for twenty users. When we sat down with the users to show them the system, they said this was the first time they were told about the new system, and they all say it didn't work like that here. So the project was a total failure because it was not what users want.

We may use prototyping to establish the technical viability, to chose the final tool or mixed tools we want use, to determine the type of the database we want use, and estimate the cost. For example, I just had a meeting with a customer, he wanted me give him an estimated cost for a system of 16 applications.

So we may use prototyping to see how well they understood the system and show them how big it may be, and by the end of prototyping we probably throw the prototypes away or leave it as part of the specification. Of course, in most cases prototypes has been used rather than throw-away.

... prototyping enable users change their mind easily without much early commitment....

RM:
I think what we are going to talk about is what we are going to do rather than what we have done. The lack of control already has caused us significant problems: over-budget, late, what we meant prototyping was not what client meant by prototyping and it was not clearly defined at the start of the project, there basically wasn't any control that's why we need put some controls in place.

SS:
People like it because they don't have to think too much about what is they want, they can wait until having the prototypes of the system; the fact is that they don't have much money to spend on, they want do small amount of work cheaply at a time before they move to next stage. This can cause problems because that means we don't have enough time at the beginning of the project to write a specification.

So what happened was that we were often led into building prototypes without having clearly defined sets of requirements. What we trying to do now is to trying make our customer understand that there are some prerequisite tasks to be carried out at the right beginning of a project.

One of the tasks we are enforcing is to insist the customer write a definition of the project, though we are embarking on prototyping which is unstructured approach, i.e. we don't have a full specifications to start with, we would like to have a overall agreement to what the whole project is all about in term of the scope and the objectives.

One thing we thought we could get away with was not to have the specifications and it seemed attraction to people, but it has not been worked. (RM: no absolutely not!) The problem has been that there is no baseline to refer back to therefore lost control. Another thing we have to recognise is that prototyping in commercial is very different with prototyping in IT department with a large company or organisation: we simply cannot afford the time and money: to have a cup of tea and go around the loops.

Most our prototyping effort is between 5—10 man-days, average is about 7 man-days, which is about 10 percent of the whole project effort.

If the application is built up using Lotus Notes, much of the prototypes can be put into use, but our rule of thumb is not to use the prototypes as the delivered system, if it is needed to use it we will use it as a pilot system along with some appropriate documents.

Talking about the standard and procedures, I know it is very different between the commercial environment like us and a IT department in a large organisation. I worked in IT department, they put as much as possible standards and regulations in, and made a project as long as possible, because they are paid full salary and they don't have to concern about the cost and the time. Whereas here we have to get our work done with a fixed amount money and within a certain time limit.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

4.2 Follow-up Agreement

They agree to arrange a date for next meeting:

a. to conduct the personality test

b. to collect some documents

4.3 Comments on the meeting

4.3.1 About The RAD model

4.3.2 About The Personality Test

4.3.3 About The Questions and Answers

(see 4.1.2.a) The company is going to adopt TickIT as managing and control baseline. Because they seem to have no any kind of standards or procedures so far, it is likely to jump to a new standards blindly, which may cause more suffer than benefits.

(see 4.1.2.b) Their answers to this question is apparently plausible —not to give a definite delivery date as a safety-guard and put off customer's demand for short time deliver for it often more wishful than possible. The dilemma here is that the danger of loosing the competence is equally obvious. However this problem is, in my opinion, not insoluble if we look under the surface of the matter.

There are two important factors here: one is that their whole system development approach seems entirely based on individual experience, and the other is that projects are mostly undertake by one developer; the former may result in inadequate estimates, and the latter may cause long-run project. The treatment is clear: having more rigorous methods for more accurate estimates; increasing the team size to shorten projects length and therefore to meet customers needs. Of course, in both cases, a sound management and control procedures and guidelines has to be enforced to ensure its success.

4.3.4 About Other Information

4.3.5 About The Supporting Documents

4.4 Key Issues Raised from the meeting
a. How to tailor the standards to their own taste, i.e. how to make it practically useful?
b. Is it a potential danger of loosing customers for not allowing their "bad planning"? if not, why; if yes, how to keep competent?(4.1.2.b)

Interview Form D3

1 Meeting Details

Date	3/3/94
Duration	120 minutes
Company Reference	D
Department	
Meeting Reference	D3

2 Interviewee Brief

Interviewee Reference	RM
Job Title	Technical Director
Process Role	Project managing, prototyping
Other Information	
Interviewee Reference	SS
Job Title	Sales Manager
Process Role	Managing
Other Information	
Interviewee Reference	GG
Job Title	Application development manager
Process Role	Managing, prototyping
Other Information	
Interviewee Reference	IC
Job Title	Application developer
Process Role	prototyping
Other Information	
Interviewee Reference	SG
Job Title	Application developer
Process Role	prototyping
Other Information	
Interviewee Reference	MO
Job Title	Application developer
Process Role	prototyping
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Carry out the Personality Test

b. Get some supporting documents

c. Ask a few questions[3]

d. Refine the RAD model

3.2 Following-up Plan

None. This is the last meeting intended.

3.3 Data Gathering Methods

a. Tape Recording

b. RADs Modelling

c. Personality Test

4 Summary

4.1 Information Gathered

4.1.1 The RAD Model

4.1.2 The Personality Test Results

Reference	Personality type
RM	ISTJ
SS	ESTJ
GG	ENTJ
IC	ENTJ
SG	ENTJ
MO	ISTJ

4.1.3 The Key Questions asked and Answers replied

Questions	Answers
a. How long have you been practising software prototyping?	IC: I have been with Company D for one and a half years. Most of the time I've been involved in the implementation, the activity according to your diagram here [refer to the activity(black
	box): implementation in RADs of Company D], rather than the prototype building.
a.1: Any user's involvement in your work?	No, I just coding according the specifications — mostly the prototypes. (a.1 — a.5 are sub-questions asked to IC.)

[3]The questions to be asked will based the questionnaires further investigation, which is given in section 5.1. Ideally I would like to run through the whole questionnaires, but given the time limit (an hour for three or more developers) only a few questions can be asked.

a.2: Who you contact to in your work?	The prototyper.
a.3: Do you know that some projects, from start to finish, have been carried out just by one person?	Yes, it all depends on the projects. But I haven't done it so far.
a.4: Do you build on top of the prototypes or throw the prototypes away?	Build on top. The prototypes are just screens, I code the functions behind it.
a.5: Any quality checks at all?	No.
	SG: [answer to question a] I joined the company last year this time when I graduated with BSc in computer science. So I have one year commercial experience.
b. What are the sizes of the project you involved in? and how much effort for the prototype building?	SG: I've just involved one project for about five months, the project is internal, so most department has been involved. There two of us as developers, and sometimes RM help us out. GG: The sizes is really depends, but I would say that on average, the effort for the prototype building is about 5-10 percent of the whole project.
c. Any comments on the RADs model[4]? How you play your role in the process?	SG: Yes, the RAD model looks fine to me. I perform the prototyping role, my work involve the prototypes building and its implementation. But not involving the sign off.

[4]This was the first version RAD model which had been modified based on the previous meeting with RM and SS. Before ask this question I explained the RAD notations and the Model to them, and let them have a few minutes to look through.

	GG: Yes, it's OK. My roles on the diagram are managing and prototyping. Within the managing role I involve in gathering the initial requirements and signing off the specifications, and, in prototyping I'm mainly involved in interviewing customers and building the prototypes, and I rarely do any coding.
d. How do you conduct the initial requirements gathering?	GG: The key at start seems to me is to get the right people from the customer. After the customer makes the request, I will have some meetings with both the customer and the users to discuss what they want. I normally take notes in the meetings, and how much notes depends on how much the customer and user are prepared—sometimes they have no documented requirements, they just talk about it, while some have documented requirements with various level of details. So, the contents of the process may vary according to the project itself and, more importantly, the people you talk to. The interaction here for me is mostly just once, i.e. have one meeting before the building. Again it depends on projects, it could be a couple or more times. I would like to go into some level of detail before the building rather than get a bit then start.
e. I know that you don't have any company wide life cycle methods, standards/procedures to follow. But do you follow any methods or standard yourself?	SG: Not explicitly, mainly use the experience. T3E: I have been trained in SSADM and some other development methods but I prefer the SSADM and use some part of it for the work.

f. Do you think they that it is important to have some company wide methods and standards?	MO: Yes, I think so. Now seems OK because of the size of project, but when the size grows and more people in a team, it will cause big problem without standards. I am not sure about the company wide but at least team wide standards is essential. But the company seems not prepared to pay for it. CL: Not yet, but we will do.
g. What happened if there were some change requests during the implementation?	GG: Once the requirement specifications (prototype) is signed off, the customer and user will not normally be involved. If they want some changes, as sometimes is the case, the requests will go through the whole process as if it were a new project except we won't do the prototype unless they ask.
h. How about quality issues?	SG: There is no quality concerns the moment, we just get the system working by the deadlines. Because, apart from we haven't got quality checks for the development, the customer is

4.1.4 Other Information

When I showed RM the second version of the RAD, he commented:

a. that the loop around the initial requirement gathering is normally twice, and maximum three times.

b. the activity — write function specifications — is not there most of the time.

c. The sign-off the requirements specifications on the diagram is basically signing off the prototypes.

RM also mentioned that: In future, we are going to put two or three people to do the specification, and the rest to do the development. So we'll have analyst, designers and programmers. The reason is the department is getting too big and the project is getting too big for one person one project.

When I asked about the standards and procedures that they have been trying to develop, RM said the progress has not been made as far as they hoped because it is very difficult task.

4.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

Proposal for "Prototype Executive Information Systems"	**a.** Introduction **b.** Existing System **c.** Project Proposal & Plan **d.** Outline Specification **e.** Hardware & Software Requirements

4.2 Follow-up Agreement

a. GG, the application development manager, agreed to send me some relevant documents such as project proposal, functional specifications and sign-offs.

b. Most of them agreed that future contact can be made directly to them if it is needed.

4.3 Comments on the meeting

This was the third and last interview with The company. The main purpose of this visit were (1) to chase up some documents that SS (the sales manager) agreed to send to me, (2) to collect any other relevant supporting documents and (3) to conduct the personality test with as much as possible software developers.

4.3.1 About The RAD model

4.3.2 <u>About The Personality Test</u>

Here I just want to point out one interesting phenomena seen from the test results is the similarity: all of them have 'T's (Thinking) and 'J's(Judging) (for detailed score see the separate sheet). Also there are four 'E's (Extroversion), three 'S's (Sensing) and two 'I's (Introversion); None of them have 'F' (Feeling) or 'P' (perceptive) in their type.

Furthermore three of have exactly the same type: ENTJ, and the rest are almost the same: two ISTJ and one ESTJ , obviously the difference here is the latter have 'E' instead of 'I' in the former. In addition, most prototypers — three out of five — have attributes 'E'(extroversion) and 'N' (intuition).

From the analysis above, one simplest possible explanation is that (1) for both managers and developers attributes 'T' and 'J' may be prerequisite, while (2) 'E' and 'N' may be essential quality to prototypers. Of course more representative explanations will be given when all the other test results (from other four companies) are compared.

4.3.3 About The Questions and Answers

4.3.4 <u>About Other Information</u>

Ref. 4.1.3 c. The impression was they had some pressure from top for enforcing the standards, and the difficulty was that:

(1) no previous experience;

(2) too busy — lots projects to do;

4.3.5 About The Supporting Documents
4.4 Key Issues Raised from the meeting **a.** How should we avoid the danger of adopting available standards without careful considerations for tailoring to its own need?

Interview Form E1

1 Meeting Details

Date	4/11/94
Duration	45 minutes
Location	Poole, the company's office
Company Reference	
Meeting Reference	E1

2 Interviewee Brief

Interviewee Reference	MG
Job Title	IT Manager
Process Role	Project Managing, Process Design
Department	IT
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Introduce myself and the research;

b. Get some background information of the company, the department and their prototyping practice;

c. Define the co-operation.

3.2 Following-up Plan

a. To get a draft RADs model of their practice.

b. To talk to some of the MG's colleagues involving in the process.

4 Data Gathering Methods

Interviewing with tape-recording

5 Summary

5.1 Information Gathered

The information gathered is arranged in following:

5.1.1 The RAD Model

5.1.2 The Personality Test Results

Reference	**Personality type**

5.1.3 The Key Questions asked and Answers replied

Questions	**Answers**
Q1: What is your application domain?	A1MG: information systems.

Q2: What is the purpose of prototyping?	A2MG: We use software prototyping mainly for : (1) Feasibility study (2) Process Design.
Q3: How long software prototyping has been used here?	A3MG: About four years — from late 1990 up to new.
Q4: How about the method and standards for prototyping?	A4MG: We have a company-wide system development methods and standards in use. The methods and standards had been established from beginning of 1990 (?) to end of 1992. The reason behind was to provide more flexible and practical system development framework.

5.1.4 Other Information

When I explained what I want to model was the process of what happened or happening rather what should happen, then he said, humorously, "what happened was totally chaotic".
He said that he would prefer some "short sharp sessions" to long ones — from 30 to 60 minutes — in our future meetings.

5.1.5 Supporting Documents from the Interviewee and their Contents

Documents	**Contents**

5.2 Follow-up Agreement

a. Mark agreed to send me the company-wide standards and procedures:
(1) System Life Cycle Methodology Standards;
(2) System Life Cycle Management & Techniques.
b. Meeting date, length of meeting and information to provide are subject to the company's convenience. Number of the visit was not limited.
c. I would send a report in the end of the field work.
d. Arranged a date for next meeting—modelling a draft RADs

5.3 Comments on the meeting

5.3.1 About The RAD model

5.3.2 About The Personality Test

5.3.3 About The Questions and Answers

5.3.4 About Other Information

This is the first meeting with MG, the IT manager of the company, aiming to find out the way of our further co-operation. To this end the purpose of the meeting is well served. There are two reasons for me to try to look into Company E's software prototyping practice. One simple reason was that it is accessible in term of their willingness and the distance. The other reason is its unique organisational settings: IT department in a large company.
5.3.5 About The Supporting Documents
5.4 Key Issues Raised from the meeting
Why he said "what happened was totally chaotic"? what was the problems?

Interview Form E2

1 Meeting Details

Date	22/11/94
Duration	45 minutes
Location	Poole
Company Reference	E
Meeting Reference	E2

2 Interviewee Brief

Interviewee Reference	MG
Job Title	IT manager
Process Role	Managing, process designing
Department	IT
Other Information	

3 Meeting Agenda

3.1 Purpose of the Meeting

a. Get the draft RADs[5] model of the process

3.2 Following-up Plan

Making the next meeting date for:

a. Clarify the first version RADs

b. Modify and/or extend the first version RADs

4 Data Gathering Methods

a. The RADs modelling

b. Interviewing with Tape Recording

5 Summary

5.1 Information Gathered

The information gathered is arranged in following:

5.1.1 The RAD Model

Most information gathered in the meeting were shown on the RADs model (see appendix D).

5.1.2 The Personality Test Results

Reference	Personality type

5.1.3 The Key Questions asked and Answers replied

Questions	Answers

[5] The draft RADs is a hand-drawing RADs produced at the time of meeting

a. What are the distinct roles involved in a typical prototyping project?	We have not explicitly used prototyping approach to our systems development. This is the first time we practise such an approach as defined in our recently developed methods and standards. For the current project, I can identify seven distinct process roles. They are: Business control, project managing, process design, system developing, training, customer and end-user.
b. What are the main functions of each of the roles?	They are defined in our methodology, please refer to the document I sent you.
c. How is a project initiated?	This project was initiated by our business control board as an IT project to satisfy our own business needs.
d. How initial requirements are gathered?	By meetings with all relevant business representatives, company directors and me (the IT manager).
e. What you give to the prototyper to start to build the prototype(s)?	Written specifications.
f. What the prototypers do on their own? who they contact to?	Coding and testing the prototypes. They'll normally have direct contact with customer and/or end-user for gathering and clarifying requirements, and with their group leader or design manager mostly for technical problems.
g. What and who is involved in the prototype(s) evaluated?	Normally the customer and user(s), managers and prototyper(s).
h. Do you sign off the prototypes with customer as (or as part of) specifications?	Yes, we do.
i. Any change request come to you either from the prototyper or user group or end users?	Mostly from design manager or group leader, customer.

j. What and who involved in the decision making?	The business control board, managers and customer, and sometime the prototypers;
k. Who involves in the final sign off?	It will be me (IT manager) and customer;
l. Could you indicate any other explicit controls on the diagram?	No.
5.1.4 Other Information	
5.1.5 Supporting Documents from the Interviewee and their Contents	
Documents	**Contents**
Methodology Standards and Management & Techniques	
5.2 Follow-up Agreement[6]	
A date were arranged for the clarification and refinement of the first version of the RADs model.	
5.3 Comments on the meeting The comments are brief accounts on those data collected. They are arranged in corresponding to the data categories:	
5.3.1 About The RAD model This was the second meeting with MG, which resulted in a draft RADs of their prototyping process. Because of the time available, activities and interactions were not modelled detail enough to make a comparison at this stage.	
5.3.2 About The Personality Test	
5.3.3 About The Questions and Answers	
5.3.4 About Other Information	

[6]the following meeting with MG was cancelled due to his leaving the company, and consequently further co-operation was also unable to proceed.

5.3.5 About The Supporting Documents

A copy of the company wide System Life Cycle "Methodology Standards" and "Management & Techniques" were obtained from MG. Having looked through the Module 4 of the Methodology Standards (which is the stage where prototyping is explicitly employed), the following statistics were observed: about 18 roles, 14 activities and 44 task appeared in this module. Each activities and tasks are elaborated. However roles and their interface are not clearly defined. In comparison, a striking discrepancy was shown in term of the number of roles: 18 in the method defined and only seven roles identified in the draft RADs. This raised some issues (the key issues is listed in the following section) to further clarification before tidying up the draft RADs.

5.4 Key Issues Raised from the meeting

a. What are the following roles which appear in the documents but not clearly defined?

1. project leader, 2. subsystem leader, 3. business analyst, 4. data administrator, 5. database administrator, 6. data analyst, 7. database designer, 8. operation analyst, 9. operations planner, 10. user representative, 11. system developer, 12. system project co-ordinator, 13. project director, 14. senior business management, 15. system designer, 16. system tester, 17. trainer, 18. user support analyst

b. What are the relationships of these roles in relation to those roles used in the first draft?

c. Why particularly such a "role collapsing " happen?

d. How this has been done and why?

Appendix E(c) Guidelines for Further investigations

Guidelines for Further investigations

The following guidelines comprises the key questions and the reasons behind for data collection during Further Investigations. These questions were categorised under each key management and control areas identified, which was based on the findings of the previous stage — Field Modelling.

1) Prototyping initiation

Team member selection criteria, configuration and resource allocation

1. How is a prototyping project team made up? i.e. the people involved, the control structure and resource allocation.
2. How and why individual prototypers are chosen?

Guidelines and responsibilities

1. What, if any, guidelines and principles are specifically made and used for prototyping projects? if no, why?
2. If yes, do they include clearly defined responsibilities for the team members especially managers and prototypers?

purpose:

a. to provide better prototyping team selection criteria;
b. to provide clearer guidelines for both managers and prototypers; i.e. who , what and when;

2) Initial user requirements gathering

Methods

1. How are the initial user requirements gathered?

Statistics

1. How many iterations?
2. What is the ratio of initial requirement gathering compared with the final user requirements?
3. What is the ratio of time spent on this period compared with the whole prototyping effort?
4. How much prototyping effort compare with the whole project?
5. Are these statistics relatively stable across projects?

purpose:

a. to be able to suggest better practice for initial requirement gathering based on both qualitative and quantitative data;

3) First prototype building

Methods and key activities

1. What are the key activities involved?

2. Is requirement analysis and design a distinct activity within prototyping construction? if yes, how are they carried out?

Statistics

3. How much is the analysis and design effort taken compared with the whole first prototype building phase?
4. How long the first prototype take compared with the whole prototyping effort?

Quality issues

5. What, if any, quality assurance activities are carried out? (If the prototype is not throwaway).
6. What, if any, standards and or guidelines are used?

purpose:
a. to provide guidelines such as what those distinct activities are and what care should be taken of for each activity;
b. to provide guidelines on what the minimum managerial data are needed;

4) Change request control

Methods

1. How are CR decisions made?
2. How does this differ from non-prototyping projects?
3. What kind of changes are requested to managers by prototypers /customer /users ?
4. What, if any, explicit criteria are used for the control?
5. What, if any, explicit quality considerations are taken in making control decisions?

Statistics

1. What is the variations the different types of changes over each iteration?
2. What is the ratio of total changes compared with the core functionality changes over each iteration?

purpose:
a. to identify the key factors for achieving better process efficiency and better product quality;
b. to provide guidelines on more efficient CR decision making.

5) Customer and/or end-users perspective

1. How customer/users are involved in each phase identified above and why?

purpose:

a. to provide clearer guidelines on how customer/users can be involved more efficiently.

Appendix F(a) Summary Report of the Personality Test

Summary Report of the Personality Test

1. Background and purpose of the test

The issue of personality of the process participants came about as a result of the field modelling (chapter 5) where the prototyping team make-up was identified as one of the key control areas. Thus, the personality test was included as part of the further investigation. The main aim for such a test was to try to understand and learn about the significance and impact of personality on process roles and their interactions.

2. Meyers-Briggs Personality Test

The test tool used for the study was the Meyers-Briggs Personality Test. This test is widely recognised[7] and used for categorising personality types among various social groups. The test consists 50 yes-no questions. The answers to these questions give scores in 8 distinct personality attributes or 4 opposing pairs. They are: E (extrovert) vs. I (introvert), S (sensing) vs. N (intuition), T (thinking) vs. F (feeling), and J (judging) vs. P (perception). Each personality type is a combination of 4 different attributes — one out of each pairs, thus it gives a total of 16 personality types. For example, if Fred has a higher score of: E to I, N to S, T to F, and P to J, then it would give him a personality type of ENTP.

Obviously, these 16 types are merely a gross simplification of human personalities, and it is not for the identification of individuality, rather it is used for grouping 'similar' individuals under each type. For more details about the test itself — the questions and the personality type explanations — refer to the appendix F(b).

There are some other similar personality test such as Keirsey Temperament Sorter - Jungian Personality Test[8] which, apart from having slightly

---

[7]One piece of evidence is that the personality test is on hundreds of Internet sites from various organisations to individual home pages. A single Lycos search on the subject gave about 5000 matching items.

[8]which can be found on the Internet at:
http://sunsite.unc.edu/personality/keirsey.html

different question construction, is virtually the same as the Meyers-Briggs Personality Test.

3. Limitations on the test results

Three out of the five intended companies (with which the further investigation being carried out) took part in the test. The total number of participants were 12 — 5 managers and 7 prototypers. These account for only two type of process roles: manager and prototypers. The other two typical roles, i.e. customer and end-user, were unable to be included due to their availability and the limited time on part of the researcher. In addition, each participant was only tested once, which might cause some degree of inaccuracy, especially when a pair has a close score. For example, if one scored a P = 9 vs. J = 10 then according to the test one would have J instead P element in one's personality type, however the reverse might happen if one is tested again. Therefore to achieve a reasonable stable result, repeating tests should be carried out. Again, for same reasons as mentioned above, this was impossible at the time.

4. Data (the test results) organisation

The data are grouped into three categories: managers, prototypers, and combined. They are presented in 11 charts. The reason for the first two groups — managers and prototypers — is that they correspond to two common process roles, and for the third — the combined group — is that their roles often overlap.

5. Result Analysis

The analysis is based on the data obtained with reference to some of the interview summaries. Here the three groups stated above are analysed at two levels: the personality attributes and the personality types. In other words, it is intended to look at the possible interpretations of personality type (e.g. ENTJ) distributions among each group and the distribution of each attributes — E, T, P, J etc. — among each group. Having analysed the personality types and their attributes distribution, the following are observed:

(1) The managers' personality types seem to be evenly distributed (see chart 2), while the prototypers' seem to be clustered (see chart 1) among the 16 personality types. Furthermore, there is only one personality overlap — the ISTJ — between the two groups. This may suggest that certain roles are more likely to fall into certain personality types, and further study of the common characteristics and behavioural patterns within and between each group may reveal some useful knowledge. For example, conclusions may be drawn such as: type A manager tend to work better with type B, C, D of prototypers, and type B and C prototypers are likely to have conflict.

Chart 1 Prototypers Personality Type Distribution

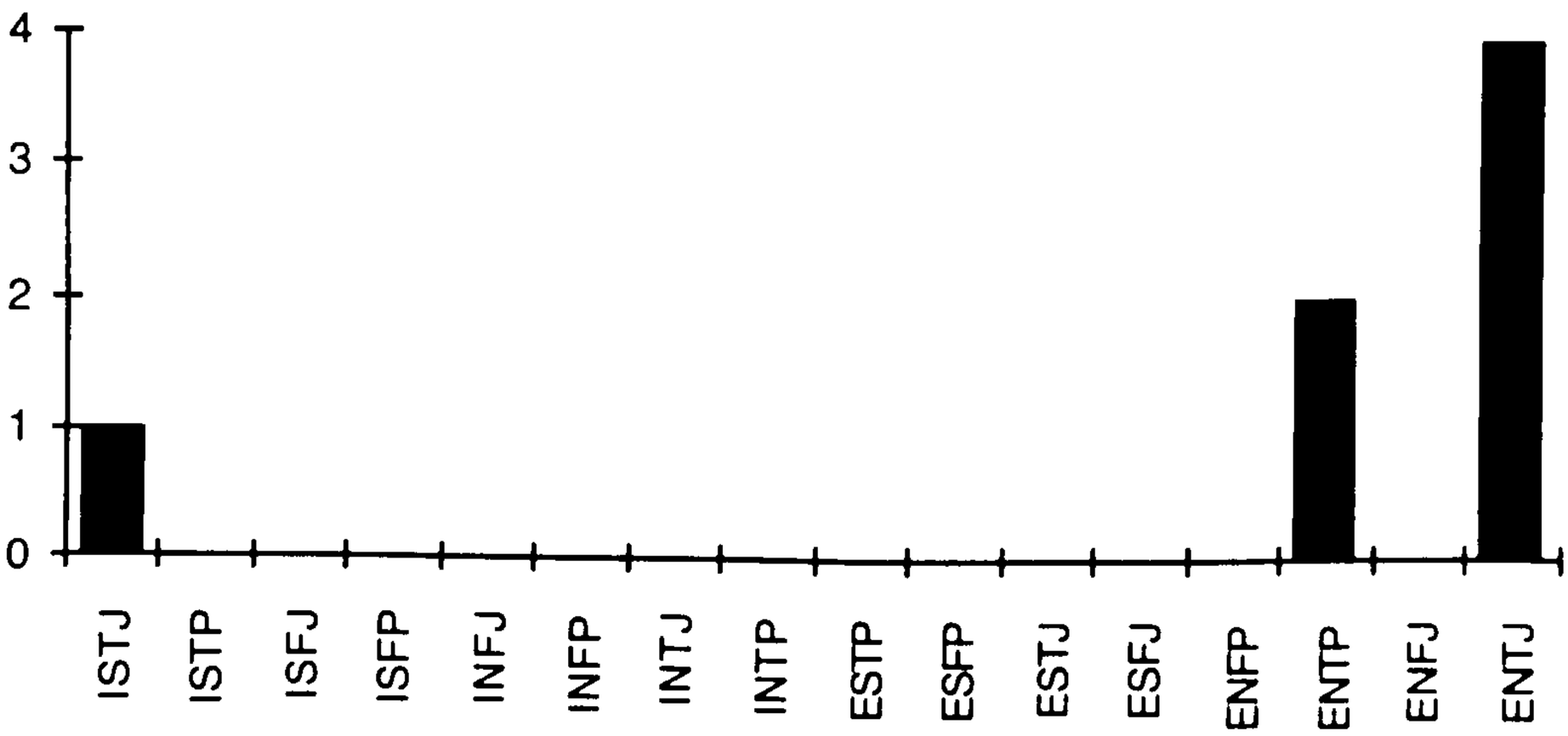
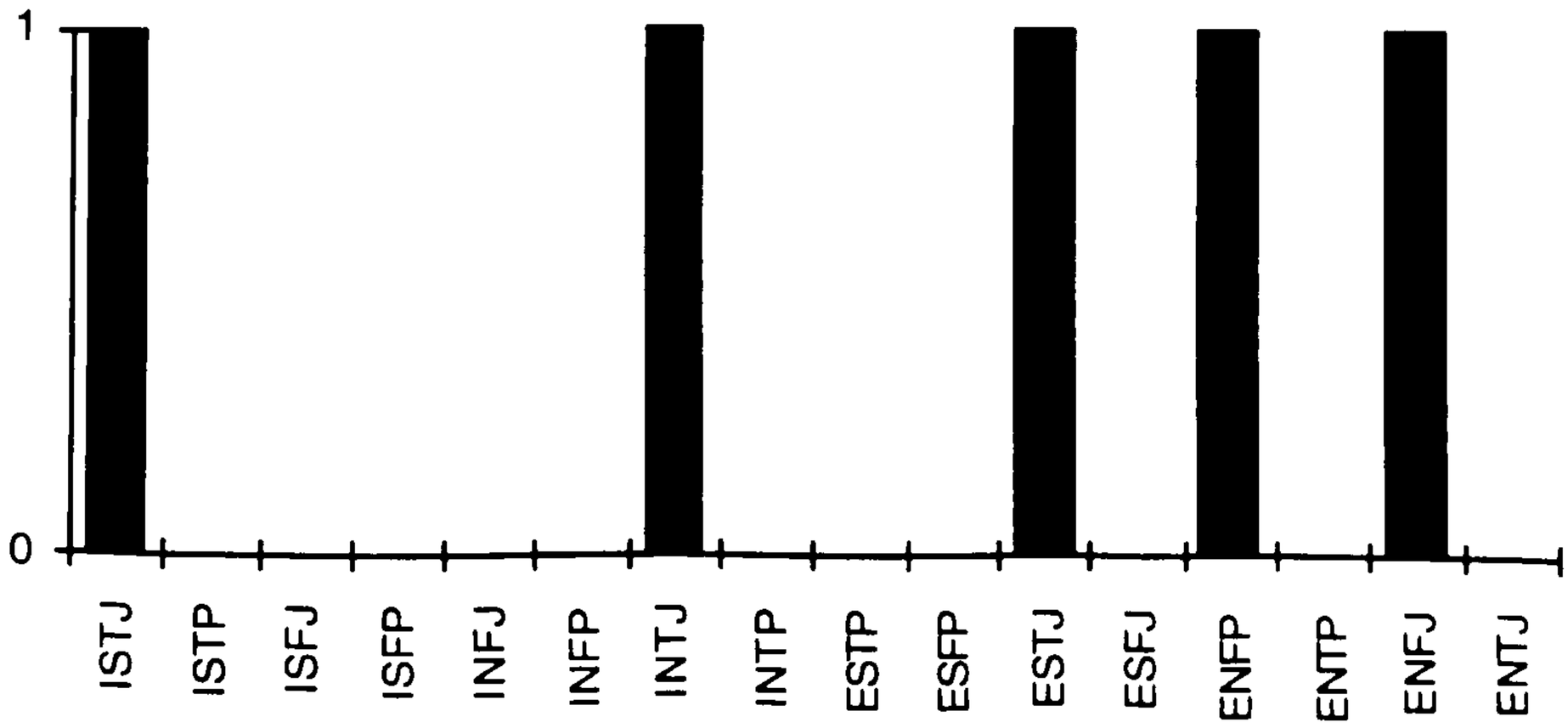


Chart 2 Managers Personality Type Distribution



(2) ENTJ was the most frequently occurring personality type within and across companies. Chart 1 shows clearly that ENTJ was the most frequently occurring types — 4 over 12 or 33%. From chart 4 it is clear that ENTJ was

also the only type appeared in all the companies. From the types explanation (see the appendix F(b)), this type of person is described as quick, decisive, good at group activity.

Chart 3 Number of Occurrences of Each Personality Type

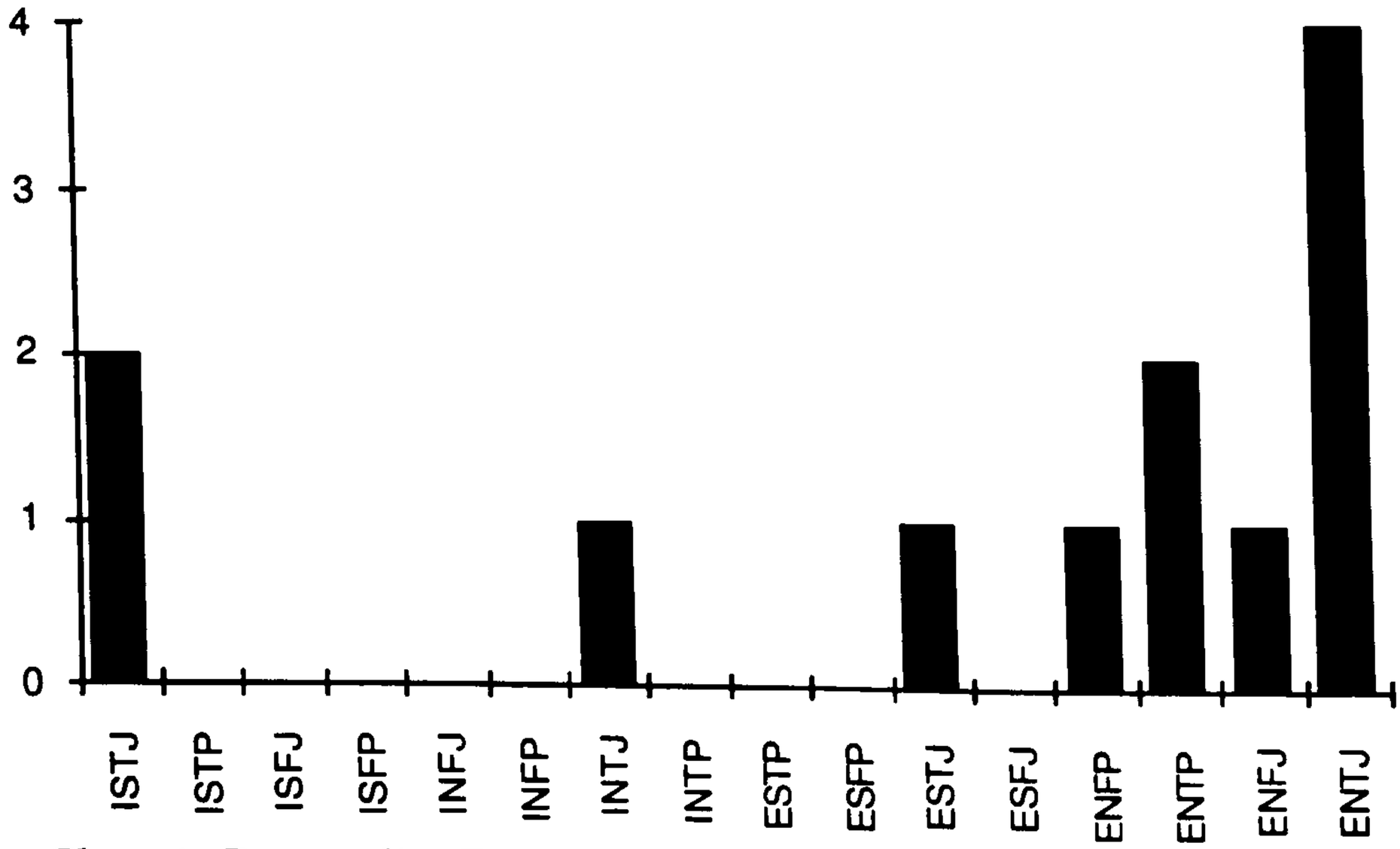
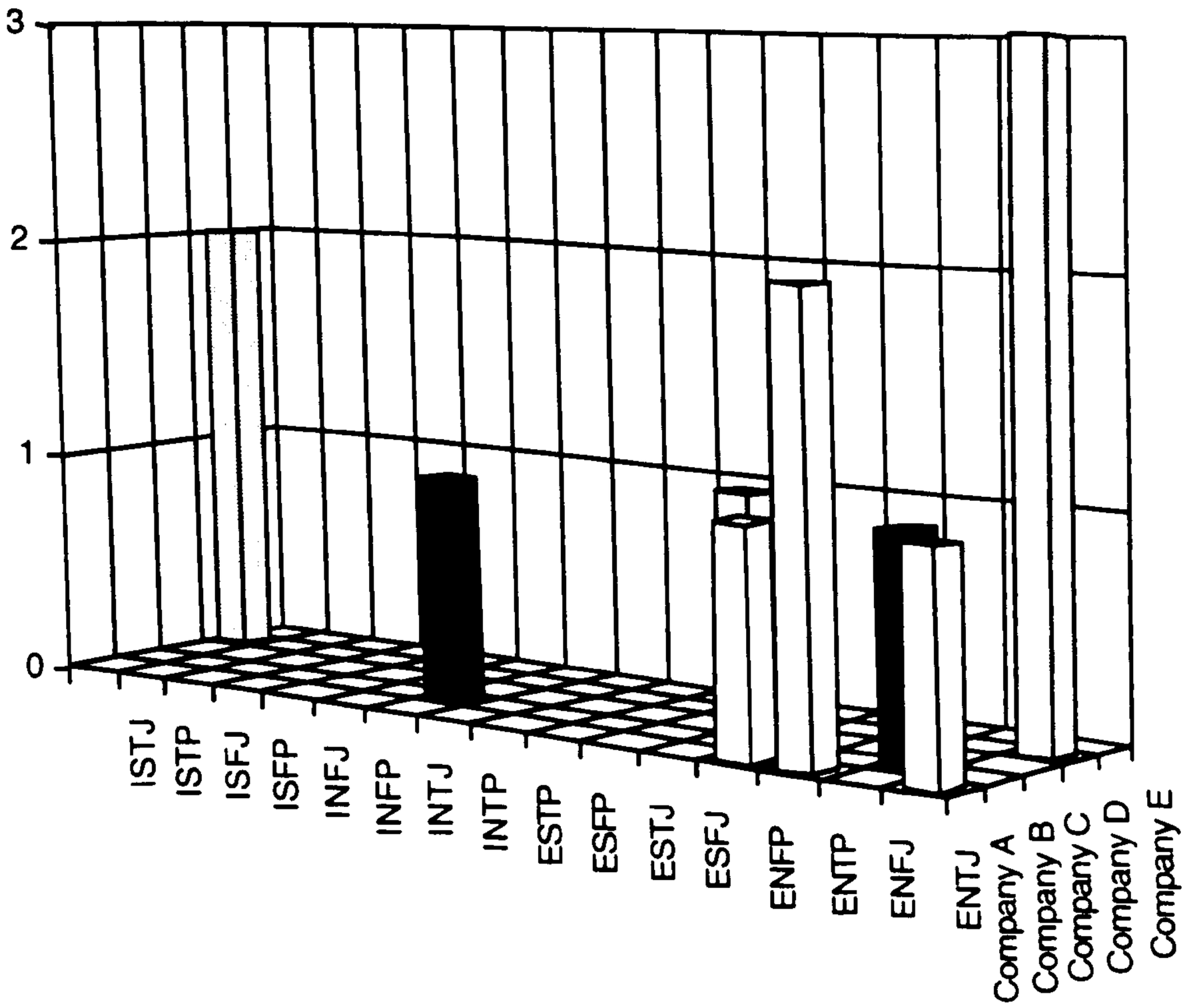


Chart 4 Personality Type Distribution Within Each Company



(3) ENTJ and ENTP were most common among prototypers. All 4 ENTJ types (more than half of the prototypers) and 2 ENTP (2 out of 7 prototypers) fell into the prototypers group (see chart 1). Apart from the fact that these two types are roughly a mean of the two extreme types: ESTJ the total extrovert or the opposite INFP, there seems no obvious explanation at this stage. One possible conjecture is that it is firstly because normally there are fewer managers than prototypers in an organisation thus less choice for managers, and secondly the prototypers in a project team are normally chosen by their managers. This implies that managers may have been aware of the team members personality types either consciously or unconsciously. For example, KR [the manger in company A] let the prototypers freely associate their team-mates, and the prototypers felt happy about it [refer to Interview Form A3 (4.1.4: about the team makeup)]. So making managers consciously aware of team members personality types in selecting the team members might reduce the chance of personality clash and therefore increase the team performance on the whole.

(4) Managers had more diverse personality types. By contrasting chart 1 and chart 2, it is clear that, while all 7 prototypers fall into three types, all the 5 managers have different personality types (chart 2). This, on one hand, might suggest that the clustered personality type in prototypers may reduce some control variables. On the other hand, various managers' types and their power over project control indicate the likelihood of the various individual personalities having an effects on their managing practice. For example, manager KR of company A has a personality type of ENFP, he seemed to put much of his emphasis on the participants' passion or enthusiasm for job satisfaction [refer to Interview Form A2 (4.1.3:e); A3 (4.1.3:c2)], while RM — manager of company C, type ISTJ — is more concerned with rigorous control such as procedures and standards [refer to Interview Form D2: 4.1.3: f and D3: 4.14: RM]. Note that ISTJ and ENFP are completely opposite types. We can see the apparent link between their personality types and their philosophy towards their managing and controlling practice. Both styles have their advantages: one may bring more individual's potential power into play; the other may have more measurable product quality. The counterpart disadvantages are the former may more likely make quality unknown, the latter may be time-consuming and possibly have a detrimental effect on prototypers' motivation. Although it is not yet known how much the *impact* the different

personality types have on the development process, the awareness of this phenomena could make amends for one's weakness by exploiting one's strengths.

The following presents some more observations from the attribute level of the personality types.

(5) Extrovert attributes E, J, N, T were much stronger than their opposing attributes I, P, S, F in the mixed group — managers and prototypers. Looking at chart 5 and chart 6 (on next page), it is obvious that attributes E, J, N, T have a much higher occurrence than their counterparts: I, P, S, F. Obviously T (thinking) and J (Judging) are essential, E (extrovert) is probably related to the nature of such a process where intensive interactions occur during the process at all levels. N (Intuition), by definition, means immediate apprehension without reasoning. Therefore the notable high score of N(intuition) may indicate that this kind of quickness is also one of the prevalent qualities among prototyping project team members, especially the prototypers.

Chart 5 Total Number of Each Personality Attribute

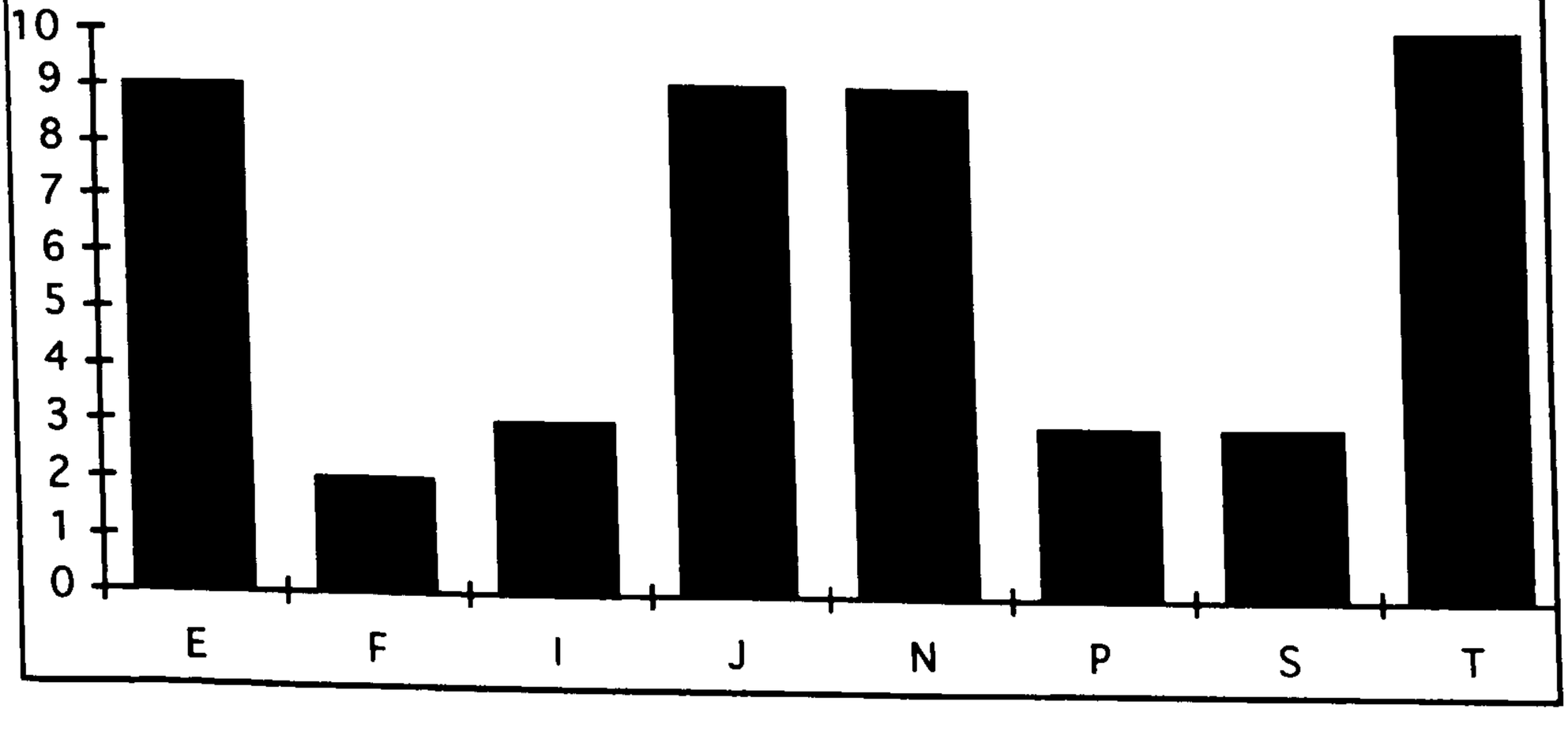
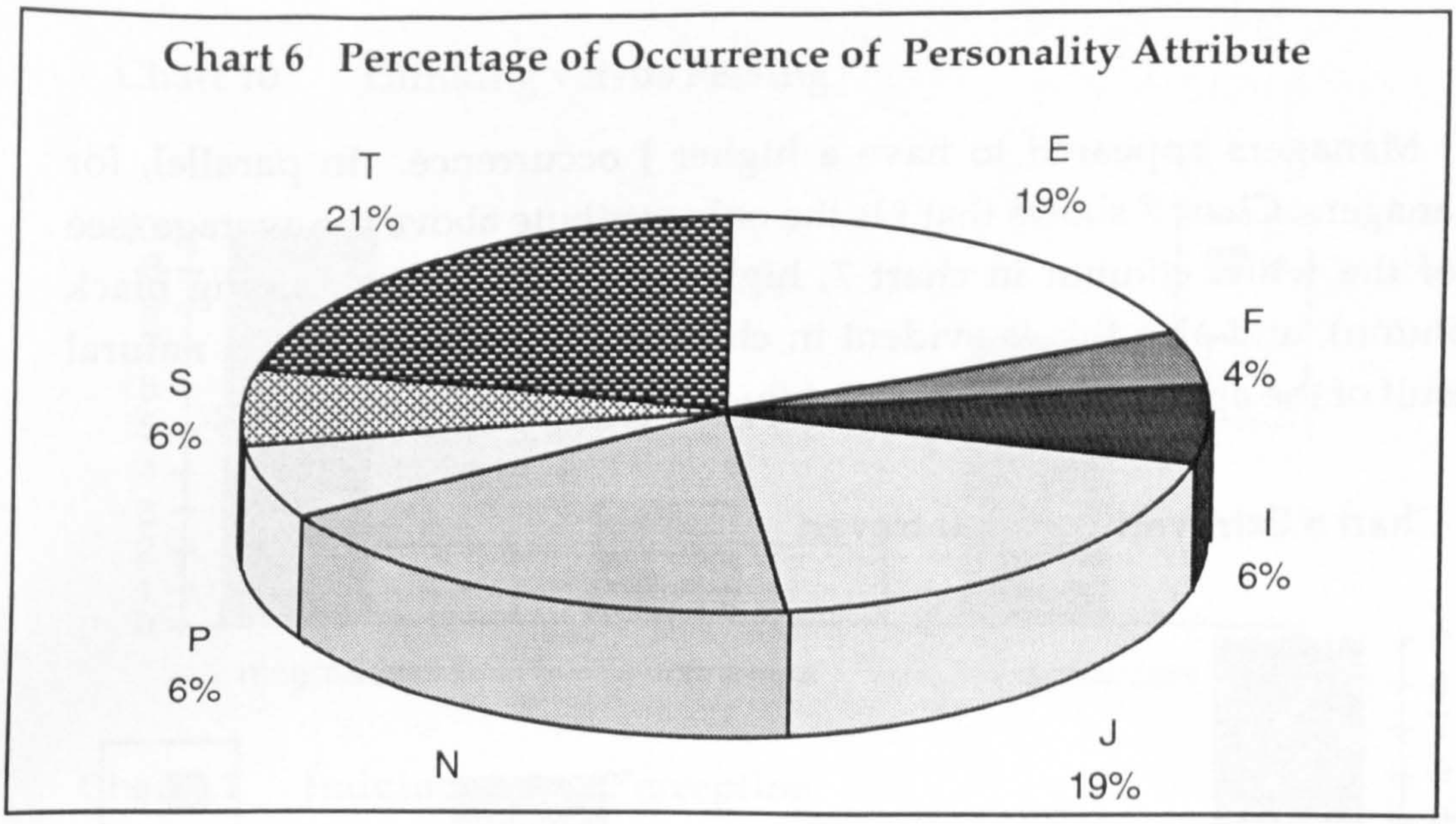
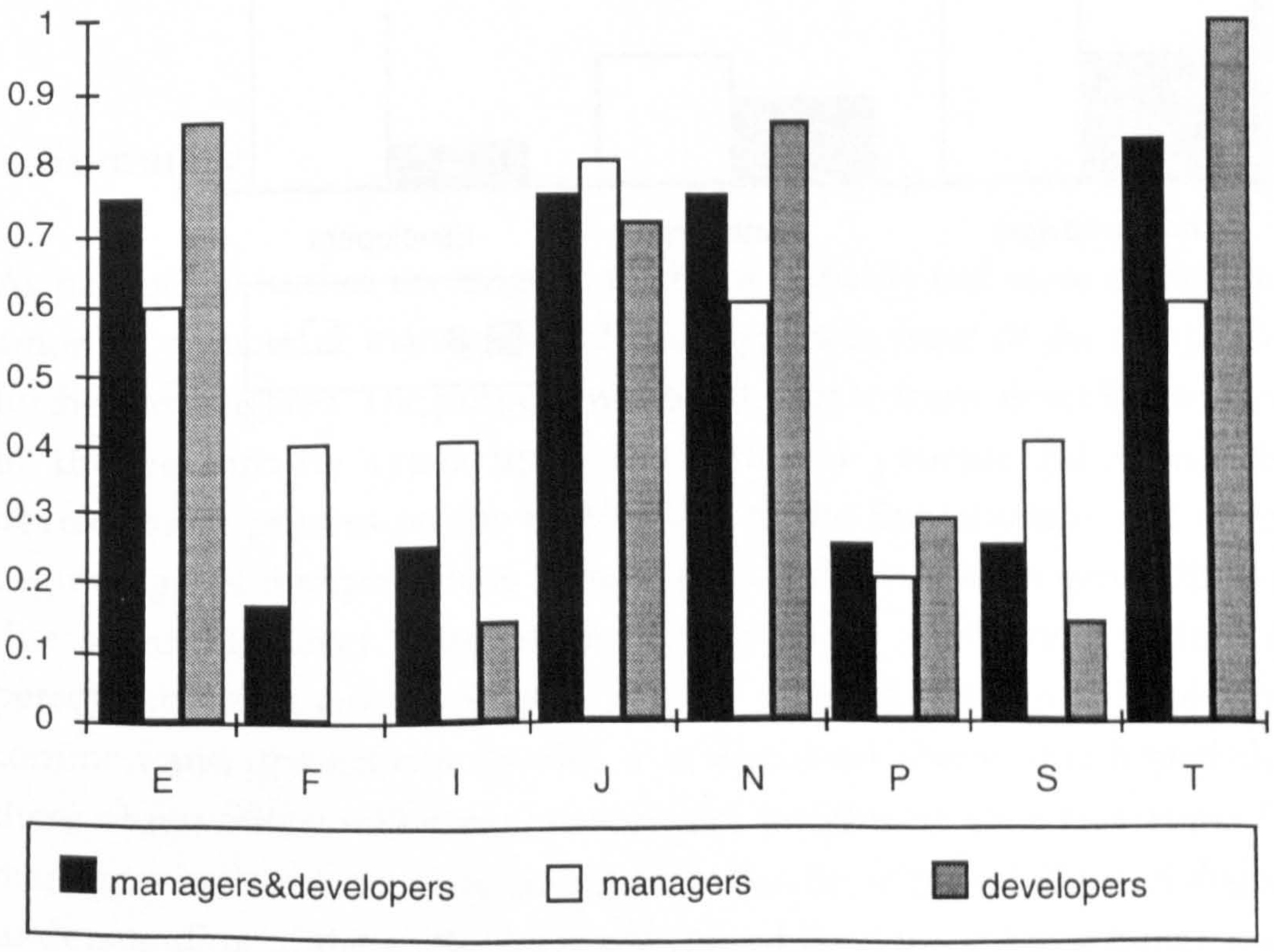


Chart 6 Percentage of Occurrence of Personality Attribute



(6) E, N, T were the dominant attributes among the prototypers. Chart 7 demonstrates that, for prototypers, attributes E, N and T (the grey columns in chart 7) are well above the average (the black columns in chart 7): 100% for T, 90% for E and N; Further comparison with their opposing attributes (see chart 8 - 11), shows clearly that E, N, T are also much higher than counterparts: I, S, F. This indicates that E, N and T, as explained above, might have particular importance for prototypers.

Chart 7 Comparison of Attribute Distribution



(7) Managers appeared to have a higher J occurrence. In parallel, for managers, Chart 7 shows that J is the only attribute above the average (see J of the white column in chart 7, higher than the corresponding black column), and also this is evident in chart 11. This seems to be a natural result of the apparent link between J (judging) and decision making.

Chart 8 Extrovert versus Introvert

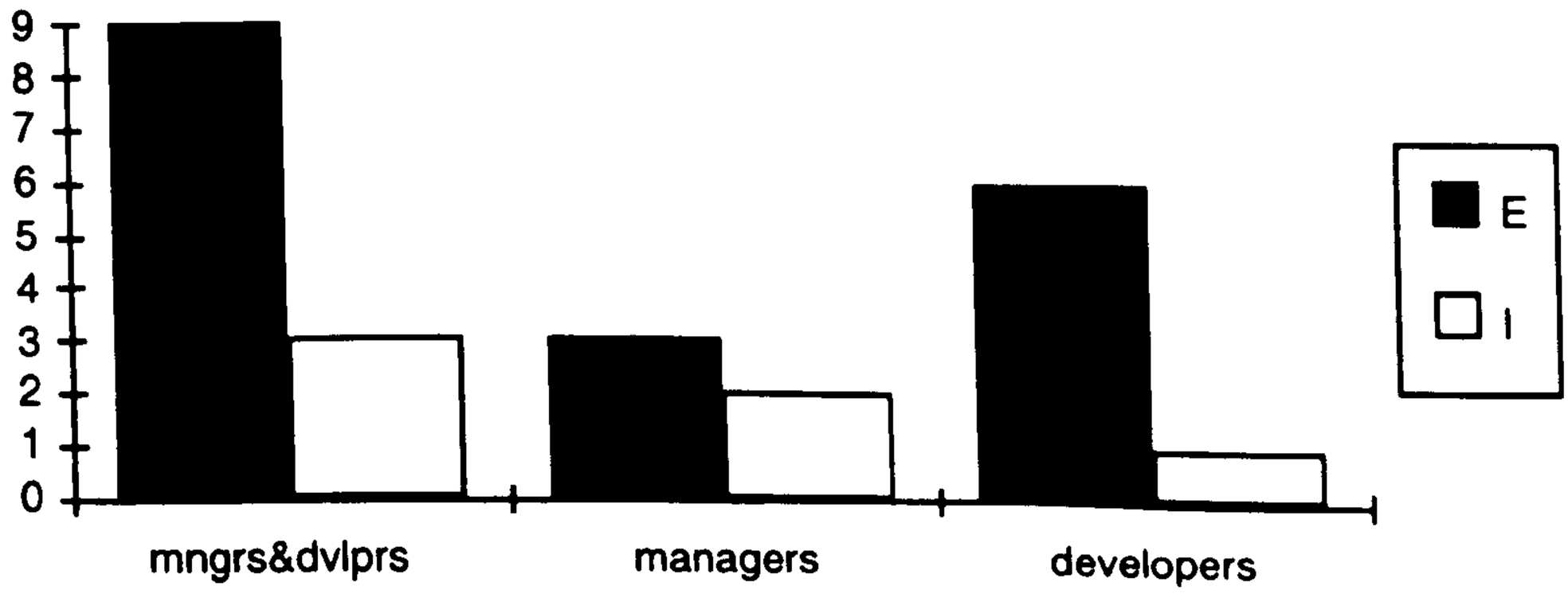


Chart 9 Sensing versus Intuition

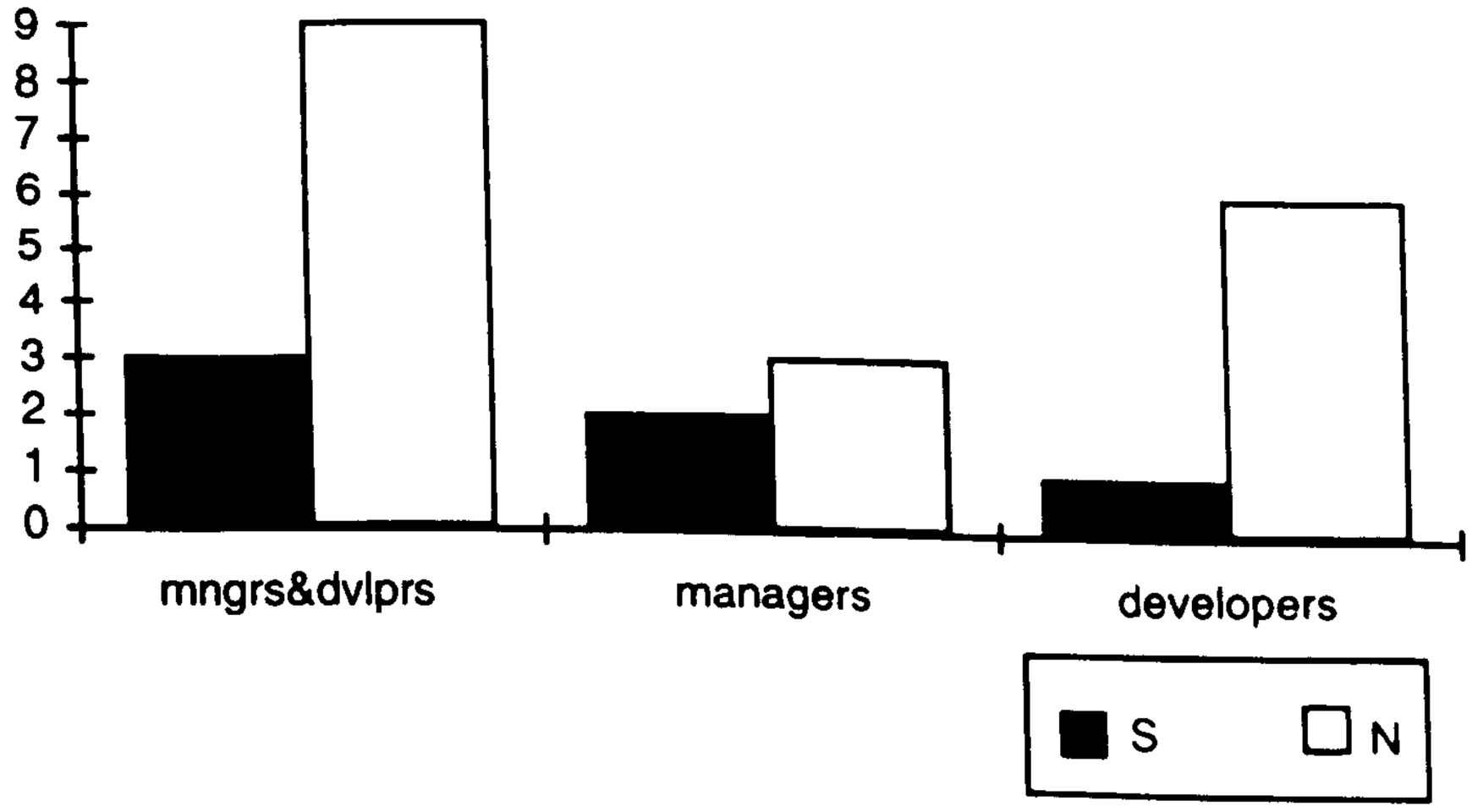


Chart 10 Thinking versus Feeling

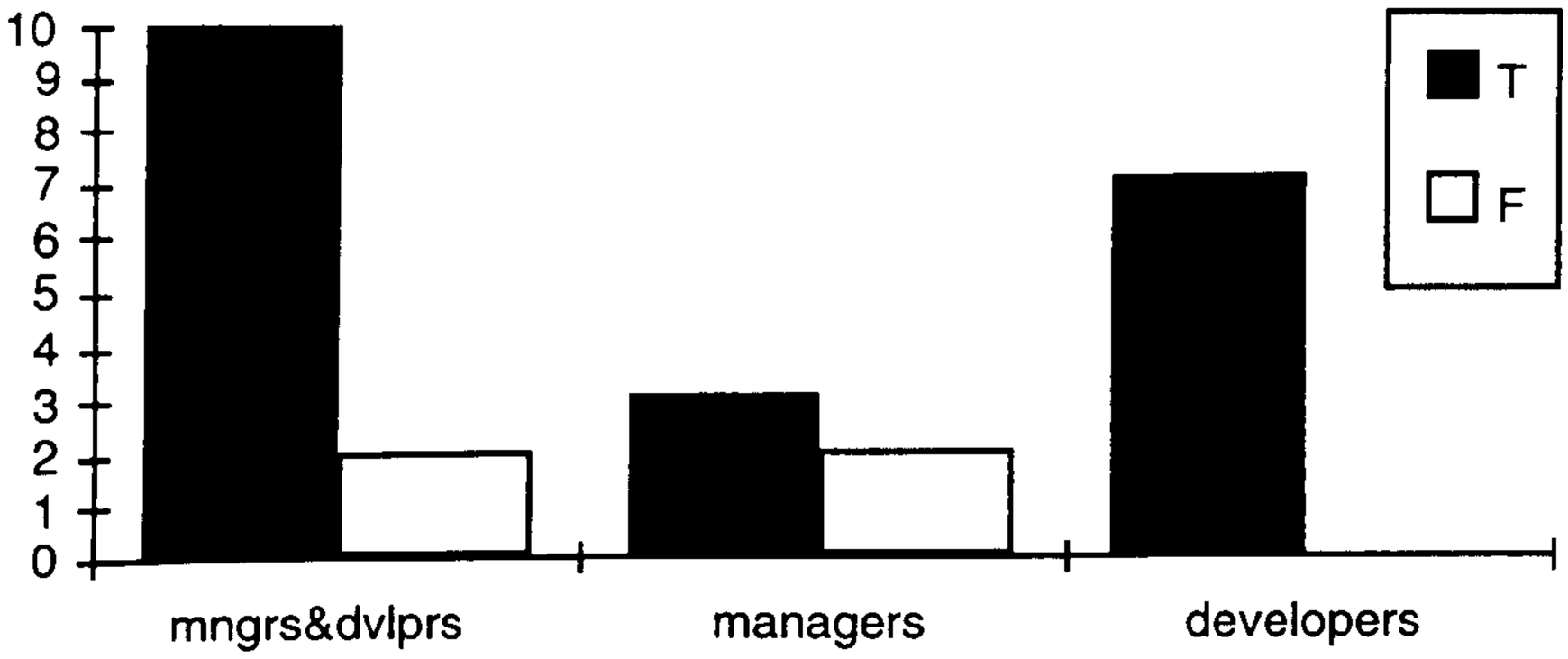
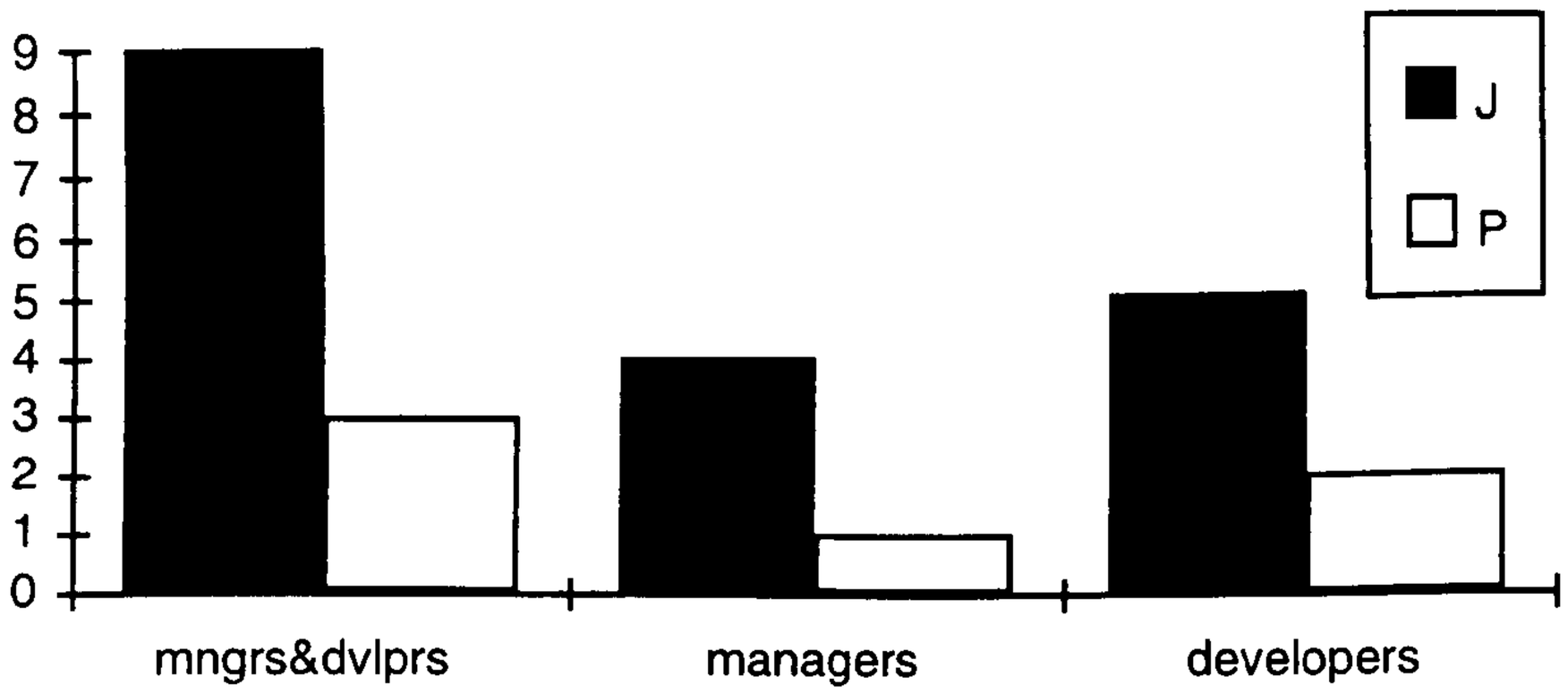


Chart 11 Judging versus Perception



6. Summary

As part of the further investigations, the personality test were carried out among a number of managers and prototypers in most of the companies further investigated. The purpose was to attempt to learn about the impacts of the personality types upon the different process roles and the development process on the whole. Due to the limitations stated at the beginning, the interpretations of the results is difficult to be generalised at this stage. However, there seems clearly to be a different pattern of personality types among different process roles, and each role have some common and distinct characteristics as discussed above. It is hoped that these observations will bring some useful insights or least awareness for managers in their team selection practice. Finally, to have fuller and deeper understanding of the subject, the test should be extended to a larger scale

with more participants and a wider scope including more roles such as customers and end-users.

Appendix F(b) Questions and type explanations of Myers-Briggs Personality Test

Questions and type explanations of Meyers-Briggs Personality Test

a. Questions

Q1: When you go somewhere for the day, would you rather plan what you will do and when, or just go?

Q2: If you were a teacher, would you rather teach fact courses, or courses involving theory?

Q3: Are you usually a 'good mixer' or rather quiet and reserved?

Q4: Do you often let your heart rule your head or your head rule your heart?

Q5: In doing something that many other people do? does it appeal to you more to invent a way of your own, or do it in the accepted way?

Q6: Among your friends, are you full of news about everybody or one of the last to hear what is going on?

Q7: does the idea of making a list of what you should get done over a weekend appeal to you, or leave you positively cold and depressed?

Q8:When you have a special job to do, do you like to organise it carefully before you start, or find out what is necessary as you go along?

Q9: Do you tend to have broad friendships with many different people, or deep friendships with a very few people?

Q10: Do you admire more the people who are conventional enough never to make themselves conspicuous, or to original and individual to care whether they are conspicuous or not?

Q11: Do you prefer to arrange dates, parties, etc., well in advance, or be free to do whatever looks like fun when the time comes?

Q12: Do you usually get along better with realistic people, or imaginative people?

Q13: When you are with a group of people, would you usually rather join in the talk of the group, or talk with one person at a time?

Q14: Is it a higher compliment to be called a person of real feeling, or a consistently reasonable person?

Q15: In reading for pleasure, do you enjoy odd or original ways of saying things, or like writers to say exactly what they mean?

Q16: Do you talk easily to almost anyone for as long as you have to, or find a lot to say only to certain people or under certain conditions?

Q17: Does following a schedule appeal to you, or cramp you?

Q18: When it is settled well in advance that you will do a certain thing at a certain time, do you find it nice to be able to plan accordingly, or a little unpleasant to be tied down?

Q19: Are you more successful at following a carefully worded out plan, or at dealing with the unexpected and seeing quickly what should be done?

Q20: Would you rather be considered a practical person, or an ingenious person?

Q21: In a large group, do you more often introduce others, or get introduced?

Q22: Do you usually value sentiment more that logic or value logic more than sentiment?

Q23: Would you rather have as a friend someone who is always coming up with new ideas, or someone who has both feet on the ground?

Q24: Can the new people you meet tell what you are interested in right away, or only after they really get to know you?

Q25: In your daily work, do you usually plan your work so you won't need to work under pressure, or rather enjoy an emergency that makes you work against time?

Q26: Do you usually show your feelings freely, or keep you feelings to yourself?

Q27: Which word in the pair below appeals to you more: Scheduled or Unplanned?

Q28: Which word in the pair below appeals to you more : Facts or ideas?

Q29: Which word in the pair below appeals to you more: quiet or hearty?

Q30: Which word in the pair below appeals to you more: convincing or touching?

Q31: Which word in the pair below appeals to you more: imaginative or matter-of-fact?

Q32: Which word in the pair below appeals to you more: benefits or blessings?

Q33: Which word in the pair below appeals to you more: peacemaker or judge?

Q34: Which word in the pair below appeals to you more: systematic or spontaneous?

Q35: Which word in the pair below appeals to you more: statement or concept ?

Q36: Which word in the pair below appeals to you more: reserved or talkative?

Q37: Which word in the pair below appeals to you more: analyse or sympathise?

Q38: Which word in the pair below appeals to you more: create or make?

Q39: Which word in the pair below appeals to you more: determined or devoted?

Q40: Which word in the pair below appeals to you more: gentle or firm?

Q41: Which word in the pair below appeals to you more: systematic or casual?

Q42: Which word in the pair below appeals to you more: certainty or theory?

Q43: Which word in the pair below appeals to you more: calm or lively?

Q44: Which word in the pair below appeals to you more: justice or mercy?

Q45: Which word in the pair below appeals to you more: fascinating or sensible?

Q46: Which word in the pair below appeals to you more: firm-minded or warm-hearted?

Q47: Which word in the pair below appeals to you more: feeling or thinking?

Q48: Which word in the pair below appeals to you more: literal or figurative?

Q49: Which word in the pair below appeals to you more: foresight or compassion?

Q50: Which word in the pair below appeals to you more: hard or soft?

b. Type explanations

1) ISTJ
Serious, quiet, earn success by concentration and thoroughness. Practical, orderly, matter-of-fact, logical, realistic, and dependable. See to it that everything is well-organised. Take responsibility. Make up their own minds as to what should be accomplished and work toward it steadily, regardless of protests or distractions.

2) ISTP
Cool onlookers - quiet, reserved, observing and analysing life with detached curiosity and unexpected flashes of original humour. Usually interested in impersonal principles, cause and effect, how and why mechanical things work. Exert themselves no more than they think necessary, because any waste of energy would be inefficient.

3) ISFJ
Quiet, friendly, responsible, and conscientious. Work devotedly to meet their obligations. Lend stability to any project or group. Thorough, painstaking, accurate. May need time to master technical subjects, as their interests are usually not technical. Patient with detail and routine. Loyal, considerate, concerned with how other people feel.

4) ISFP
Retiring, quietly friendly, sensitive, kind, modest about their abilities. Shun disagreements, do not force their opinions or values on others. Usually do not care to lead but are often loyal followers. Often relaxed about getting things done, because they enjoy the present moment and do not want to spoil it by undue haste or exertion.

5) INFJ

Succeed by perseverance, originality and desire to do whatever is needed or wanted. Put their best efforts into their work. Quietly forceful, conscientious, concerned for others. Respected for their firm principles. Likely to be honoured and followed for their clear convictions as to how best to serve the common good.

6) INFP

Full of enthusiasms and loyalties, but seldom talk of these until they know you well. Care about learning, ideas, language, and independent projects of their own. Tend to undertake too much, then somehow get it done. Friendly, but often too absorbed in what they are doing to be sociable. Little concerned with possessions or physical surroundings.

7) INTJ

Usually have original minds and great drive for their own ideas and purposes. In fields that appeal to them, they have a fine power to organise a job and carry it through with or without help. Sceptical, critical, independent, determined, often stubborn. Must learn to yield less important points in order to win the most important.

8) INTP

Quiet, reserved, impersonal. Enjoy especially theoretical or scientific subjects. Logical to the point of hair-splitting. Usually interested mainly in ideas, with little liking for parties or small talk. Tend to have sharply defined interests. Need careers where some strong interest can be used and useful.

9) ESTP

Matter-of-Fact, do not worry or hurry, enjoy whatever comes along. Tend to like mechanical things and sports, with friends on the side. May be a bit blunt or insensitive. Adaptable, tolerant, generally conservative in values. Dislike long explanations. Are best with real things that can be worked, handled, taken apart or put together.

10) ESFP

Outgoing, easygoing, accepting, friendly, enjoy everything and make things more fun for others by their enjoyment. Like sports and making things. Know what's going on and join in eagerly. Find remembering facts easier than mastering theories. Are best in situations that need sound common sense and practical ability with people as well as with things.

11) ESTJ

Practical, realistic, matter-of-fact, with a natural head for business or mechanics. Not interested in subjects they see no use for, but can apply themselves when necessary. Like to organise and run activities. May

make good administrators, especially if they remember to consider others' feelings and points of view.

12) ESFJ
Warm-hearted, talkative, popular, conscientious, born co-operators, active committee members. Need harmony and may be good at creating it. Always doing something nice for someone. Work best with encouragement and praise. Little interest in abstract thinking or technical subjects. Main interest is in things that directly and visibly affect people's lives.

13) ENFP
Warmly enthusiastic, high-spirited, ingenious, imaginative. Able to do almost anything that interests them. Quick with a solution for any difficulty and ready to help anyone with a problem. Often rely on their ability to improvise instead of preparing in advance. Can usually find compelling reasons for whatever they want.

14) ENTP
Quick, ingenious, good at many things. Stimulating company, alert and outspoken. May argue for fun on either side of a question. Resourceful in solving new and challenging problems, but may neglect routine assignments. Apt to turn to one new interest after another. Skilful in finding logical reasons for what they want.

15) ENFJ
Responsive and responsible. Generally feel real concern for what others think or want, and try to handle things with due regard for other person's feelings. Can present a proposal or lead a group discussion with ease and tact. Sociable, popular, sympathetic. Responsive to praise and criticism.

16) ENTJ
Hearty, frank, decisive, leaders in activities. Usually good in anything that requires reasoning and intelligent talk, such as public speaking. Are usually well-informed and enjoy adding to their fund of knowledge. May sometimes be more positive and confident than their experience in an area warrants.