An investigation of methodologies for software development prototyping

Susan M. Jones
Edith Cowan University

1993

An investigation of methodologies for software development prototyping

Susan M. Jones
Edith Cowan University

An Investigation of Methodologies for Software Development Prototyping

by

Susan M. Jones

A dissertation to be submitted in partial fulfilment of the requirements for the degree of

Master of Applied Science
(Computer Studies)

Department of Computer Science
School of Information Technology and Mathematics
Edith Cowan University
Perth, Western Australia

Supervisor: Ah Hung

July 1993

Susan M. Jones

[redacted]

Abstract

The computer industry has a poor record of system development using the traditional life-cycle approach. The main cause of user dissatisfaction is the unacceptably large amount of time between specification and delivery of a system. In addition, users have limited opportunity to influence how the system will look when implemented once development has commenced.

With the advent of 4GLs, system development using a prototyping approach has become a viable option. This has reduced the development time significantly and, together with the use of prototyping, has allowed users to become more involved in the development process.

However, this change in the development process has meant that often the use of an accepted methodology/system life cycle has been ignored or altered. This has resulted in systems where the definition-of-requirements phase was often fast-tracked or omitted totally and the system documentation is insufficient for effective maintenance.

Thus, this approach has not proved to be as successful as expected. However, the opportunities that prototyping offers should not be discarded because of the use of inappropriate software development methodologies, languages or tools.

This study seeks to identify factors that may influence the success or failure of a prototyping project and to assess the importance of any development methodologies being used.

Information was gathered via interviews, questionnaires and, where deemed necessary, the reviewing of development procedures used.

Conclusions have been drawn from data gathered from various organisations in Western Australia that have used prototyping for a number of projects, thus, suggesting a refinement of the development process.

Two main areas appeared to affect the success of a software development project. The first is the lack of flexibility in the methodology used and inappropriateness of the development tools and languages. The second is insufficient requirements analysis.

The results indicate that a methodology is required that provides a good framework, but is flexible enough to handle different types and sizes of project. It should specifically address prototyping and include guidelines

as to how to select the most suitable prototyping approach for each project. It should contain examples of different deliverables and various development cycles appropriate for each type of prototyping. There should be automated tools available to handle documentation and code generation where possible.

The development of a methodology with the above characteristics is required if the advantages of prototyping are to be maximised in the future.

Declaration

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree in any institution of higher education; and that, to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where due acknowledgment is made.

Signature:

Date: 15/12/93

Acknowledgments

I would like to thank all those people who have helped and encouraged me throughout this project, particularly my supervisor Ah Hung, colleagues Bob Cross and Ron Hartley for their advice and encouragement and Bill Laidman who meticulously checked this document to ensure its completeness and accuracy. I would also like to thank Tony Fetherston for his advice on the layout and structure of the thesis, particularly regarding the statistics.

Special thanks to Steve and Paul who have tolerated countless meals of pasta, due to it being quick to prepare, and endured me working evenings and weekends whilst writing up my results.

TABLE OF CONTENTS

List of Tables

1. Introduction

To determine a method of software development that will be consistently successful is the goal of most software developers. There are so many different factors affecting the outcome of any project that this seems an impossible goal.

However, by studying both successful and not so successful projects and analysing the mix of methodology, tools, language and project type in the light of the developers' experience and training, the critical factors should become more identifiable.

Prototyping has become more viable as new development environments, tools and languages become available. Prototyping, by its very nature, will usually result in a working system, which is an improvement on previous methods of software development. However, the speed, efficiency and cost with which it happens depend upon the above-mentioned mix of factors.

This study aims to identify those factors which together provide the right mix for a successful development project using prototyping.

2. The Problem

2.1 Background to the Study

The computer industry has a poor record of system development using the traditional life-cycle approach.

One survey of software projects (Gladden, 1982) states that 25% of systems were never delivered and 47% were delivered but never used. Thus, only 28% of systems were actually used.

There are various reasons for the lack of success in system development. One of the main causes of user dissatisfaction with the systems delivered is the amount of time between the analysis and design of the system and the implementation. According to Martin and Carey (1991) "traditional approaches [to software development] not only seem to deliver late systems that do not please the user, but are also costly". In addition, systems developed this way may be "difficult to learn and use". The backlog of projects awaiting development also increases the amount of time the users have to wait for their system.

Current systems can be extremely complex and require a large development team. This increases the number of lines of communication within the project team and the users, making the project more difficult to manage (Brooks, 1982).

Users who have no previous experience of computer systems find it extremely difficult to visualise how their system will look and act, when depicted using traditional techniques (data flow diagrams, data dictionary and functional specifications). They rarely have an accurate picture of their informational needs (Martin and Carey, 1991). Additionally, by the time of implementation the users' requirements have almost always altered. This may be due to external constraints, but also to the change in the users' perceptions of the computer's capabilities once they have some experience of using a computer system.

Users have always wanted to see how their system will work at an early stage in development in order to better understand the functionality of a computerised system. The increase in the use of 4GLs has enabled prototyping to become a viable method of development and this allows users to become more involved in the development process.

As project leader and system designer for a 4GL and prototyping project, the author felt that a much better job could have been done had the circumstances and facilities been different and if an appropriate methodology had been available. More recently the author was a supervisor for a CEED project that required the student to produce a generally acceptable methodology for use in a prototyping environment. No formal methodology had been used in the development of several successful projects using prototyping with a 4GL. The systems were developed by a user department and the Computing department were insisting that all future systems be developed using a formalised methodology. The user department considered the methodology used by the Computing department to be inappropriate for prototyping and decided to develop one that would reflect the stages and processes that had been refined during the development of several systems.

It appears from the literature that many different types of approach are used, ranging from the complete system life-cycle (Carey, 1990 and Rowen, 1990) to the ad hoc, no methodology approach.

Some authors present prototyping as a methodology in itself (Palvia and Nosek, 1990; Wojtkowski and Wojtkowski, 1988).

The way that prototypes are used varies widely. Some authors maintain that the prototype should never be used as the final system as it is only for defining what the final system should look like. Others use incremental prototypes until the final version is implemented as the production system.

With such a wide range of approaches to prototyping, it appeared that a study on the most successful and effective methods of system development using prototyping could be very informative and useful.

2.2 Significance of the Study

The poor record of system development using a traditional approach has been well-documented (Martin, 1985; Brooks, 1982; Gregory and Wojtkowski, 1990). A system may take two years to develop, by which time the requirements of the users may well have changed and, as the users have had no opportunity to use the system during development, the system may not meet the users' original expectations.

The increase in the use of 4GLs has enabled prototyping to become a practical method of development. This has reduced the development time significantly and, together with the use of prototyping, has allowed users to become more involved in the development process.

However, this change in the development process has meant that not only has the time factor been reduced, but the use of an accepted methodology/system life cycle has been ignored or altered. This has resulted in systems where the definition-of-requirements phase was often fast-tracked or omitted totally and the system documentation is insufficient for effective maintenance.

The opportunities that prototyping offers should not be discarded because of the use of inappropriate software development methodologies.

This study seeks to identify factors that may influence the success or failure of a prototyping project.

2.3 Theoretical Framework

2.3.1 Identification of variables impacting on the research questions and their inter-relationships

There are a number of factors that will affect the outcome of the research questions. The type of information that will need to be gathered during the fact-finding process will be:

- The method of prototyping used.
- The methodology used.
- The strengths and weaknesses of the methodology and the development tools used.
- The development language and/or tools used.
- The suitability of the development language for the adopted methodology.
- Was a thorough requirements analysis carried out prior to the commencement of prototyping?
- The type, size and complexity of each project.

- The training and experience levels of the developers and the users involved.

- The level of user involvement in each project.

- Was the development successful? If not, why not?

- What criteria were used to judge the level of success?

- Was the system delivered on time and within budget?

- What refinements were made to the development process used?

- What improvements could be made to the methodology used?

2.3.2 Identification of theoretical and philosophical assumptions underpinning the study

Certain assumptions have had to be made concerning the data gathered:

- The information supplied is true and has not been doctored for political motives. This could happen if management do not wish any project failures to be widely known. This can be avoided by reassuring participants that the published data and results will not associate organisations with particular data.

- It must be assumed that each project was correctly costed and scheduled. It may be beneficial to discover what methods of estimation were used.

- Although the author will try not to view the data or results with any bias, it should be noted that the author has spent many years in a system development role in industry and thus is not approaching the study in a purely theoretical manner.

2.4 Statement of the Problem to be Investigated

This study sets out to determine the significance of the system development methodology used, by reviewing the development process and resulting systems that have been developed when using a prototyping approach. The outcome of this study should be of benefit to future system developers by providing them with a better approach to prototyping.

Because three significant methods of prototyping exist, different solutions may be found to be appropriate for each method.

2.5 Statement of Research Questions

Are the current life-cycle methodologies appropriate for system development using a prototyping approach?

How does system development using prototyping differ from traditional system development?

Does the system development life-cycle need to be modified or is a totally new approach required?

2.6 Delimitations and Limitations of the Study

The study will rely on the willingness of organisations to allow their experiences to be included in the study.

There is a need for honesty from contributors to ensure the integrity of the resulting conclusions. Therefore, all participants must be guaranteed anonymity and this should also help to preclude any political motives.

There has not been a large number of software development projects undertaken in WA using a prototyping approach.

2.7 Definition of Terms

Within the computing industry there is a variety of terms used to describe different functions and processes. This is reflected in the literature. For the sake of clarity the terms used in this study are defined below:

Prototyping

iterative (or evolutionary) -

the final iteration becomes the production system;

piloting (or rapid) -

used to determine feasibility and test alternative solutions;

modelling (or throw-away) -

to determine user requirements and/or screen and report requirements and processes to be performed on the data; the final model is discarded and rewritten, generally using a different method or language.

4GL - there is no precise definition of the term 'fourth generation language', although James Martin (Martin 1982) is credited as being the first to use it. Now it is generally used to describe a complete environment of development tools, language, database and screen painter. The language not usually being a third-generation language, but more likely some sort of 'specification language' (Grindley 1987).

3. Review of Relevant Literature

3.1 General Literature

Due to the speed of change in the current technology and software development methods, the literature search has concentrated on articles and books published since 1987. Some literature prior to this date has been included when it is deemed to be of particular significance.

There is a number of reasonably recent papers and a few books dealing with software development using Prototyping. Surveys carried out have generally been more concerned with the type and size of project, its suitability for prototyping and its degree of success, rather than with the type of methodology used to achieve this. However, a small number of articles has been identified that describe surveys of prototyping methodologies.

There are two main trends in prototyping methodologies: the first is that prototyping is used in conjunction with an established methodology, the second is that prototyping is the methodology.

The books and articles found fall into a number of categories:

- guidelines on how to use prototyping
- how to use prototyping within the structured system design cycle
- one specific project and the methodology and tools used
- prototyping as a methodology for requirements analysis
- descriptions of software development tools and 4GLs suitable for prototyping
- surveys and descriptions of different methodologies (not specific to prototyping).

3.2 Specific Studies Similar to the Current Study

Doke (1990) uses his survey to attempt to answer the following questions:
- which specific prototyping methodologies exist?
- to what extent are they being used?
- how important are they to system development projects?

He identifies four distinct classes of prototyping methodology (Illustrative, Simulated, Functional and Evolutionary), three of which produce disposable systems and one where the prototype evolves into the final system. The prototyping methods vary from the simple building of sample screens and reports, to the "iterative heuristic development process, in which the user guides system design by reviewing and interacting with models of the proposed system and making suggestions for its modification and improvement". This continues until the users consider the system acceptable.

He surveys relatively large organisations, finding that those with fewer software development staff were less likely to prototype. His research raised additional important questions:
- When should the various methodologies be used?

- What is the impact of the methodologies on the traditional life cycle?
- Is it appropriate to employ multiple methodologies concurrently?
- As tools such as 4GLs become more popular and operationally efficient, what is the expected impact on the prototyping methodologies?

It is hoped that the current study will not only answer the research questions stated in section 2.5, but will also go some way towards answering these questions that Doke has raised. The research questions for both this study and Doke's study are very similar, but the current study aims to gather information on aspects of the development process other than the methodology in order to gauge the importance of the methodology in the outcome of the project.

A second study that is similar (Necco, Tsai and Gordon, 1989) considers prototyping to be of significant benefit during the requirements analysis phase. However, the results cause the researchers to note that "the prototyping approach is not a substitute for the Systems Development Life Cycle approach". The survey shows that prototyping is used to develop all different types of information system, although some organisations use prototyping for only certain types of system. In their conclusions Necco et al. state that the prototyping approach is being used by some organisations to develop systems that are unsuited to this type of development. Users were more likely to be satisfied and the resultant systems required less maintenance. However, it was concluded that developers should be more selective in the projects that they choose to prototype; they should use it in conjunction with existing methods as appropriate to the project; and that a "formal strategy for its use should be prepared".

A third study found to be similar was conducted by Martin and Carey (1991). They define prototyping as "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". They describe the problems inherent in software development using traditional methods and propose that "the goals of prototyping are development of information systems that are functionally correct, delivered quickly, less expensive and easy to learn and use". They identify two types of prototyping: iterative and throwaway.

A mail survey was conducted and the results discussed in this paper. The paper examines the use of prototyping for transaction-processing systems and their conversion from prototype to operational system.

Generally prototyping has been used for small decision-support systems, rather than large, stable, transaction-processing systems. Martin and Carey suggest that a sensible approach would be to use prototyping to develop the system and then tune the system until an acceptable level of performance is reached.

They identified two key research questions as follows:
"Are Transaction Processing Systems being developed by prototyping methodologies?
What strategies exist for conversion of prototype models to operational Transaction Processing Systems?"

Deciding whether to develop a throwaway or an iterative model is the other major aim of the paper. They assert that "one of the primary goals of prototyping is user communication". This is of particular importance during the analysis phase and thus, there is no reason to continue with the prototype after this stage. However, few developers are willing to discard a working model without sufficient justification. This is in spite of the differing requirements of a prototype and an operational system. The ideal development language would be a 4GL which is quick and easy to use, but may not have all the necessary functionality of a 3GL, may be less likely to be as self-documenting as a 3GL and may not be as suitable to top-down structures required by operational systems. Other differing requirements concern documentation, computer architecture, access control and development of procedures. To convert a prototype to an operational system all these matters have to be considered. To ease this conversion Martin and Carey suggest the following approach should be taken: the prototype should be programmed in the language designated for the operational system; the model should be fully documented as it is built; the development should use the same hardware as the operational system; the prototype should be considered iterative, even though it may be thrown away eventually.

A survey was conducted with the intention of supporting or refuting these ideas. In spite of a very low response rate of only 7.1% Martin and Carey considered the results worthy of analysis. Only 56% of the respondents were prototyping and only 15% of those actually threw away the prototype completely. 42.5% used the prototype as the operational system and 42.5% discarded the prototype for design purposes, but used it for such things as "demonstration, reuse of code, training and system documentation". 70% programmed the prototype in the same language as the operational system, of which 55% used a 3GL and 15% used a 4GL. The other 30% prototyped using a 4GL and then built the operational system in a 3GL. The most common prototyping language was COBOL, as this was the most used language for the operational systems. Another interesting observation was that the development times for building transaction-processing systems was very similar to the development times for decision-support systems. Martin and Carey considered these results to be atypical of what is generally believed about prototyping and thus, felt that academics and computing professionals should be made aware of them.

Palvia and Nosek (1990) conducted a survey in order to evaluate two types of methodology: the System Development Life Cycle methodologies and the Prototyping methodologies, based on actual projects in business and industry. Their objectives were to assess the methodologies on their appropriateness at each phase of development, for different system types, for structured and unstructured problems and to determine the "perceived value of the attributes associated with the methodologies". The analysis of the data collected produced the rather surprising result that "more practitioners found prototyping useful for design than analysis (64.3% versus 50%). Less surprising was that the system development life cycle approach was found to be more suitable for structured problems, whilst prototyping is more suitable for unstructured problems.

Prototyping was found to be a little less costly, easier to use, much easier to learn, better for communicating with the user and with other computing professionals, produced a more flexible design and made early identification of problems easier. However, project control was not as good, the systems were slightly less maintainable producing higher ongoing costs and the overall quality of the documentation was not quite as good as for the system life cycle methodologies.

3.3 Other Literature of Significance to this Study

There is a number of papers and books that expound prototyping as a methodology, rather than an approach to be used with a traditional system development methodology. These are of interest as they give the steps that are followed when using this approach. There is literature that describes the use of prototyping within a traditional life cycle and some that discusses other issues relevant to prototyping.

3.3.1 Prototyping as a methodology

A discussion paper (An Accelerated Methodology, 1990) outlines the advantages of prototyping and lists guidelines of when prototyping should be used and when it should not be used, as proposed by Milton Jenkins (1990). Jenkins states that a methodology for prototyping is essential and that it is unreasonable to use the same development methods for all projects.

Although he recognises that there are three different types of prototyping, his view is that prototyping should also produce an operational system, not just a model.

His assertion that prototyping produces systems in 5 to 10 percent of the time and at 10 to 15 percent of the cost of traditionally developed systems is not supported by any data or references to studies, although the author of the article states that Jenkins has 190+ prototyping case studies from 40 different organisations. These claims are not reflected in the Necco et al. (1989) survey where about two-thirds of respondents reported that their systems were developed in less time and only about half reported that the system development was less expensive.

One requirement that Jenkins considers critical to the success of the project is to use real data, not test data, when prototyping. He also emphasises that large systems should be broken down into "manageable chunks", otherwise they are not suitable for prototyping.

Jenkins lists factors that influence the use of systems and discusses the risk issues that arise when prototyping. He also outlines the type of costs and benefits involved. One of the costs listed is reorganisation due to prototyping "flattening the organisation and eliminating the need for middle management in IS". This observation has not been encountered in other literature.

The advantages of prototyping are also discussed by Owen (Owen, 1989), but he maintains that software development should be completely "disconnected" from "traditional (and failed) development methodologies". Owen lists the advantages that he considers prototyping provides: shortened development cycle, earlier implementation, simpler project management, lower development costs, improved user developer communications, improved quality assurance, lower enhancement and maintenance costs, concentration of business functions and improved user satisfaction. The first four advantages he attributes to the shorter project cycle. However, he claims that the development cycle will be shortened by 6 to 12 months. This seems to be a difficult claim to make without qualification, as this would be dependent on the size of the system to be developed. It would appear unreasonable to expect a very small system development to be reduced by as much as six months. Improved quality assurance will be due to the use of "advanced development tools" which will "produce much of the actual code". This is not supported by the current study or by other surveys in the literature, as not all developers are using the latest in advanced development environments. An industry survey concerning the conversion of prototypes to operational systems (Martin and Carey, 1989), found that prototyping in a third generation language was common. Owen next lists the perceived disadvantages: these include machine inefficiency, different skills required, lack of error trapping capability and inadequate functionality of the system.

Owen states that prototyping delivers the system in segments and that this is one of its valuable characteristics, whereas Jenkins (1990) states that the system **must** be split into segments in order for the prototyping process to be successful. Finally, a major advantage of prototyping is the much improved communication between user and developer, which helps their understanding of one another's problems, as seen by Jenkins, Owen, Martin and Carey .

Prototyping as a methodology is described by Wojtkowski and Wojtkowski (1988) as being used for the "system requirements determination". They acknowledge the view of practitioners of prototyping that it should not replace adequate analysis and design. However, they attribute failure in prototyping to insufficiently trained users with unrealistic expectations, prototyping inappropriate projects, using the wrong type of prototyping, not having the "proper technical environment" and ineffective project management. They propose solutions to these problems which include the "development and documentation of a prototype life cycle" that is appropriate for a particular system.

They expand these concepts (1990) to include such topics as: responsibilities of the prototyping participants, different life cycle models, selecting projects suitable for prototyping, prototyping tools and management issues. They discuss possible "pitfalls" of prototyping and also give some success stories.

3.3.2 Prototyping within a life cycle methodology

Carey (1990) explores the different types and uses of prototyping. He observes that prototyping has been used as a methodology, whereas he thinks that it should be used within a system development methodology. He describes the advantages and disadvantages of using prototyping and suggests a methodology into which prototyping can be incorporated. The factors affecting which types of systems are suitable for prototyping and what type of prototyping should be used are discussed. The importance of the human factors in a system are stressed and guidelines given as to what these factors are and how they should be considered during the system design phase. Two case studies are provided, one a successful prototyping project and one a failure. Carey is illustrating that success depends on the suitability of the system for prototyping and selection of the right tools. For example, a system where performance is important may have response times that are unacceptable if a 4GL is used to develop the operational version. A better approach would be to develop a model with the 4GL and then rebuild the final version using a 3GL.

A computer aided prototyping methodology which uses modified data flow diagrams and a Prototyping System Description Language outlines the advantages of this approach to system development (Krista and Rozman, 1989). The strategy of this approach to prototyping is based on "the recognition and understanding of the requirements of the system" and the "gradual evaluation of the system which is defined by a model prototype". They stress that decomposition of the problem into workable modules using a top-down problem-oriented approach is a key factor for increased productivity. They treat prototyping as a process of modelling different aspects of a system. This methodology includes detailed analysis using data flow diagrams and uses the model as a documentation and communication tool for verifying the requirements.

Rowen (1990) also believes that prototyping should be used within the framework of a formal life-cycle methodology. The importance of user involvement is stressed. The model that is built is used to promote user discussion and thus to clarify the system requirements, which Rowen suggests are incomplete, inconsistent and ambiguous when first received. He states that the prototyping approach is attempting to "expand the requirements and explore many alternatives before narrowing and freezing the necessary components". The difference between prototyping and traditional development is the means of developing the system, not the end result, which should always be a working system that satisfies the user requirements. Both the 'throw-it-away' and 'incremental' methods of prototyping meet the life cycle's need for early user feedback, whilst maintaining a controlled development structure. He provides a generalised table of contents for a requirements document to aid developers in eliciting the correct type of information from users. The requirements documentation should evolve over the life cycle.

Using a traditional development methodology this would not be viable, because changes in requirements are difficult to incorporate once the system design is complete. However, when there is an ongoing prototype of the system, changes can be incorporated relatively easily if a good 4GL environment is being used.

3.3.3 Using prototyping

Tate (1990) describes the different types of prototypes, the economics of prototyping, some examples of their practical application and briefly looks at the life-cycle issues. He gives the primary reasons for prototyping as "to buy knowledge and thus, reduce uncertainty" and to improve the chance of the development being successful. He discusses the economics of prototyping from two viewpoints, one being the risk factors and consequences associated with the project failing and the second being the possibility of improved productivity. He also uses an approach by Davis, Bersoff and Comer (1988) to define productivity as "functionality delivered per unit cost". Tate considers this approach to be more conventional than risk management, but qualifies this by adding that both are valuable and should be used. He continues by discussing different methods of prototyping. Docker (1989) is quoted by Tate as claiming that "requirements that are not rigorously specified cannot be validated" and adds that Davis (1988) proposes that a formal technique for specifying requirements should be used "when you cannot afford to have the requirement misunderstood".

Tate lists some prototyping problems, including boundary definition, the question of whether to use the evolved prototype as the operational version and system performance. When fast response times are essential, as in real-time systems, iterative prototyping may result in poor response times.

Tate describes various life cycles proposed by a number of other authors and suggests that these should all be considered as they are complementary to one another, rather than mutually exclusive. He concludes with a brief discussion of the future of prototyping.

Lea and Chung (1990) go further and propose an approach using structured analysis that results in an executable prototype. They use a standard set of deliverables from the analysis phase of development, i.e. a set of data flow diagrams, a set of mini-specs and a data dictionary. They describe reasons why an executable system cannot be built directly from these deliverables and explain how they have overcome this in their method. They have devised a specification mechanism which has two classes: transactions and objects. This is outlined with examples and followed by the prototyping procedure that they have defined for use with this specification method. The interpreter for the specification language, which was written in C and Prolog, consists of a specification preprocessor and a running environment. This method of development does not take into account the user interface, such as screen or report design, but is interested in verifying the functional requirements of the user.

Martin (1988) outlines his Prototyping Software Development Cycle and maintains that the requirements specification drives the prototyping phase. The main object of the prototyping is to clarify the requirements, but Martin is rather ambiguous as to whether the final prototype is implemented or used as a model for building an operational system.

3.3.4 Other issues relevant to prototyping

Budde and Züllighoven (1990) look at the way prototyping has developed, identifying trends and commenting on research and development that shows promise for the future in this area. They describe the different forms of prototyping and construct definitions for these. They examine the trends that have developed with the emergence of 4GLs and application system generators (such as dBaseIII), logic programming languages (such as Prolog), hypertext systems and object-oriented design. A discussion concerning the current popularity of object-oriented modelling for prototyping is also given.

Connell and Shafer (1989) stress that good project management is essential to avoid the prototype being caught in an endless loop of "demonstration and revision". They cover many aspects of structured rapid prototyping, including managing the process, incorporating formal specification methodologies, selection of prototyping method, suitable applications and case studies.

They suggest that few modifications need to be made to the traditional life cycle milestones, but that they will not occur at the same time as they would in a traditional life cycle development. They include an extra phase for preliminary requirements analysis prior to commencing the prototype. The final requirements specification to be completed once the user has approved the functionality of the working prototype. There are other phases that are similar to the traditional life cycle, but their names have been modified to suit the prototyping process. The changes required to the deliverables are discussed and each of the proposed life cycle deliverables is described. There are less deliverables than would be normal for a traditional life cycle development, but the same information is generally still available in a different form. They stress the need to emphasise the Requirements Analysis phase and that this is unlikely to be reduced in time, but will actually be longer than in traditional development. This is due to the need to produce a preliminary requirements analysis in order to commence building the prototype, then to build the prototype and whilst developing and modifying the prototype to produce a detailed requirements analysis. However, having improved the requirements analysis function, the rest of the development should be much faster to develop, debug and test.

They continue by describing how to build, tune, implement and maintain a rapid prototype system. They address the issues of management, causes of failure, future trends and some additional topics concerning data modelling, normalisation, information centres and tools. Case studies of different types of prototyping projects are given and advice on how to make prototyping work for your organisation.

There are several articles covering the review, evaluation and selection techniques for system development methodologies. Modha, Gwinnett and Bruce (1990) review a number of different methodology selection techniques in an attempt to determine the selection criteria that should be used. Although prototyping is not covered specifically, the issues discussed in this paper would be of interest when considering a methodology for prototyping. Fitzgerald, Stokes and Wood (1985) provide a framework for evaluating methodologies. The methodologies are not being assessed for prototyping but the framework and guidelines proposed would help a developer select a methodology.

Other literature concerning prototyping tools (West, 1986) and 4GLs (Crinnion, 1989), (Lehman and Wetherbe, 1989) and (Gryczan and Kautz, 1990) although not directly relevant to this study would be of interest to anyone considering using a prototyping approach to system development.

3.4 Methodologies that Address Prototyping

One methodology that was cited in the questionnaires is designed specifically for prototyping. It contains guidelines for iterative, piloting and modelling approaches. The differences between the required phases for a traditional approach and a prototyping approach are outlined. The methodology provides fourth generation development tools and a relational database management system. In spite of this methodology being designed for prototyping the respondent who used it qualified it with the phrase "sort of", which implies that it did not provide all that was required.

Another respondent used a CASE tool that was effectively a methodology and an application generator in one package. This was found to be excellent for prototyping and had been used for several projects in addition to the one described in the questionnaire.

Recently a student was required to produce a generally acceptable methodology for a client, that would reflect the stages and processes that had been refined during the development of several successful projects using a 4GL and a prototyping approach.

All of these methodologies followed some of the phases of the system life cycle, but did not have the detailed analysis and design phases. A requirements analysis was included but this was not as detailed as it needs to be for a traditional approach. However, this does not preclude a very detailed requirements analysis being done if it is warranted because of the complexity of the system or other constraints.

4. Research Design

4.1 Design of the Study

In order to undertake this research the following steps were taken:

4.1.1 As many organisations as possible, in Western Australia, that have used a prototyping approach for software development were identified.

These were preferably organisations where several projects have been developed, so that the developers have had the opportunity to refine the process and establish standards and guidelines within their organisation.

4.1.2 The initial contact with each organisation was by telephone or personal contact, to enquire as to their suitability, interest and willingness to participate in the study. If they had experience relevant to the study and were interested in participating, they were asked to provide information about relevant development practices.

4.1.3 Questionnaires were sent out to the most appropriate persons for distribution to the specific developers and users of prototyping.

4.1.4 The information gained from the initial contact and the questionnaires was evaluated as to whether follow-up interviews were necessary with any particular participant.

4.1.5 Once it was apparent that no more questionnaires were going to be returned the data collected were analysed.

4.1.6 The data were collated in order to look for patterns or an indication of factors that affect the success or failure of a prototyping project.

4.1.7 These factors were considered in relation to any software development methodology that was used with a view to answering the research questions.

4.2 Research Sample

The research population used was taken mainly from computing departments and computing companies in Western Australia. However, it also includes some user departments that have developed more than one project using the prototyping approach. This covered large and small projects from both the public and the private sector.

As there is no user group specifically aimed at prototyping in WA, access to contacts was by personal recommendation or by direct telephone contact with MIS management. The personal recommendations came mainly from the author's existing industry contacts, plus those suggested by other academics who have an interest in the field of prototyping and 4GLs.

Four telephone calls were made to companies who did not use prototyping at all. Four more used prototyping for small parts of systems, but not sufficient to be included in this study. Two others had tried prototyping for one project, but had so far not used it again and, therefore, would be unable to comment on how they had altered their methods of development in the light of previous experiences with prototyping.

In total 38 companies were approached, of which 28 had used a prototyping approach sufficiently to be included in the study. 28 questionnaires were sent out of which 19 were returned. Thus, 73.7% of companies were prototyping, whereas in Doke's survey (1990) it was 61% and in the Necco et al. (1989) survey it was only 38% prototyping. This would seem reasonable considering the increase in availability of better development environments over the past few years. The return rate was 67%, as compared with Doke's 19%. The difference here was probably due to the initial number of questionnaires sent out by Doke being much larger; he did not talk to each participant prior to sending out the questionnaire and he did not follow up non-returned questionnaires; all of which occurred for this study.

4.3 Description of Instruments Used

4.3.1 The collection of data has been mainly by way of questionnaires, but in some cases, follow-up interviews were conducted also.

Follow-up interviews took place where deemed appropriate, with questions dependent upon the information gathered from the initial contact, the questionnaire and the type of project development taking place. Interviews were carried out for 12 of the projects, but as some respondents submitted more than one questionnaire this involved only 9 people.

In one case, comprehensive discussions took place with a member of a particular company. Unfortunately, this person had left the company by the time the questionnaires were sent out and they were not returned by the remaining employees. However, as this company had used prototyping extensively a description of the original discussions will be included in section 5.3.

4.3.2 Construction of Questionnaire

The questionnaire was pretested for three different prototyping projects. Minor changes were made to wording to remove ambiguity and to offer respondents more space to enter their own comments.

A personalised, covering letter was sent out with each questionnaire explaining the purpose of the study and the type of projects that should be included (Appendix 8.2) and reply-paid envelopes were enclosed with the questionnaires. The confidentiality of the data was stated, but respondents were asked to include their name and address if they wished for a copy of the results of the study.

The questionnaire was designed to lead the respondent through the questions in a logical sequence with the simplest questions at the beginning.

Questions 1 - 6, 10 - 12, 14, 16, 18 and 20 required non-judgemental or quantitative answers that should have been easy for the respondent to complete. Questions 7 and 8 required a 'yes' or 'no' answer and were open-ended only if the answer was 'no'. Question 13 also required a 'yes' or 'no' answer, but was open-ended only if the answer was 'yes'.

Questions 9, 15 and 17 gave a selection of options to rank, but were open-ended allowing the respondents to add any options that they felt were necessary. Questions 19 and 21 were open-ended, requiring judgements to be made and opinions stated.

Question 1 asked for the role of the respondent in the project. By offering three options (Project manager, Project Leader and Other) the type of person who should be capable of knowing the answers to all the questions is implied.

Each question was as concise, clear and unbiased as possible and each addressed one topic only. A copy of the questionnaire is given in Appendix 8.1.

4.4 Data Collection

4.4.1 Collection Method

The main method of obtaining data was from questionnaires and interviews.

The questionnaires were based on the variables impacting the research questions, as described in section 2.3.1. A copy of the questionnaire can be found in appendix 8.1.

Subsequent interviews were based on the initial information gathered and were designed to clarify any ambiguities or to provide further detail.

The information elicited by the open-ended questions is discussed in section 5.3. Few of the questions produced data that displayed obvious quantifiable patterns. However, some common traits are observable and these are described in section 5.3 and any implications discussed in section 6.2.

Several of the questions requested that the respondents should add their own criteria to the questionnaire and all these additional criteria will be listed and commented on for each question.

Criteria that were never referenced are also listed and their lack of relevance to the respondents discussed. They need to be discussed specifically as the author had considered them relevant and it is important to determine why the respondents did not rate them as such.

The relationships between different criteria/factors and the resulting outcome will be looked at for each questionnaire and any obvious trends documented.

4.4.2 The Questionnaire

The questionnaire consisted of 21 questions. The relevance of each question to the study is described below.

What was your role in this project?

The perspective of the respondent may differ between the project manager, the project leader, a developer and a user.

Type of Project?

Some types of project are far more complex than others. For example, a very large stock control project may be much simpler than a small payroll system. This could have a bearing on the time factors affecting the development. Projects that can be broken down into manageable sections are more suited to prototyping (Jenkins, 1990).

Size and complexity of project:

> *What was the elapsed time of the project development?*
>
> *Approximately how many person-months did the project take?*
>
> *On average, how many staff worked on this project at one time?*

One of the aims of using a prototyping approach is to develop systems **fast** (Martin, 1988). The ideal team size for prototyping should be small in order to keep the number of communication lines as few as possible (Brooks, 1982).

What were the training levels of the staff involved?

Lack of training in the products and methods used can have a significant impact on the development process, (Carey, 1990). [The author has experience in using a 4GL environment where training in the 4GL was given to the programmers, but not to the analysts.]

What were the experience levels of the staff involved?
Experience in software development is of particular importance when prototyping, as a certain amount of fast-tracking is often involved and without sufficient background, this may be used at inappropriate times and phases of development (Carey, 1990).

What was the level of user involvement? (In days per week).
The involvement of the user(s) is considered to be of prime importance to the success of the project (Necco et al, 1989). The higher the level of involvement, the more likely the project is to succeed. This is mainly due to two factors: firstly, the quality of the user's knowledge leads to a more accurate and useful system; secondly, users feel that they have more 'ownership' of the system, because of the amount of input they have made to its design.

Do you consider the project was a success? If not, why not?
The developers' assessment of the success of the project should affect their attitude to future

development using prototyping. Their reasons for viewing the project as less than successful could be very relevant to other projects and other developers.

Do you think the user would consider the project was a success? If not, why not?
The users' assessment of the success of the project may be based on totally different criteria to that of the developers.

What criteria were used to judge the level of success?
Respondents were asked to rate the criteria given, plus their own criteria, in order of importance.

At what point in the system development process did you identify the critical success factors?
If the critical success factors were not identified at the start of the project, the development process may have followed a course that was not as focussed as it should have been. Boehm (1987) considers it 100 times more expensive to fix a problem after delivery of the system, than it is to fix it during the requirements analysis or early design phases. Thus, if the critical success factors have not been identified early in the project development there is a greater risk of the system not meeting the user requirements.

Was the system delivered on time? Indicate how much

it differed from the schedule.
There are many factors affecting the time schedule. The methods used to estimate the schedule and the experience of the estimator, being the most important.

Was the system delivered within budget? Indicate how much it differed from the budget.
As the budget is often dependent on the time schedule, any problems with the methods used to estimate that schedule will also affect the budget. However, in some cases the budget is fixed before any estimation is made, as no more money is available. Due to these factors it would be useful to know what constraints there were on the project, but unfortunately this is an area that companies may be loathe to discuss with outsiders.

Have you changed, or do you intend to change, the way you estimate time and cost of a project? If yes, in what way?
Unless the time and budget estimates have been particularly accurate, it is hoped that the methods used to produce them will be refined, in the light of each prototyping experience. In the author's experience estimates of time and cost are not always made in the most optimal manner. Time estimates are not always formulated as there are external deadlines existing over which the developers have no control. There may be a limited amount of money available or, if the project was

put out to tender, the prospective developers may have underestimated the cost in order to win the tender.

What development languages and/or tools were used?
A good methodology may not bring about a successful project if the language and/or tools used are poorly supported or inappropriate for the task required of them.

What did you feel were the strengths and weaknesses of the development languages and tools used?
Respondents were asked to rate the strengths and weaknesses listed, plus those that they add to the list, in order of importance.

What methodology was used?
A selection of the most widely used methodologies is given for respondents to choose from, whilst allowing them to add the methodology they used, if it is not listed.

What did you feel were the strengths and weaknesses of the methodology used?
Respondents were asked to rate the strengths and weaknesses given, plus those that they add to the list, in order of importance.

What method of prototyping was used?

There are three options given: Iterative, Piloting and Modelling, with a description of what each of these entails.

What refinements were made to the development process used?
This is a most important question as it should show what the developers felt needed to be improved or changed in the development process, when using a prototyping approach.

How much did the methodology used affect the success of the project?
Although this is a very subjective viewpoint, it is important to know how much confidence the developer had in the methodology used.

What improvements could be made to the methodology used?
In determining what makes a successful methodology for prototyping, the responses to this question should be most helpful.

4.5 Data Analysis

As far as is practicable the data gathered have been organised in a tabular form to make analysis easier. However, not all the data suits this approach and in such case the description is textual.

The final analysis attempts to take into consideration all the variables that affect the success of a project, prior to any conclusions or recommendations being made.

5. Findings

5.1 Analysis of Questionnaire.

5.1.1 Additional options to open-ended questions.

A number of questions asked the respondents to add their own options to the answers if necessary. These additional options have been categorised by the author, based on her understanding of their meanings, in the following tables:

Question 9 -What criteria were used to judge the level of success?

Table 1

Additional criteria - question 9

Additional criteria	No. of responses
requirements satisfied	1
provision of accurate information	1
access to historical data	1
saved time, relative to previous system	1
speed of reporting	1
decommission of old platform	1

Question 15 - What did you feel were the strengths and weaknesses of the development languages and tools used?

Table 2

Additional criteria - question 15

Additional criteria	No. of responses
productive environment	1
easy to develop	1
provides right sort of functions	1
Time Series Database	1
corporate standard	1
capable of handling large databases quickly	1
allowed rapid development	1
good end-user appearance	1
requires other mainframe software knowledge	1
excessive resource requirements	1
poor response times/performance	2
high operating costs/cost of products	2
lack of use (community)	1

Question 17 - What did you feel were the strengths and weaknesses of the methodology used?

Table 3

Additional criteria - question 17

Additional criteria	No. of responses
fit for the purpose	1
designed to maximise time available in hands-on mode	1
developed informal methodology as went along	1
examples of deliverables	1
involved regular user input	1
allowed use of prototyping	1
promoted poor project management	2
no capacity planning done	1
not seen as "formal" approach	1
laborious/long-winded	1
difficult to maintain without a case tool	1
lacking in depth	2
required correct (management and technical) resource	2

5.1.2 Several of the options added are very similar to those that very given in the questionnaire. This implies that the respondent felt that the slight difference in definition was important enough to state implicitly. These issues are discussed further in section 5.3.

These additional options are not included in the tables in section 5.1.3.

5.1.3 Options that were never referenced.

Only in question 17 were there options not referenced directly.

Question 17 - What did you feel were the strengths and weaknesses of the methodology used?

- difficult to use
- too restrictive in its framework

Neither of these options were actually referenced in the completed questionnaires. However, three of the options added were:

- laborious/long-winded
- difficult to maintain without a CASE tool
- lacking in depth.

The first two imply that the methodology probably is difficult to use, the third could imply that it is too restrictive.

5.1.4 Summary of responses.

Question 1 - What was your role in the project?

Table 4

Respondent's role in the project

Role	Number	%
Project Manager	7	36.8
Project Leader	3	15.8
Analyst/programmer/developer	3	15.8
All the above	3	15.8
Client Project Manager	1	5.2
Management	1	5.2
"Fixer"	1	5.2

Question 2 - Type of project.

Table 5

Type of project

Type of project	Number	%
DSS / MIS	12	63.1
Financial / Accounting	3	15.8
Other	4	21.1

Question 3 - Size and complexity of project.

What was the elapsed time of the project development?

The range was 2 to 36 months, with a mean of 11.3 months.

Approximately how many person-months did the project take?

The range was 1.5 to 390 person-months, with a mean of 67 person-months.

On average how many staff worked on this project at one time?

The range was 1 to 15, with a mean of 3.7 staff. Where the respondent gave a range, such as 5 to 6 staff, the lower figure was used in the calculation of the mean.

Question 4 - What were the training levels of the staff involved?

Table 6

Staff training levels

Training levels of staff (%)	0-20	21-40	41-60	61-80	81-100
High	9.37	6.25	0	6.25	15.6
Medium	6.25	9.37	12.5	0	12.5
Low	6.25	6.25	6.25	3.13	0

The respondents were asked to give the percentage of staff who had high, medium and low levels of training.

Question 5 - What were the experience levels of the staff involved?

Table 7

Staff experience levels

Experience levels of staff (%)	0-20	21-40	41-60	61-80	81-100
High	6.45	6.45	6.45	9.67	25.8
Medium	6.45	0	12.9	3.22	6.45
Low	9.67	3.22	3.22	0	0

The respondents were asked to give the percentage of staff who had high, medium and low levels of experience.

Question 6 - What was the level of user involvement?
The range was from less than 1 day per week to 5 days per week, with the mode being 1 day per week.

Question 7 - Do you consider the project was a success? If not, why not?

Table 8

Respondent's view of success of project

Response	Number	%
Yes / overall yes	14	73.7
Eventually	1	5.2
Partially	1	5.2
No/ Not satisfied/ Questionable	3	15.8

Question 8 - Do you think the user would consider the project a success?

Table 9

User's view of success of project

Response	Number	%
Yes / overall yes	16	84.2
Eventually	1	5.3
In parts	1	5.3
No	1	5.3

Question 9 - What criteria were used to judge the level of success?

Five criteria were supplied and the respondents were asked to add any of their own to the list and to rank all those that were applicable, in order of importance. Six other criteria were added, but these tended to be quite specific to particular projects. The four most commonly cited success criteria are shown in Table 10.

<u>Table 10</u>

<u>Ranked criteria of success</u>

Criteria	Rank				
	1	2	3	4	5+
User satisfaction	8	2	3	3	3
Improved management info.	5	6	3	2	0
Improved planning	1	3	4	2	3
Management goals	4	2	2	4	2

Question 10 - At what point in the system development did you identify the critical success factors?

Table 11

Identification phase of critical success factors

System development phase	Number	%
Project Initiation	9	47.4
Feasibility Study	1	5.3
Analysis and Design stage	5	26.3
System Testing	1	5.3
Implementation	1	5.3
Other / not applicable	2	10.5

Question 11 - Was the system delivered on time? Indicate how much it differed from the schedule.

Table 12

Project completion time

Project completion time	Number	%
Early	0	0
On time	7	36.8
25% late	5	26.3
> 25% & < 500% late	4	21
500% + late	2	10.5
Not applicable	1	5.3

Question 12 - Was the system delivered within budget? Indicate how much it differed from the budget.

Table 13

Project cost compared to budget

Project cost	Number	%
Under budget	1	5.3
On budget	7	36.8
25% over	3	15.8
>25% & <500% over	3	15.8
500% + over	2	10.5
Not applicable	3	15.8

Question 13 - Have you changed, or do you intend to change, the way you estimate time and cost of a project? If yes, in what way?

Table 14

Change of estimation methods

Change estimation method?	Number	%
Yes	11	57.9
No	8	42.1

Question 14 - What development languages and/or tools were used?

Ada, AME (a 4GL environment), Artemis, C, CICS, COBOL, Code locator, dBXL, DB2, Excel, GENIFER, Gupta SQL Windows, Hyperchannel, Interbase, JCL, Natural, Oracle RDBMS, Oracle development tools (SQL*FORMS,SQLMENU, SQLPLUS, SQL*REPORTWRITER), PILOT command centre, Powerhouse, Quicksilver, Rally, RPG, SAS, SQL, SYNON2, TODAY, Toolset, Turbo Pascal (abandoned).

Question 15 - What did you feel were the strengths and weaknesses of the development languages and tools used?

Table 15

Ranked strengths and weaknesses of languages and tools

Strengths/weaknesses	Rank				
	1	2	3	4	5+
Easy to use	5	5	2	0	0
Good interfacing capabilities	3	2	2	2	2
Provided most required functions	4	4	3	1	2
Widely used	2	1	2	0	3
Good technical support	0	0	2	5	1
Difficult to use	1	2	1	0	0
Poor interfacing capabilities	0	2	0	0	1
Lack of functionality	0	0	0	1	0
Poor technical support	3	0	1	2	0

Question 16 - What methodology was used?

Table 16

Methodology used

Methodology	Number	%
Internally written methodology	9	47.4
Internally written methodology / Prototyping	1	5.3
Evolutionary	1	5.3
None/no formal methodology	4	21
APT	1	5.3
Powerdesign	1	5.3
PRISM	1	5.3
SYNON2	1	5.3

Question 17 - What did you feel were the strengths and weaknesses of the methodology used?

Table 17

Ranked strengths and weaknesses of methodology

Strengths/weaknesses	Rank				
	1	2	3	4	5+
Easy to use	3	6	4	0	0
Provided a good framework for development	7	0	0	2	0
Specifically addressed prototyping	2	3	6	0	0
No guidelines for prototyping	1	1	0	1	0

Question 18 - What method of prototyping was used? (Iterative, Piloting, Modelling)

Table 18

Method of prototyping

Method of prototyping	Number	%
Iterative	13	68.4
Piloting	2	10.5
Modelling	1	5.3
Piloting & Iterative	1	5.3
Piloting & Modelling	1	5.3
All three	1	5.3

Question 19 - What refinements were made to the development process used?

Respondents gave refinements that they intended to make, as well as those that they had already made. No distinction will be drawn between the two groups. They are all listed below:

- Use software tools that are more flexible and thus, more suited to a prototyping approach.
- A more flexible development process was used, involving the developer and the user in the prototype process, where each prompted

discussion and further system development.

- Some tidying up of the process - deleting the test environment.
- Needed to handle implementation for a large, multiple site organisation.
- Introduced staff impact documents.

Question 20 - How much did the methodology used affect the success of the project?

<u>Table 19</u>

<u>The effect of the methodology on the outcome of the project</u>

Effect of methodology	Number	%
Not at all	0	0
Small amount	4	21
Reasonably important	1	5.3
Highly significant	10	53
Totally responsible	3	15.8
Other	1	5.3

Question 21 - What improvements could be made to the methodology used?

All comments made by the respondents are listed below:

- A more structured traditional methodology might have been more suited to the inflexible tools and mainframe processing.
- The methodology used was more comprehensive than the project required. A method of "short-cutting" would be desirable.
- Incorporate capacity planning.
- Improve project management.
- Use the associated case tool to automate the laborious documentation process and to generate code.
- Formalise what was actually done as the basis of a methodology suitable for the development of DSS. (Assess whether this methodology would be suitable for developing transaction [processing] systems.)
- Address prototyping.
- Provide a better structure.
- More documentation.
- Formal reviews.
- Quality assurance checks.
- More examples. More about training and implementation.
- Guidelines as to which type of prototyping

should be used for different projects. (The respondent suggested that iterative prototyping should not be used for "mission critical applications".)

- Only one person should manage all aspects of the prototyping phase.
- Need an improved coding language with functions that match the prototyping tool better.
- Need to have a much better understanding of the scope and requirements of the required system before prototyping starts.
- If a methodology had been used it might have shown up the weaknesses in the original plan.

5.1.5 A section was included at the end of the questionnaire for the respondents to give their name and address if they wished for a copy of the results.

Table 20

Respondents wishing to know the results of the study

Respondent wishes to know results YES/NO	Number	%
Yes	13	68
No	6	32

The fact that 68% of the respondents wished to know the results of this study, indicates that they are interested in knowing how other prototyping projects are handled, and thus, what improvements could be made to their own methods. This is a fairly high percentage in comparison with Doke's survey (1990) where only 32% of respondents supplied name and address. This could be due to an increased interest in prototyping having occurred in the intervening three years.

5.2 Additional Information from Interviews

The question of whether a detailed requirements analysis was carried out prior to the prototyping commencing was not asked implicitly in the questionnaire. When this information was not present, respondents were contacted about this and any other information that was unclear from their questionnaire.

Table 21

Requirements analysis prior to prototyping

Requirements analysis first?	Number	%
Detailed	9	47.4
High-level/functional	2	10.5
Insufficient	2	10.5
None	3	15.8
Not known	3	15.8

One respondent commented that they had been 'burnt' a couple of times in the past, because of the lack of a thorough requirements analysis.

Early on in the research for this study, a project manager was interviewed from a company who had used prototyping for the development of 6 projects.

Unfortunately, he had left the company by the time the questionnaires were sent out and no other staff member completed one. Thus, the following information could not be included in any of the tables of data, but due to the experience of the developers it is of relevance to this study.

The 4GL development environment used was 'TODAY', which worked well. All of the projects developed using prototyping were successful, with the exception of one project for which no requirements analysis was done prior to commencement of prototyping. For all other projects a thorough, detailed requirements analysis had been done. As they had successfully developed other projects for one particular client, who therefore was considered to be an experienced user with prototyping, they decided to prototype in order to define the requirements. [Owen (1989) states that prototyping is "viewed primarily as a means for obtaining requirements from the users".] As the TODAY environment enabled changes to be made fast and easily to the prototype, daily modifications were made, but the client was never satisfied and the requirements never finalised. It was several months and many software changes later before they realised that the client was not able to define the requirements.

5.3 Discussion of Results

The questionnaires were analysed in an attempt to determine trends of identifiable patterns of factors that either help or hinder the development process.

The diversity of the projects and their development methods made it difficult to draw meaningful conclusions from the data collected. In order to verify any trends in the data, the answers to the open-ended questions needed to be analysed and assessed with the other data.

52.6% of the questionnaires were completed by either the Project Manager or Project Leader. 15.8% were completed by an Analyst/Programmer/developer. 15.8% of the respondents were fulfilling all of these roles. In addition, one client MIS manager, one 'management view' person and one "fixer" type person completed questionnaires.

The type of project varied widely. Decision support systems and management information systems were represented more than any other type of system. Only three of the 19 projects were transaction-processing systems (Martin and Carey, 1991).

The training levels of the staff tended to be higher for one person projects than for larger projects. The larger projects included people with lower levels of training. There were far more highly **experienced** people on all the projects, relative to highly **trained** people. The respondents felt that general experience in system development was important when undertaking a project using prototyping, whereas specialised training can always be obtained during the project if necessary.

User involvement fell mainly into two categories: either full-time on the project, or one day a week on the project. Five of the projects had user participation of less than one day a week, but these were all for one person projects, four of which came in close to time and budget. User involvement and feedback are considered (Carey, 1989; Rowen, 1990; Tate, 1990) to be essential to the success of the project.

73.7% of respondents felt that the project was successful; if not immediately, eventually (a further 5.2%). This is not surprising considering that 68.4% of projects used an iterative approach; thus, they continued to refine the system until it was acceptable. Within this type of development environment it would be unusual to completely abandon the project, unless it was found to be totally infeasible. One respondent stated that they felt the project was not a success **because** the prototyped system was installed as the final version. His objections to this are due to the fact that the prototyped system has been continually modified and redesigned, the resultant system, he considers, is a "band-aided version". Necco et al. (1989) cite "users who wanted to use the prototype as a production system" as the "second most reported problem" and they encourage their readers to thoroughly consider the disadvantages of doing so. When systems require fast response times, as in real-time systems, using the resultant prototype of an iterative approach may be unsuitable. In this case a modelling approach should be used where the operational system is built in an efficient development language, using the prototype as the requirements specification (Tate, 1990). Another project was already a "failure" before the respondent took it over with the intention of "fixing it up".

84.7% of respondents felt that the user would consider the project a success. Some supported this with statements from the users. Performance was the only type of problem mentioned. It is interesting that this is a higher proportion than of the respondents themselves, as often the users had not known of the problems that had occurred during development.

When asked what criteria were used to judge the level of success, in spite of user satisfaction being selected most commonly, more than half of the respondents did not put it first. This raises issues on the nature of project success that need to be researched further. Six criteria were added by the respondents, one of which was "requirements satisfied". To differentiate between this and "user satisfaction" (which was a supplied option), implies that although the requirements of the project have been met technically, the user might not be satisfied with the project.

For 47.4% of projects, the critical success factors were identified at the Project Initiation stage. This was true for projects that ran on time as well as for those that ran very late and over budget. However, all the projects where the critical success factors were not identified until the Analysis and Design stage, or later, ran very late and over budget.

Of the projects that were completed late, only one of them came in more over budget than over time, in terms of percentages. All the respondents whose projects came in on time and budget do not intend changing their methods of estimation. All those that came in late and over budget have already, or will in the future, change their estimation methods. One project came in late but on budget and the developers do not intend changing their methods of estimation.

The products used were not all advanced development tools as might be expected (Owen, 1989). Instead the products range from CICS COBOL and JCL, spreadsheet and database products, through to various 4GL environments, such as Oracle and Today. This was similar to the experience of Martin and Carey (1989) who found that prototyping in a 3GL was quite common.

The strengths and weaknesses of the development language and tools had thirteen criteria added by the respondents. Two of the thirteen criteria added were very similar to those offered. One of them, "easy to develop" appears to be emphasising the ease of developing systems, as opposed to the 'ease of use' of the product. The other criteria added was "provides right sort of functions," as opposed to the option that was offered which was "provided most required functions". The respondent seems to be stressing the appropriateness of the type of functions to the task, rather than just the provision of most of the functions needed. Necco et al. (1989) cited the lack of appropriate tools as a significant problem. This was not found to be a general problem in this study, probably because there has been a great increase in the number and sophistication of the available tools and development environments since 1989.

52.7% of the projects used a methodology written within their own organisation. Or'y four different types of commercial methodologies were used.

The strengths and weaknesses of the methodology had thirteen criteria added by the respondents. Five of the additional thirteen criteria can be compared to four of the options offered. Both "fit for the purpose" and "examples of deliverables" could both be considered to be part of 'a good framework for development'. However, as the respondents have specifically added these options, it implies, in the first instance, that although the methodology is adequate, it does not necessarily provide a good framework and, in the second instance, the inclusion of examples of deliverables has improved the useability of the methodology. Glasson (1989) uses deliverables "to define a system of being in a particular state of evolution". By providing extensive examples of deliverables, the developer is able to use those that are appropriate for the system being developed, allowing the system development process more flexibility than is normally possible.

In order to draw any conclusions from the data, it is necessary to know the respondents' definition of prototyping. There are three main views of prototyping: iterative (or evolutionary) - the final iteration becomes the production system; piloting (or rapid) - used to determine feasibility and test alternative solutions; modelling (or throwaway) - to determine user requirements, screen and report requirements and processes to be performed on the data; the final model is discarded and rewritten, generally using a different method or language.

One respondent stated that they had used all three types of prototyping for different parts of the project, but other than that only four projects had used piloting, one project used modelling and all the others used an iterative approach. The percentage of respondents using the iterative approach was 68.4%, plus 5.3% who used both an iterative and a piloting approach. This gives a total of 73.7% who used an iterative approach, which is very similar to the results found by Doke in his survey (1989), where 71% used an iterative (Doke calls this evolutionary) approach.

The refinements made to the development process were almost all intended to improve the flexibility of the products and the methodology used. Respondents felt that prototyping was a flexible approach and therefore needed equally flexible tools. These refinements included greater involvement of the user in the process, which prompted discussion and further system development.

For all the projects that were completed on time, the methodology was said to be either "highly significant" or "totally responsible" for the successful outcome of the project. Of those that were completed late, most said that the methodology had only a small amount of impact.

The refinements made to the development process were mainly to increase flexibility, whereas the suggested improvements to the methodology are very much in favour of more formalisation, better structure, more documentation, formal reviews, quality assurance checks, incorporate capacity planning, more examples and more guidelines. Automated case tools should be used for documentation and code generation. Project management needs some improvement and that should happen if a methodology was available that incorporated the improvements suggested. One respondent stated that weaknesses in the project plan might have shown up if a methodology had been used!

The need to have a better understanding of the scope and requirements was listed as an improvement to the methodology. When no mention was made as to the requirements analysis carried out this was discussed during follow-up interviews. The information gathered during these interviews indicated that the timing and amount of detail involved in the requirements analysis was of significance to the success of the project. This view is reflected in the literature. Necco et al. (1989) found that inadequate requirements analysis was a major problem when prototyping. They felt that "in the prototyping approach, the focus is on the physical design, not the logical design." This can lead to the wrong problem being solved. Thus, their assertion that "prototyping shou! ' be used to support adequate systems analysis, not replace it."

Jenkins (1990) describes prototyping as an "accelerated methodology" that should be an "alternative" to the requirements definition phase of the standard development life cycle. His methodology requires that the user's basic needs are identified, but that the purpose of the initial prototype is to define the detailed requirements of the system.

Carey (1990) states the "the methodology should include thorough requirements definition and design stages before any prototyping is attempted".

In order to judge more clearly the effect on the project, the timing and detail of the requirements analysis has been tabulated according to how late of over budget the project was.

Table 22

Requirements analysis/project on time/budget

Requirements analysis / amount over time or budget	= time or = budget		< 200% over		> = 200% over	
		%		%		%
Detailed	5	27.7	2	11.1	2	11.1
High-level	1	5.5	0	0	0	0
Insufficient	0	0	0	0	2	11.1
None	1	5.5	0	0	2	11.1
Not known	2	11.1	1	5.5	0	0

There were four projects that came in late or over budget, where a detailed requirements analysis had been carried out. Each of these projects has been examined to determine what caused the overrun.

Although a detailed requirements analysis was done for the first of these projects, the complexity of those requirements was not fully investigated. This caused the time and budget estimates to be unrealistic.

The second overdue project was 25% over time and 25% over budget, which would have been considered acceptable in the past, using traditional system development methods. The respondent felt that the use of Function Point Analysis would have improved the estimation techniques. The other factor that could have affected the project was that the methodology was "laborious and long-winded" and "difficult to maintain without a case tool". This particular methodology has an associated case tool to automate the documentation and generate code. The respondent stated that this would be used in the future.

The third overdue project was subject to a number of adverse factors. The project was scheduled and budgeted before sufficient information was gathered concerning the complexity of the requirements. The time and budget were underestimated in order to win the tender for this project. A new product was used for the development, for which there were no experienced practitioners in Australia. The staff, although experienced in system development, were not sufficiently trained in this new product.

The last of these overdue projects was 200% overdue and 25% over budget. Although a requirements analysis was done, it took place five years before the system was developed. The baseline functional specification was at a fairly high-level and the appropriateness of this document was not ratified prior to the commencement of development.

Having analysed all the data collected it is necessary to consider what bearing it has on the research questions (section 2.5, page 22).

Are the current life-cycle methodologies appropriate for system development using a prototyping approach?
The comments elicited by the open-ended questions indicate that the current life-cycle system development methodologies are not sufficiently flexible when using a prototyping approach. Only 21.2% of respondents used commercial methodologies, the others used no methodology, used prototyping as the methodology, or used an internally-written methodology. This implies that the commercially available methodologies do not suit the needs of most developers. The strengths and weaknesses of the methodology that respondents rated as being most important during development were that it was easy to use, provided a good framework and specifically addressed prototyping. Where the methodology provided no guidelines for prototyping this was considered to have a negative effect on the development.

How does system development using prototyping differ from traditional system development?
The prototyping development process aims to clarify requirements as early as possible during development and to produce a final product faster than would be

possible using a traditional approach. Prototyping development has much greater involvement of the user than is normal in traditional development. There was wide use of 4GL development environments that enabled rapid development using screen builders, code generators and other tools that helped to produce a system prototype quickly. However, there are still prototyping projects being developed using tools and languages that are inappropriate for developing systems fast.

Does the system development life cycle need to be modified or is a totally new approach required?
The system development life-cycle is still relevant but needs more flexibility to allow iterations to take place for individual and groups of phases. It should be possible to omit or modify phases that are inappropriate to a particular project. Examples of different life cycles and deliverables that are suitable for specific types of project should be included.

Having addressed the research questions for this study it is worth looking at the questions that emerged from Doke's study to see if any of these can be answered.
When should the various methodologies be used?
The iterative approach was used in 71.7% of prototyping projects and was considered to be successful. Piloting was used to test new tools and to

ascertain the feasibility of particular functions. Modelling was used when performance was critical to the success of the system, as in real-time systems, and where the prototype development environment did not provide the required level of performance.

What is the impact of the methodologies on the traditional life cycle?
This question is answered by all three of the research questions for this study.

Is it appropriate to employ multiple methodologies concurrently?
A few of the respondents used all three types of prototyping when developing large projects. For prototyping to be successful it is necessary to be able to decompose the system into modules for development, this then allows the developer to select the approach most appropriate for each module.

As tools such as 4GLs become more popular and operationally efficient, what is the expected impact on the prototyping methodologies?
The only result relevant to this question is the increase in prototyping identified in this study as compared with earlier studies, due to the increased availability and functionality of the latest tools and development environments.

6. Conclusions and Implications

6.1 Conclusions

6.1.1 Conclusions to be drawn based on the findings.

There were two main areas indicated that had an effect on the success of a software development project. The first is the lack of flexibility in the methodologies used and to a lesser extent the inappropriateness of the development tools and languages. The second is insufficient requirements analysis. There is much literature that promotes prototyping as a methodology that can be used to define requirements and to develop the system. However, in practice it appears that prototyping, particularly when used iteratively, should be clarifying requirements, not defining them.

6.1.2 Alternative explanations for the findings

There are other factors that have affected the success of a project. The experience of the staff in system development, particularly when using a prototyping approach, will have a significant bearing on the project.

The size of the project is very important. A small project will often involve less developers and thus there are less lines of communication. It is faster to build the initial prototype which helps both the user and the developer to visualise where they are headed.

The development environment, tools and languages used can have a significant impact on the project. Attempting to prototype using CICS COBOL and JCL may not provide an optimal environment for rapid development.

The type of project is significant as some systems are inherently complex and careful consideration should be given as to whether a prototyping approach is suitable. The most suitable projects are those that are small or easily decomposed into modules.

6.1.3 Limitations of the study

The sample population was not particularly large. However, the results can still be generalised to other projects as there was a wide range of types of system which were representative of the general population. Other factors need to be taken into consideration such as the experience of the developers and the complexity of the project (6.1.2).

Additional information, such as the specific development stages and deliverables at each stage, would have made it easier to draw conclusions from the data. However, this would have made the completion of the questionnaire significantly more onerous and could have deterred participants from responding.

6.2 Implications

6.2.1 Implications for professional practice

A methodology is required that provides a good framework, but is flexible enough to handle different types and sizes of project. It should specifically address prototyping and should include guidelines as to how to select the most suitable prototyping approach for each project.

It should contain examples of different deliverables and various development cycles appropriate for each type of prototyping. It should include guidelines for training and implementation.

There should be automated tools available to handle documentation and code generation where possible.

6.2.2 Implications for further research studies

The next logical step in this research would be to discover more about the individual methodologies used and identify the parts that were useful for each project. From the information obtained an outline methodology could be built which would allow for different types of development. After discussions with experienced prototypers this could be expanded to include more detail, until there is a sufficiently developed framework for it to be tested on a new development project. Eventually, a complete methodology could be developed that could be adapted for any type of development strategy.

6.3 Conclusion to Thesis

Prototyping is becoming more popular for software development, but few developers are completely satisfied with the methodologies and tools available. There is a definite lack of case studies, examples and guidelines relating to prototyping: how to assess the suitability of a project for prototyping and which method of prototyping to use. A flexible methodology which would provide a good framework and supportive tools is required if the advantages of prototyping are to be maximised in the future.

7. Bibliography

An Accelerated Methodology. (1990). *System Development,* 10 (12), 5-7.

Boehm, B.W. (1987, September). Industrial Software Metrics top 10 list. *IEEE Software,* pp. 84-85.

Brooks, F.P. (1982). *The Mythical Man-Month.* Reading, MA: Addison-Wesley.

Budde, R. & Züllighoven, H. (1990, May). *Prototyping Revisited.* Paper presented at the IEEE International Conference on Computer Systems and Software Engineering, Tel-Aviv, Israel.

Carey, J.M. (1990). Prototyping: alternative systems development methodology. *Information and Software Technology,* 32 (2), 119-26.

Connell, J.L. & Shafer, L.B. (1989). *Structured Rapid Prototyping. An Evolutionary Approach to Software Development.* New Jersey: Prentice-Hall Inc.

Crinnion, J. (1989). The systems implications of Fourth Generation languages. *Journal of Information Technology,* 4 (2), 71-80.

Doke E. Reed. (1990). An industry survey of emerging Prototyping Methodologies. *Information and Management,* 18 (4), 169-76.

Fitzgerald, G., Stokes, N. & Wood, J.R.G. (1985). Feature analysis of contemporary information systems methodologies. *The Computer Journal,* 28 (3), 223-29.

Gladden, G.R. (1982). Stop the Life Cycle I want to get off. *ACM Sigsoft Software Engineering Notes,* 7 (2).

Glasson, B.C. (1989). Model of System Evolution. *Information and Software Technology,* 31 (7), 351-6.

Grindley, K. (1987). *Fourth Generation Languages, A Survey of Best Practice.* IDPM Publications.

Gryczan, G. & Kautz, K. (1990, May). *A Comparative Case Study of Prototyping Tools - Experiences and Conclusions.* Paper presented at the IEEE International Conference on Computer Systems and Software Engineering, Tel-Aviv, Israel.

Jenkins, A.M. (1990, September). Background, Usage and Future of Prototyping. *ShowCASE V, sponsored by the Center for the Study of Data Processing, Washington University, St.Louis, MO.*

Krista, R. & Rozman, I. (1989). A computer aided prototyping methodology. *ACM Sigsoft Software Engineering Notes,* 14 (6), 68-72.

Lea, R-J & Chung, C-G. (1990). Rapid prototyping from structured analysis: executable specification approach. *Information and Software Technology,* 32 (9), 589-97.

Lehman, J.A. & Wetherbe, J.C. (1989, summer). A survey of 4GL users and applications. *Journal of Information Systems Management,* pp. 44-52.

Martin, C.F. (1988). *User-Centred Requirements Analysis.* New Jersey: Prentice Hall.

Martin, J. (1982). *Applications Development without Programmers.* New Jersey: Prentice Hall.

Martin, J. (1985). *Fourth-Generation Languages volume I Principles.* New Jersey: Prentice Hall.

Martin, M.P. & Carey, J.M. (1991). Converting prototypes to operational systems: evidence from preliminary industrial survey. *Information and Software Technology,* 33 (5), 351-6.

Modha, J., Gwinnett, A. & Bruce, M. (1990). A review of Information System Development Methodology (ISDM) Selection Techniques. *Omega,* 18, 473-90.

Necco, C.R., Tsai, N. & Gordon, C.L. (1989). Prototyping: use in the development of computer-based information systems. *Journal of Computer Information Systems,* 30 (1), 62-6.

Owen, D.E. (1989). Prototyping: Essence of pragmatic IS development. *Information Strategy: The Executive's Journal,* 5 (2), 21-5.

Palvia, P. & Nosek, J.T. (1990). An empirical evaluation of system development methodologies. *Information Resources Management Journal,* 3 (3), 23-32.

Rowen, R.B. (1990). Software project management under incomplete and ambiguous specifications. *IEEE Transactions on Engineering Management,* 37 (1), 10-21.

Tate, G. (1990). Prototyping: helping to build the right software. *Information and Software Technology,* 32 (4), 237-43.

West, M.G. (1986). *Prototyping. State of the art report. (Chapter title - A taxonomy of prototyping-tools and methods for decision support and transaction systems).* Maidenhead: Pergamon Infotech.

Wojtkowski, W.Gregory & Wojtkowski, Wita (1988). Prototyping and its place in Information Systems Development. *ISECON '88 Seventh Annual Information Systems Education Conference - proceedings of the conference sessions.* Park Ridge, Il. USA: Data Processing Management Association.

Wojtkowski, W.Gregory & Wojtkowski, Wita (1990). *Applications Software Programming with Fourth-Generation Languages.* Boston: Boyd and Fraser Publishing Company.

8. Appendices

8.1 Blank Questionnaire

QUESTIONNAIRE

1	What was your role in this project?	Project Manager Project Leader Other ....
2	Type of project. eg. payroll, HRM, DSS, inventory, etc.	
3	Size and complexity of project: What was the elapsed time of the project development?	
	Approximately how many person-months did the project take?	
	On average, how many staff worked on this project at one time?	
4	What were the training levels of the staff involved?	.... % high .... % medium .... % low
5	What were the experience levels of the staff involved?	.... % high .... % medium .... % low
6	What was the level of user involvement? [In days per week.]	5 4 3 2 1 less than 1
7	Do you consider the project was a success? If not, why not?	
8	Do you think the user would consider the project was a success? If not, why not?	

9	What criteria were used to judge the level of success? List others as required. [Rank them in order of importance, (1 is of highest importance).]	Management goals		User satisfacuon	
		Improved management information			
		Improved planning			
		Improved communication			
10	At what point in the system development did you identify the critical success factors?	Project initiation Feasibility study Analysis and design stage System testing Implementation Other			
11	Was the system delivered on time? Indicate how much it differed from the schedule.	EARLY 75% 50% 25% ON TIME 25% 50% 75% 100 % + LATE If 100%+ state amount .... %			
12	Was the system delivered within budget? Indicate how much it differed from the budget.	UNDER 75% 50% 25% ON TIME 25% 50% 75% 100 % + OVER If 100%+ state amount .... %			
13	Have you changed, or do you intend to change, the way you estimate time and cost of a project? If yes, in what way?				
14	What development languages and/or tools were used?				

15	What did you feel were the strengths and weaknesses of the development languages and tools used? [List others as necessary.] [Rank them in order of importance, (1 is of highest importance).] [Delete those that are not applicable.] If you are using more than one tool/language, please include any additional information on a separate sheet and attach it to the questionnaire.	Easy to use		Difficult to use	
		Good interfacing capabilities		Poor interfacing capabilities	
		Provided most required functions		Lack of functionality	
		Widely used		Poor technical support	
		Good technical support			
16	What methodology was used? [List others as necessary.]	Internally written methodology		SSADM	
		PRISM		PRIDE	
		APT			
		SDM ??			
17	What did you feel were the strengths and weaknesses of the methodology used? [List others as necessary.] [Rank them in order of importance, (1 is of highest importance).] [Delete those that are not applicable.]	Easy to use		Difficult to use	
		Provided a good framework for development		Too restrictive in its framework	
		Specifically addressed prototyping		No guidelines for prototyping	

18	What method of prototyping was used?	Iterative [The final iteration becomes the production system.]		Piloting [Used to determine feasibility and test alternative solutions.]		Modelling [To determine user requirements, screen and report requirements, processes to be performed on the data. The final model is discarded and rewritten, generally using a different method or language.]
19	What refinements were made to the development process used?					
20	How much did the methodology used affect the success of the project?	Not at all	Small amount	Reasonably important	Highly significant	Totally responsible for the project outcome
21	What improvements could be made to the methodology used?					

Complete the following information if you wish to receive a copy of the collated results.

22	Name and Job Title	
23	Name and address of organisation	
24	Telephone number Fax number	
25	Project name	

8.2 Covering Letter

Edith Cowan University
Department of Computer Science
2, Bradford Street
Mount Lawley
WA 6050

date

Dear *participant*,

As we discussed on the telephone, I am enclosing *n* copies of my questionnaire.

I am collecting data for my Masters thesis, "An Investigation of Methodologies for Software Development Prototyping".

The purpose of the study is to determine how the methodology used in a prototyping development impacts on the success or failure of a project.

It is necessary to know as much as possible about the development environment in order to ascertain which elements of the methodology affect the outcome of the project and which are due to other factors, such as the tools and languages used.

I would be grateful if you could complete the questionnaire as accurately as possible.

If you wish to have a copy of the collated results, please complete the section at the end of the questionnaire, with your name and address.

All the information gathered will be strictly confidential.

Thankyou very much for giving up some of your time for this activity, it is much appreciated.

Yours sincerely,

Sue Jones

8.3 Spreadsheet of Questionnaire Responses

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	
question 1	Role?	Project Leader	Project Manager	Project Manager	PL/developer	Project Manager	PM/PL/A/P	Project Leader	Project Manager	Project Leader	Project Manager	Project Manager	Analyst/Progra	Analyst/Progra	Client PM	Management	Project Manager	Mender	Prog/Developer	All	
question 2	Project type?	DSS	Works manage	EIS	Risk manageme	Proc. monitor/Q	Accts. receivabl	Fin. & Work ord	DSS	HRM	Accounting	Material require	Diagnostic syst	Catalogue syste	Lab test mgt	Prod. schedulin	Prod. statistics	Prod. record. &	Phone acct.s/w	Comms system	
question 3																					
	size	14	12	36	3	3	6	30	7	21	8	30	2.5	2	16	4	4	11	4	2.5	11.31579
	pers-mths	40	80	30	2	2	4	600	42	84	40	390	2	1.6	30	3.6	18	22	4	2.5	87.13158
	staff	5	6	2.7	1	1	1	15	6 to 8	3	5	13	1	1	2 to 3	1.2	4	2	1	1	3.756824
question 4	Training levels?																				
	high	100			100		100	10		20	40	10			75	100	80	25	100		
	medium		60			100		60	30	30	60	60	100	100	20		20	25		100	
	low			10				30	70	50		30			5			50			
question 5	Experience levels?																				
	high	100	80	60	100	100	100	15	50	40	40	10			75	100	80		100	100	
	medium							70	50	50	60	50	100	100	20		20	50			
	low							15		10		40			5			50			
question 6	User involvement (days per week)?	1	4	1	less than 1	less than 1	1	6	5	1	2	5	less than 1	1	5	1	5	2	less than 1	less than 1	2.705714
question 7	Do you consider project a success?	Partial	Overall, yes	Yes	Yes	Yes	Yes	Yes	Yes	Eventually	Yes	Not satisfied	Yes	Yes	Questionable	Yes	Yes	No	Yes	Yes	
question 8	Does the user consider the project a success?	No	Overall, yes	Yes	Yes	Yes	Yes	Yes	Yes	In parts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Eventually	Yes	
question 9	Success criteria.																				
	management goals	5	5	2	3		3	1		1		4	4	1	4	2	1	4			
	improved management information	1	1	3	2	1	2	4	2	2		2	1	3	3	4	2	1			
	improved planning	3	2			4	4	3		3		3	6	5	3	1	5	2			
	improved communication	6	6	4		2	5			4		1	7	8	6	6	3	5			
	user satisfaction	4	4	1	1	3	1	2	1	5	1	5	5	2	1	3	4	3	1	1	
	requirements satisfied										2										
	provision of accurate information		3																		
	access to historical data												2								
	saved time, relative to previous system													4							
	speed of reporting												3								
	decommission of old platform	2																			
question 10	Critical success factors identified when?	A & D	A & D	N/A	PI	PI	PI	PI		PI	PI	PI	FS	PI	A & D	PI	A & D	Implementation	ST	A & D	
question 11	System delivered on time?	200% late	25% late	N/A	On time	On time	On time	25% late	On time	500% late	25% late	200% late	25% late	On time	200% late	25% late	on time	1100% late	300% late	On time	
question 12	System delivered within budget?	Over budget	60% over	N/A	On budget	On budget	On budget	25% over	On budget	250% over		400% over	On budget	25% under	25% over	on budget	on budget	2000% over	300% over	25% over	
question 13	Changes to time and cost estimation?	Yes	Continuous	No	No	No	No	Yes		Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes	
question 14	What development languages/tools used?	Oracle	Artemis/DB2	PILOT	PILOT	PILOT	SYNON2	Powerhouse &	SQL windows	Rally/COBOL	Oracle	Oracle & others	SAS & m/f s/w	dBXL & Q'silver	4GL/CICS/COB	Genifer, Excel,	Genifer, RPG	Today	Tu. Pascal, Ada	C, Ada, code locator	
question 15	Strengths and weaknesses of languages/tools?															Gen / Excel	Gen / AS400 world				
	easy to use	1	2	1	2	2	3	4	3					2	1	2	1 0				
	good interfacing capabilities	5	4	4	1	3	2	1	1					6	3			2			
	provided most required functions	2	3		3	4	1	5	2	6				3	2	1 2		1	1		
	widely used	3	6					2	5		6			1		3	1				
	good technical support	4				6	4	3	4		4			4				3			
	productive environment		1																		
	easy to develop			2																	
	provides right sort of functions			3							1										
	corporate standard										3										
	handles large databases quickly												1								
	allows rapid development										2										
	good end-user appearance																2 0				
	time series database					1															
	difficult to use									3			-1								
	poor interfacing capabilities		-5							2			-2				2		-2		
	lack of functionality												-4								
	poor technical support		-4							1			-3		4				-1	1	
	requires other mainframe s/w knowledge												-6								
	excessive resource requirements		-2																		
	poor response times/performance		-3							4											
	high operating costs/cost of products		-1	-1																	
	lack of use (in community)														5						
question 16	What methodology used?	APT	Evolutionary	Own methodol	Own methodolo	Own methodolo	SYNON2	Powerdesign	No formal meth	Own methodolo	Own methodolo	PRISM	Own methodolo	Own methodolo	Own/Prototypin	Own methodolo	Own methodolo	none	none	none	
question 17	Strengths and weaknesses of methodology?																				
	easy to use	2	1	2	2		3	3	1		1		2	3	3	2	2				
	provided a good framework for development	1		4	4		1	1	1				1	1	1						
	specifically addressed prototyping	3	2	3	3		2		1		1		3	2		3	3				
	fit for the purpose			1		1															
	designed to maximise time in hands-on mode				1																
	examples of deliverables																				
	involved regular user input													4							
	allowed use of prototyping														2						
	developed informal methodology as went along								1												
	difficult to use																				
	too restrictive in its framework																				
	no guidelines for prototyping		-2							1					4						
	promoted poor project management		-1					2													
	no capacity planning done		-3																		
	not seen as "formal" approach			-1																	
	laborious/long-winded																				
	difficult to maintain without a CASE tool							4													
	lacking in depth							5		2											
	required correct (mgt & technical) resource															1	1				
question 18	Method of prototyping?	Piloting	Iterative	All three	Iterative	Iterative	Iterative	Iterative	Piloting	Iterative	Iterative	Iterative	Iterative	Iterative	Iterative	Piloting/Modelli	Modelling	Iterative	Piloting/Iterative	Iterative	
question 19	Refinements made to development process?	none		minor ones	none		none	none			none	a few changes	change tools	user discussion	none						
question 20	Effect of methodology on project success?	reas. important.		highly significan	highly significan	highly significan	totally responsi	small amount	highly significan	small amount	highly significan	highly significan	small amount	highly significan	highly significan	totally responsi	totally responsi	highly significan	highly significan	small amount	
question 21	What improvements could be made to methodology?	Shortcutting	Cap'ty. plan/mg	it's fine	don't know		no major improv	use case tool	formalise it	cater for protot	doc./QA/review	sgs/train'g/mgt	more structure	none	use modelling	one consultant	better lang & in	scope	Use one		
Interview	Detailed requirements analysis done first?	Yes	Conceptual - to determine reqs	General reqs. then informal			Functional reqs.	Replacement system - yes	Yes - user workshops	Yes	Yes	Insufficient reqs. analysis	Yes	Yes	Too high-level Spec. 5 yrs. o/c	No	Yes	No	No	Yes	