A study of process improvement activities for web development processes within a small company

Donkin, Joanna M

A Study of Process Improvement Activities for Web Development Processes within a Small Company

Joanna Mary Donkin

Submitted for the degree of Master of Science, University of Durham, 2002

Abstract

This thesis describes activities carried out in order to improve a small company's web development process, specifically focusing on the areas of reuse and web accessibility.

CACDP are the examinations board for British Sign Language and other related disciplines. Within the domain of web development they have no formal processes and no skills or knowledge with which to improve them. They wish to develop four new web based products, and to apply accessibility guidelines to both these and their existing web site. The areas of web development, process improvement and reuse are investigated, specifically in relation to their suitability for CACDP, and an action list is drawn up of tasks that will assist them in achieving their aims.

A formal process is defined and implemented in an iterative procedure, designed to gradually improve their working practices, and work towards achieving improvements in some of the Key Process Areas of the Capability Maturity Model. Reuse is targeted as a specific way to achieve efficiency within the development, and web accessibility is particularly important to CACDP as they work with many people who are affected by the lack of accessibility. The thesis describes the production of the applications using the defined process, and the problems faced during the implementation. These problems are reviewed and suggested improvements are integrated into the next implementation of the process.

This project has resulted in the successful introduction of a formal process for the development of web-based applications. Reuse is now being used within the company to reduce cost and improve productivity. Accessibility standards have been implemented in all products. CACDP have benefited from increased services for their customers, increased profitability, mature development and maintenance procedures, the introduction of a reuse programme for their future development and technical learning and training for their staff.

A Study of Process Improvement Activities for Web Development Processes within a Small Company

Joanna Mary Donkin

This thesis is submitted in candidature for the degree of Master of Science in the Department of Computer Science, University of Durham, 21st November 2002

Acknowledgements

For their help in the production of this thesis and the work contained within I would like to thank the following people:

From the Department of Computer Science, University of Durham – *Dr Elizabeth Burd, Dr Cornelia Boldyreff and Janet Lavery.*

From CACDP, Durham – *Wendy Watson, Celia Bauerfeind and Mike Holmes.*

For all their love, support and provision of chocolate when required – *Philippa Regan, Geoff Donkin and Keith Herrmann.*

I would like to dedicate this thesis to my mother, Barbara Donkin, who died on the 24th November 2001, and who I would have loved to have seen me through this, with all my gratitude for all her love and support through everything I have ever done.

Joanna Donkin
November 2002

Table of Contents

Table of Contents 1
Table of Figures 3
Acronyms and Definitions 6
Chapter 1: Introduction 7
1.1 Introduction and Background 7
1.2 Aim of Project 7
1.3 Research Areas 8
1.4 Research Problem 10
1.5 Criteria for Success 10
1.6 Organisation of Thesis 11
1.7 Chapter Summary 11
Chapter 2: Literature Survey 12
2.1 Introduction 12
2.2 Software Reuse 12
2.3 Software Process Improvement 24
2.4 Web Development 32
Chapter 3: Company Analysis 45
3.1 Chapter Abstract 45
3.2 Introduction to CACDP 45
3.3 Introduction to the Project 46
3.4 The Domain 46
3.5 Current Software Development Processes 47
3.6 Description of Current Web Development Processes 48
3.7 Formal Assessment of Development Process 49
3.8 Problems with Current Development Processes 49
3.9 CACDP Web Site at Start of Project 51
3.10 Action List 51
3.11 Summary of Chapter 53
Chapter 4: Experimentation 54
4.1 Chapter Abstract 54
4.2 Introduction 54
4.3 General Process Description 56
4.4 Proposed Product Development Process 56
4.5 Project One: Deaf Awareness Learning Package 58
4.6 Project Two: Directory of Human Aids to Communication 79

4.7 Project Three: Deafblind Awareness Learning Package 96
4.8 Project Four: Deaf Community and Culture Learning Package 101
4.9 Summary of Chapter 102
Chapter 5: Results And Evaluation 103
5.1 Chapter Abstract 103
5.2 Introduction 103
5.3 Product Evaluation 103
5.4 Reuse Measurements 105
5.5 Evaluation of Reuse Aspects 107
5.6 Process Improvement Measurements 108
5.7 Evaluation of Process Improvement Activities 109
5.8 Action List Revisited 111
5.9 Summary of Chapter 114
Chapter 6: Conclusions and Further Work 115
6.1 Conclusions 115
6.2 Further Work 121
6.3 Project Successes 122
References 123
Appendix A – Results of CMM Questionnaire 130
Appendix B – Domain Survey Results Report 134
Appendix C – Market Survey 149
Appendix D – Market Survey Results 152

Table of Figures

Figure 1 - Levels of Reuse 20
Figure 2 - Process for Development With Reuse 22
Figure 3 - Process Improvement Process 26
Figure 4 - Process Improvement Cycle 26
Figure 5 - A Generic Model of SPI 27
Figure 6 - The Five Levels of Software Process Maturity 28
Figure 7 - Layers in the Dexter Model 35
Figure 8 – Web Development Cycle 37
Figure 9 - Hypermedia Development Stages 39
Figure 10 - Waterfall Model for Software Engineering 40
Figure 11 - Example of Prototyping Process 41
Figure 12 – "J" 47
Figure 13 – "M" 47
Figure 14 – "D" 47
Figure 15 – Packages to be produced 55
Figure 16 - Process Improvement Flowchart 56
Figure 17 - The Waterfall Model 58
Figure 18 - Requirements Process 60
Figure 19 - Requirements Definition Process 66
Figure 20 - Waterfall Model with added Prototyping Phase 67
Figure 21 - Course Framework for Deaf Awareness Course 69
Figure 22 - Form Design for "Page" Table 71
Figure 23 – Initial Directory Page 72
Figure 24 - Improved Waterfall Process 80
Figure 25 - Find a Linguist Service 82
Figure 26 - Search for RID members 82
Figure 27 - NAJIT Search Form 83
Figure 28 - Requirements Process for Project Two 84
Figure 29 - UML Diagram for Directory 86
Figure 30 – Activity Diagram for the "New Registration" Process 87
Figure 31 - Activity Diagram for the "Locate_Interpreter" Activity 88
Figure 32 - Incremental Development Model 90
Figure 33 - Main Index Page 92
Figure 34 - Individual Interpreter Biography 93
Figure 35 - Reuse Process for Software Assets 98

Figure 36 - Course Framework for Deafblind Awareness Course 99
Figure 37 - Reuse in the Directory System (Project Two) 106
Figure 38 - Reuse in the Deafblind Awareness System (Project Three) 107
Figure 39 - Results of CMM Survey 109
Figure 40 – Reuse Between Systems 113

Declaration

This thesis has not been submitted for a degree at the University of Durham, or any other institution but it has been part published in the following papers:

J. Donkin, C. Boldyreff, E. Burd, and S. Marshall, "Supporting Sign Language Users of Web-Based Applications: A Feasibility Study," presented at Universal Access in Human-Computer Interaction, New Orleans, USA, 2001.

J. Donkin, C. Boldyreff, E. Burd, and S. Marshall, "The Case for the Use of Plain English to Increase Web Accessibility," presented at Web Site Evolution, Firenze, Italy, 2001

Acronyms and Definitions

ASL – American Sign Language

BDA – British Deaf Association

BSL – British Sign Language

CACDP – The Council for the Advancement of Communication with Deaf People

CGI – Common Gateway Interface

CMM – Capability Maturity Model

DWEB – Deaf Welfare Examining Board

ERM – Entity Relationship Model

HAC – Human Aid to Communication

HDM – Hypertext Design Model

HTML – Hypertext Mark-up Language

ISL – Irish Sign Language

IT – Information Technology

RDF – Resource Description Framework

RID – Registry for Interpreters of the Deaf

RMM – Relationship Management Model

SAO-IT – Senior Admin Officer – Information Technology

STTR – Speech-to-Text Reporting

UML – Unified Modeling Language

W3C – World Wide Web Consortium

WAI – Web Accessibility Initiative

XML – eXtensible Mark-up Language

Chapter 1: Introduction

1.1 Introduction and Background

"The World Wide Web is like an encyclopedia, a telephone directory, a record collection, a video shop, and Speakers' Corner all rolled into one and accessible through any computer" [1].

Everybody wants to be "on the web". There is a rush to enter cyberspace that no commercial organisation can ignore and as a result many small businesses start with web sites that are little more than brochure-ware – i.e. an advertisement for their business but with no interactivity or functions for the user. Now businesses are starting to want more complex and more interactive sites for use by their customers, as well as being an integral part of the business functions. As web sites get more complex so do their development and maintenance needs. For example, a small site consisting of a few static pages will be able to be maintained by a single person with little need for documentation or records. However, if a site gets more complicated, or larger, or there is a chance that someone else may be called upon to make changes to it, then there is a need for a documented development process.

1.2 Aim of Project

In this project a small company is used as a case study. This company (CACDP) is described further in Chapter 3. They are at a point where they have no documented processes for the development and maintenance of their web site and are experiencing problems both in the maintenance of their current web site, and in the development and expansion of new web based products. The aim of the project is to document their current processes and problems, and to help to solve these problems and improve the processes they use as well as developing new ones. This will be done by first surveying the company practices and determining how to set about introducing development methods. Then, products will be developed using the proposed methods, with ongoing review of the processes in order to iteratively improve them as further products are developed. Finally, staff will be trained in using these new processes and thus allow the organisation to continue to develop new products in a methodical manner.

1.3 Research Areas

There are three identified research areas: Web Engineering and Development; Software Process Improvement; and Software Reuse. These are introduced below, with a detailed Literature Survey for each area in Chapter 2. These areas have been chosen because they will provide considerable benefits to the company if they are introduced effectively. A formalised Web Engineering procedure will allow them to expand their range of products and services, and will enable them to maintain these new products more efficiently. To get to the stage where they have a formalised development process, they will need to make use of Process Improvement strategies as it will be impossible to introduce a very formal process immediately because the organisational structures to support it will not exist. Introduction of a Reuse strategy in conjunction with a formal process will make their development activities more efficient as they can make use of work already completed, and it will start to introduce quality standards to the development process.

1.3.1 Web Engineering and Development

Web Engineering is exactly what it suggests, engineering for the World Wide Web. A more formal definition is *"The Employment of a systematic approach to the development, operation and maintenance of hypermedia applications"* [2]. Similar problems are currently being faced in the world of web engineering, as have been evident in Software Engineering for many years. The problems include lack of processes, ad hoc development and lack of maintenance activities. The large, complicated systems that were developed using these methods are now becoming difficult to expand and to maintain so that they can continue to be useful [3]. Web Engineering seeks to improve these problems by introducing sound scientific processes, and a disciplined and systematic approach to development and maintenance [3].

The development of web-based systems has followed a similar path to that of standard software development. Enthusiasts, usually individuals or small teams, who gave no thought to planning, maintenance activities or expansion of the site, generally produced the first systems. As technology improved, and websites became more than simple documents, website developers were finding that websites were becoming problematic, in the same manner that legacy software caused problems years earlier. Legacy applications traditionally were found to be in difficulty because they had no documentation or what did exist was out of date, the original developers were no longer available, and the technology had moved on so far that expansion was a problem.

These days, websites consist of many different types of technology, and there are many different models, methods and patterns for producing them. Website developers need to be proficient in many of these. These methods often have poor development environments, are not extensible to take new technology into account, and cannot be linked together to produce a whole site [4]. Web Engineering was introduced to attempt to bring an element of systematic processes and sound engineering and management principles to the development of web-based applications [3].

1.3.2 Software Process Improvement

Process Improvement can be formally defined as "understanding existing processes and changing these processes to improve product quality and/or reduce costs and development time" [5]. Process improvement is an important part of any procedure, whether it is manufacturing, software development or administration. The analysis and breakdown of a process in order to determine any bottleneck areas, or any areas which can be made more efficient, is a very good way for an organisation to make changes and improvements to their working practices. Organisations should make use of measurements and metrics in order to help them identify where improvements can be made. The general activities carried out within any specific process improvement model are Analysis of Current Process, Identification of Potential Improvements and Implementation of Improved Process. These models tend to work on an iterative pattern, leading to continuous improvement. The models are iterated at whatever rate the organisation wishes, to allow more rapid or slower rates of change.

The benefits of process improvement include increased quality of both process and product [5], enhanced time-to-market, and increased efficiency of procedures leading to reduced cost-to-market of the products. Some of the barriers include a lack of understanding and support from management, a lack of resources allocated to projects [6], and increased costs from setting up the projects in the first place [7].

A software process improvement programme can be crucial to the success of an organisation if it is required by the customers, or if it is highly desirable within the industry. In any case, it can help to improve the quality and efficiency of any process and product.

1.3.3 Software Reuse

A formal definition of software reuse is "the systematic process of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity,

quality and business performance" [8]. This is generally known as "development with reuse". Software reuse can also include the activities of developing software, which is designed from the start to be suitable for reuse, this is known as "development for reuse". Benefits of reuse activities include more efficient development due to a reduced time requirement for both development of components, and testing as the "ready-made" components should have a history and test cases. There are costs associated with reuse as components may take longer to develop initially, and a library or repository may need to be set up to deal with the components.

Many small, and some larger, companies already operate some form of informal reuse processes. Each time a developer makes use of a procedure or function that was developed for a previous application – that is reuse at its most basic level.

1.4 Research Problem

Much research has been done into methods and models for all three of the above domains – software process improvement, reuse and web development. The procedures proposed by academics tend to be very complex and difficult to implement and as a result these models are not all suitable for putting into practice in companies that are not "tech-savvy", or that do not have the time or skill-base to implement complex theoretical solutions. These types of companies will not put procedures in place if they cannot understand the theories, or do not realize that these types of procedures exist and can be applicable to them.

In addition, small companies tend to either not have any development processes, or they are very immature and difficult to document and thus improve. They use methods that work for them, and that they have always used. This makes it very difficult to encourage them to change and to put improvements into place, as they cannot see a need for changes in what they do.

1.5 Criteria for Success

The success criteria for the research project are:

SC1: Identify state of the art procedures for software development and investigate the feasibility of adoption for small companies

SC2: Carry out a study of the software development processes and procedures of a small company and document the current activities with a view to improving these procedures

SC3: Define a development process for web based products

SC4: Develop commercial products using the process defined above

SC5: Improve the defined process by the application of Process Improvement techniques

SC6: Introduce the concept of reuse to the company in order to make their procedures more efficient

SC7: Enhance the knowledge of the organisation with regard to web development, development procedures with reuse, and process improvement

SC8: Introduce guidelines for web Accessibility to ensure that all products, new and existing, are suitable for use by all potential users.

1.6 Organisation of Thesis

This chapter contains a general introduction to the area of research, and the objectives and success criteria of the work. Chapter 2 contains the survey of all the background literature in the three areas identified above in section 1.3. Chapter 3 covers the case study company, CACDP, and provides a survey of the state of the current processes and web site at the start of the project. Chapter 4 covers the main experimentation process. Chapter 5 contains the results and an evaluation of the experiment and the processes used. Chapter 6 covers conclusions of the experiments, and possible further work. Chapter 7 contains the references cited in the text. Finally, there are appendices containing extra documentation that may be of interest.

1.7 Chapter Summary

This chapter provided an introduction to the project, and a general introduction to the three main areas of research; Web Engineering, Reuse and Software Process Improvement. It covered the objectives of the research and the criteria for success. Finally, the organisation of the thesis is explained.

Chapter 2: Literature Survey

2.1 Introduction

This chapter contains a background survey of the three identified research areas listed in Chapter 1. These are *Software Reuse, Software Process Improvement* and *Web Development*. Each area is defined and explored in the following sections.

2.2 Software Reuse

2.2.1 Introduction

Reuse, in the context of software engineering, can be defined as "*the process of creating software systems from existing software systems, rather than building software systems from scratch*" [9]. It has also been defined more fully as "*the systematic process of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity, quality and business performance*" [8].

Most traditional engineering disciplines make far more use of reusable components than Software Engineering does at present. Mechanical and Electrical engineers, for example, do not design systems which require the majority of parts to be produced specifically for that system. They base the design around existing tried and tested components, everything from nuts and bolts to entire engines [5]. Software designers have not yet made the same advances, despite there being some 100 billion lines of code in existence [10]. The concept of reuse is explained more fully in section 2.2.2, and a description of the types of components and products, which can be reused, can be found in section 2.2.3.

There are many benefits to reusing components of systems. These include more efficiency in the use of time, money and skills, and also more reliable and better quality software [5]. The benefits of reuse are explained more fully in section 2.2.6. These technical benefits will lead to associated business and economic benefits, which are explained in section 2.2.6.3.

There are different types of reuse, and different levels at which a company can operate within the reuse framework. These are explained in sections 2.2.4 and 2.2.8

respectively. There are also several different activities associated with implementing a reuse programme; these are covered in section 2.2.9.

2.2.2 What Is Software Reuse?

As stated in section 2.2.1, software reuse at its most basic level consists of making use of any existing information, artifact or product when designing and implementing a new system or product.

There are differing opinions as to which activities constitute genuine software reuse. For example, using a library routine for calculating square roots in several different programs is a good example of successful reuse, but calling that same routine many times within the same program does not technically constitute reuse [9]. Replication of an entire software program does not count as reuse. Reuse of assets is dependent upon both similarities and differences between the applications in which the component is being used [8].

Many organisations already practice a limited form of reuse, for example, most developers have libraries of components that they have developed in previous projects, or they use standard libraries that are available with many programming languages [11]. About 30% of most code is developed with this type of reusable component where the developer is simply using his own work from previous projects [12]. This is a very ad-hoc method of reuse, and while it will work very well on a small scale, for example an individual programmer or a small team, it will not be suitable for entire organisations [13]. Instead, businesses need to implement a systematic reuse program in order to gain the full advantages of reuse. Organisations need to have mature processes, which are used throughout the company and metrics to measure the success of the reuse process. They need to understand how reuse can benefit the whole organisation and how they need to build a reuse program into their way of working.

There are four 'truisms' identified with software reuse [9]:

1. For a software reuse technique to be effective, it must reduce the cognitive distance between the initial concept of a system, and its final executable implementation
2. For a software reuse technique to be effective, it must be easier to reuse the artifacts than it is to develop the software from scratch
3. To select an artifact for use, you must know what it does

4. To reuse a software artifact effectively, you must be able to "find it" faster than you could "build it"

These 'truisms' make common sense. In order for a programme to be effective, developers must see a marked improvement in efficiency or development time. Otherwise, they will continue to do as they have always done and the new system will not take up.

2.2.3 What Can Be Reused?

The definition of a reusable component is *"any component that is specifically developed to be used, and is actually used, in more than one context"* [11]. This does not just include code; other products from the system lifecycle can also be reused, such as specifications and designs [5], and even requirements on occasion [14]. 'Components' in this case can be taken to include all potentially reusable products of the system lifecycle, including code, documentation, design, requirements etc.

Some components are more difficult to reuse than others. For example, code fragments are relatively easy to reuse, providing they have the appropriate documentation, and can be modified as needed. Other components, for example, requirements specifications are more difficult to reuse as they tend to be produced in Natural Language i.e. English, which makes them informal and leads to ambiguity and a lack of structure [15].

There are various criteria that should be satisfied in order for an asset to be successfully reusable. These are grouped into General, Functional and Technical requirements [8]. General requirements focus on aspects such as compliance with relevant standards, completeness, modularity and simplicity. All components should conform to these general requirements. Functional requirements include such concerns as which business processes it will simulate or automate, and how well it does this. Functional requirements mainly concern Vertical or Domain-specific assets (explained further in section 2.2.4) and tend to be very specific to each information domain. Lastly, Technical requirements refer to criteria such as interoperability, portability, communication and security [8].

There are different levels of reuse that can be considered [5]. At the highest level, entire applications can be reused on different platforms provided they are portable. Sub-systems can be reused within different applications, possibly within different domains; for example, a login system could be used in a database application as well

as a control application. At a lower level modules or objects can be reused, and at the very lowest level single functions can be reused. This is also known as classification of the granularity of components. Fine grained is used to describe those smaller and more generic components, for example file access functions, or I/O functions. Course grained is used for the more complex components, for example user-interface packages [11].

Reusable assets can be built in-house, retrieved from legacy systems or can be bought from an external source [16]. Many components are available free of charge from Universities or non-profit-making organisations as it is very difficult to make a profit selling generic components [11]. The provision of a repository of components is something which must be considered before a company introduces a reuse programme as components may need to be adapted before they can be put into the repository [16].

In the context of Web Applications, there are various different ways to achieve reuse. These include, for example, reusing templates for the interface, or information from shared databases. Sometimes much larger components can be reused, for example, code to implement a shopping basket in e-commerce applications [17].

2.2.4 Types Of Reuse

Three types of software reuse have been identified – Vertical or Domain reuse; Horizontal or General reuse; and Product-line reuse. Vertical reuse occurs when a component is specific to an application domain, for example assets which capture the business knowledge such as the design and code for a customer management model, or financial object models. These can be reused in other applications within the same domain [8]. A domain can be defined as "*a sphere of control or influence*" [18] but is more generally known in Computer Science as an area of knowledge or information. Vertical reuse will not necessarily work within all domains, for example, it will be difficult to achieve successful reuse for domains which have time and space constraints, such as where the business model changes depending on external conditions. Some companies have attempted to turn domain-specific reuse into a marketable opportunity by carrying out domain analysis and from this, developing components to sell, but not many have succeeded so most successful programmes are internal [11].

The reuse of Horizontal assets, sometimes called General Reuse, concerns those components that can be reused within other application domains, for example user interfaces or database connection libraries [8]. Components tend to be abstract data

types with common behaviour throughout different domains that are not reliant on specific business processes or information [11].

2.2.5 Why Is Reuse Not Very Common?

There are many reasons why reuse is not very common in many organisations. Many managers cannot be convinced of the benefits of reuse until they are demonstrated in real projects, which of course cannot be done until it is introduced in the organisation [5]. There may be problems with starting to collate resources for the reuse repository, and with the categorising and indexing of them [19]. Managers may also feel that a reuse programme may lead to a loss of budget for them, due to potentially higher levels of productivity from the reduction in development and testing time required. Sometimes they may feel that a failure in the programme would involve their budget being used to deal with the problem [16]. Some managers may feel they will incur a loss of status if the organisational structure changes, or they may feel that their teams do not have enough time to instigate changes to the development process [9].

There are other considerations on the programmers side, for example, the *not-invented-here* issues where the developers may feel that they cannot trust the quality of the components, or a fear that the measurements being brought in to determine the success of the reuse programme, may be used to measure their performance as well [16]. Some programmers may enjoy the process of component development more than that of component integration, and may reject the programme on these grounds. Sometimes programmers feel that "copying" the work of others is always wrong, and this can lead to a culture of not embracing reuse within an organisation [9].

2.2.6 Benefits Of Software Reuse

There are several potential benefits to an organisation of implementing a successful reuse process. The main technical benefits include improvement of the productivity of the development team as a result of a reduced need for development and for testing, and increasing the quality of the software products produced as a result of this [20], leading to business related benefits including reduced costs, reduced time-to-market and better customer service. The software produced will be more reliable because the components used will have been tested more rigorously and in real-life situations.

2.2.6.1 Improving Productivity

The first main aim of reuse is to improve the productivity of the developers [20]. This is achieved because of the reduced development time, due to the fact that the components have already been built. The components have also already been tested,

although not yet within the new system, but potentially within a number of existing systems if they have been reused several times [5]. In order for a component to be suitable for reuse, it must have appropriate documentation and have been tested thoroughly, and ideally should also have been certified to confirm that quality standards have been met. Productivity is increased and the development time reduced which helps with the ever-decreasing time-to-market or Lead Time factor which is becoming the most important factor in the highly competitive and rapidly changing technology market [11]. Businesses can capitalise on this reduced time-to-market by becoming more competitive and providing a better service for their customers. They can produce more products in a shorter time, or they can provide better support for their existing products. Reuse will help to reduce both the development and the maintenance time required for a product [21].

There can be a cost associated with reuse, that is of developing the components the first time (Development *for* reuse), as this may take longer than usual due to the more complex nature of the product to abstract it and provide a reusable component, but this should be seen as an investment for the company [11]. There will also be a cost involved in setting up and maintaining the reuse repository, for staffing and the procurement of resources. The procurement of resources, however, can in itself be a hindrance to the implementation of a reuse program as either developers don't have time to develop components in such a way that they can be reused, or there are very few components available at the start within the organisation to successfully introduce reuse [16]. A well-stocked repository of reusable assets is a company asset.

2.2.6.2 Improving Software Quality

The quality of the software produced is increased by the introduction of a reuse programme because the components used will have already been tested in 'live' systems, and, more importantly, in a variety of situations if they have been used in other projects, and therefore any problems should have been rectified already [13]. The reuse of components also improves the reliability of the system for the same reasons. The reused components have been tested in a variety of realistic operating conditions and therefore should be more reliable [5].

Components can also be used for prototyping, which will help the client to clarify their requirements and thus reduce potentially costly changes later in the development process [13].

The introduction of a reuse programme will also help to implement quality standards within software development. Not only will reusable components introduce standards by default, as all the components must conform to these standards before they become part of the repository, but also it is easier to enforce the use of components that already satisfy the standards than it is to check new components for compliance [21]. The introduction of standards into a development process will help to improve the quality of the software because the developers will be able to verify the quality of the components.

2.2.6.3 Business Benefits

The organisation will benefit both economically and otherwise from a reuse programme. The improvements in productivity will lead to quicker production of products and components, and also to higher profits or cheaper products. This will give them a shorter time-to-market which will improve the competitiveness of the organisation within its own domain. The improvements in product quality will lead to better service for the customers and ultimately a better image for the business. Other, less obvious benefits include the chance of both direct and indirect new business opportunities as the components themselves can be sold, or new markets can be identified possibly using components of the older systems [21].

Economic benefits will include saving money on the development of new products, saving money on the productivity of the development team, and reducing the time-to-market of a new product.

2.2.7 Costs Of Software Reuse

There are both financial and organisational costs of introducing a software reuse programme into an organisation.

2.2.7.1 Financial Costs

Financial costs can be seen mainly in the set-up costs for such a programme. In order to successfully introduce reuse, a comprehensive repository of components must be built up. There are three ways in which this can be done [22]:

1. Components can be produced specifically to be reused
2. Components can be derived from legacy systems
3. Components can be produced as part of new developments

Each of these methods carries a cost along with it. Developers may need to spend more time and effort developing new components, or they may need an incentive to produce suitable components [16].

Other than the populating of the repository, other set-up costs could include the training of staff for the new programme, the need for new staff to maintain the repository, or the setting up of measurement procedures to gauge the effectiveness of the new reuse programme.

2.2.7.2 Organisational Costs

There can be problems when introducing a new strategic change into an organisation. These problems can manifest themselves at both the lower, e.g. ground level programmers and developers, and upper e.g. management, levels [16]. Development teams can find themselves with more work and no more resources, as they are required to produce reusable assets within existing projects in order to populate the repository. This can lead to a reduced productivity at the initial stages. Managers can often be unreceptive to the idea of a reuse programme as they may feel that they do not have time to implement it, or that the metrics used to measure it will also be measuring their own performance [16].

2.2.8 Levels Of Reuse

There are typically four levels of reuse that a company can be achieving. Which level an organisation is working at can indicate how mature their reuse processes are and what can be done to improve them, leading to the benefits listed in Section 2.2.6.

The four levels of reuse [13] are:

1. Ad Hoc – at this level engineers tend to reuse their own libraries and code fragments when they notice a similarity between the projects. Most organisations practice this level of reuse, even if they are not aware of it [13]. There is no systematic reuse plan, and sharing of resources tends to occur accidentally or informally [16].
2. Systematic Reuse – this is driven by a well-ordered process and co-ordinated planning. Resources are allocated specifically for a reuse program and it is fully supported by the management [13].
3. Domain-Oriented Reuse – at this level the company focuses on creating a library of reusable assets within the existing and future business domains. This allows the organisation to build up a repository of reusable components and therefore to be more competitive [13].

4. Strategy-Driven (or Cultural) Reuse – this is the highest level of reuse. Reuse helps to define the development of new products and processes. For example, a manager may ask, "how can I diversify in such a way which allows me to reuse as much as I can of the products we already produce?" [13]. Reuse is accepted as part of the process of development, and should no longer require targets or incentives [16].

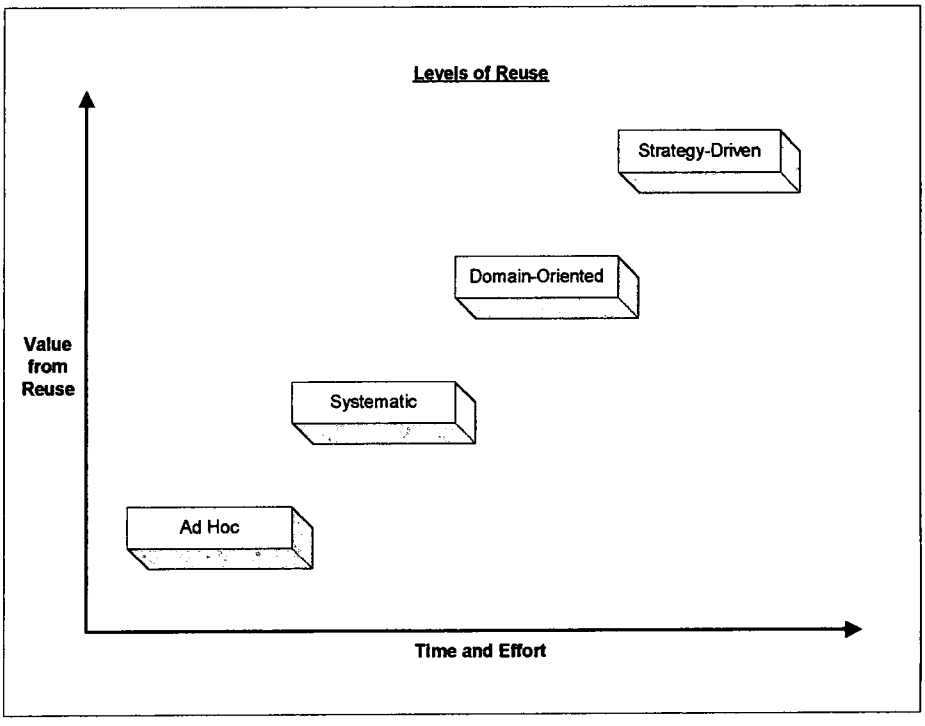


Figure 1 - Levels of Reuse

2.2.9 Reuse Activities

There are two main reuse activities – Development *For* Reuse and Development *With* Reuse [11]. The difference between the two activities is that development *for* reuse occurs when new components are designed specifically in order to be reused, and development *with* reuse occurs when existing components are made use of within a new system. Ideally the two activities should be carried out in conjunction with each other to create an environment of reuse within a company. Early introduction of development for reuse can help to encourage the reuse driven development, as there will be a large store of products available for reuse within the company, and ideally developers will start to find it easier to reuse a component rather than design it from scratch.

2.2.9.1 Development With Reuse

Development with reuse, or reuse driven development, is the activity of reusing components within a new system. It has been defined as "*the search for, evaluation, adaptation and integration of existing components in a new context*" [11]. Developers must consider the use of existing components at the design stage of the process rather than the implementation, as it is in other traditional engineering disciplines, for example, mechanical engineers design new engines from a stock of parts rather than designing the engine and then checking if any existing parts will be suitable [5].

Components should be thoroughly investigated to ensure that they suit the requirements and that they have appropriate test records and development documentation [11]. Components need to be fully documented to facilitate easy searching, evaluation of the component, investigation, adaptation and integration. This means that a component must have three types of documentation: Engineering Documentation, Product Documentation and Maintenance Documentation. These will cover all aspects of the component from the development process through to the future evolution and maintenance [11].

To carry out reuse driven development successfully there are three conditions that an organisation must fulfill [5]. They must:

1. Have sufficient resources which are easy to find and catalogue;
2. Have introduced standards which will ensure confidence in the components behaviour
3. Have associated documentation for the use of the user to help them understand the component

The process described in Figure 2 [11] will allow the successful identification and adaptation of a component into a software system. First the requirements for the component must be identified, and a search made for existing components that satisfy these requirements. When a component has been located, it must be understood and potential adaptations noted in order for it to be easily compared with others. Once a suitable component has been selected from those located by the search, it should be adapted as necessary and integrated into the system. Finally, a report should be written on the adapted component in order to assist future users of the product. [11].

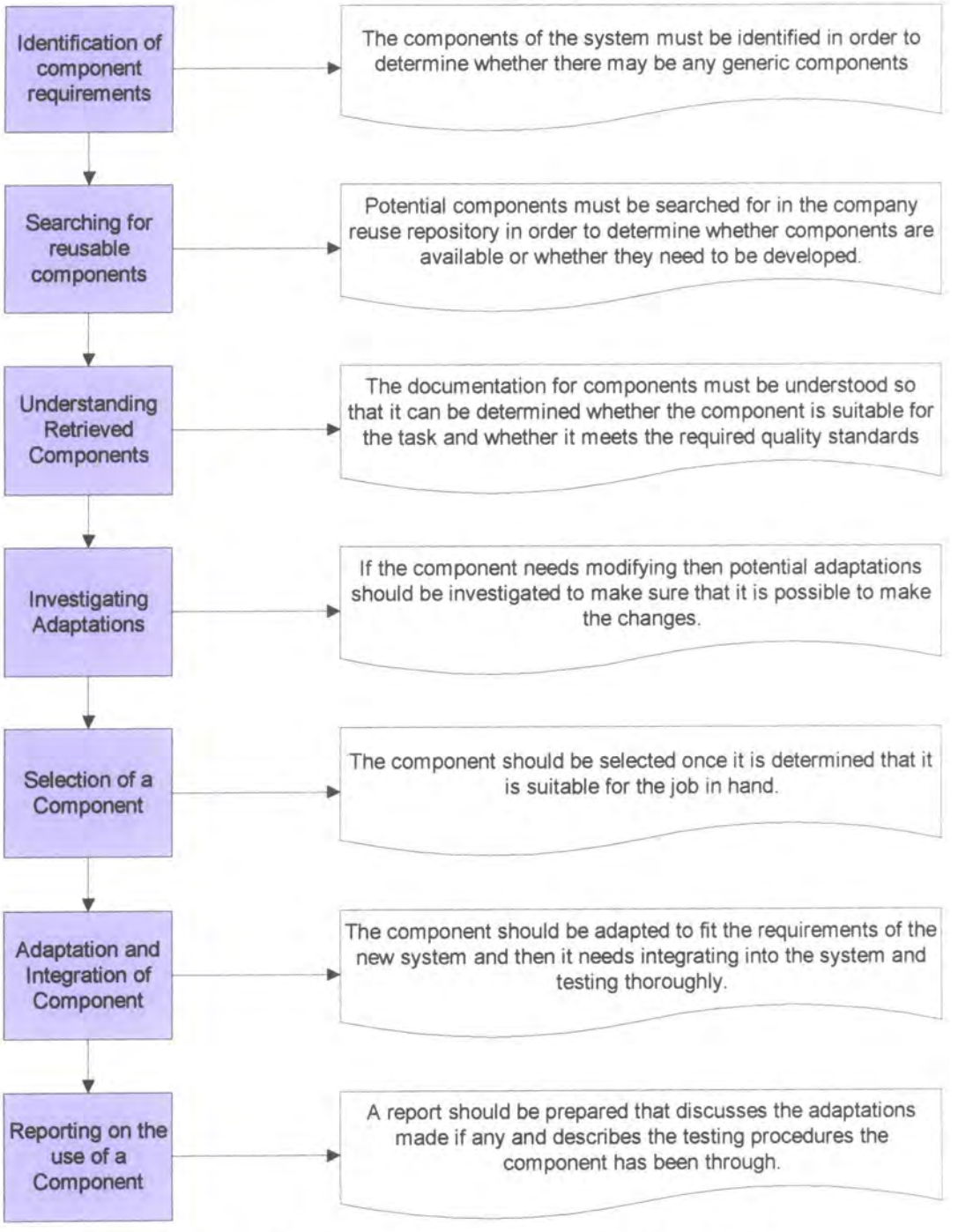


Figure 2 - Process for Development With Reuse

2.2.9.2 Development For Reuse

Development for reuse is the activity of designing new components with reuse in mind. It has been defined as *"the planned activity of constructing a component for reuse in contexts other than the one for which it was initially intended"* [11]. This makes components more difficult to design as the generic nature must be taken into account, and possibly components can be originally designed to include excess functions which are not needed immediately but will improve the marketability of the product later on.

2.2.10 Web Reuse

Various procedures have been proposed for reuse within the domain of web applications and software. Web Design Frameworks are a solution to the lack of support for reuse of design structures for web applications [17]. It is proposed that, for example, the activities which occur when a customer purchases an item in a online store does not change for different stores and therefore could be reused, with only the specific aspects needing designing [17]. A Web Design Framework is defined as "a generic design of possible Web application architectures, including conceptual, navigational and interface aspects, in a given domain" [17].

Another way to improve web maintenance by reuse is by separating the contents and the page structure [23]. This approach supports multilinguality, that is a web site can be presented in different languages without redoing the development work. Generic aspects of the page design can be specified in a similar way to templates in order to give a consistent look to a site, and to make maintenance of the aesthetics of the site very simple [23].

All attempts to introduce reuse into web engineering are focused on reducing the time and effort it takes to design and create a web application. Many are written in several languages and contain a large amount of repetitive work. If this could be reduced then sites could be created more quickly and could be maintained more easily [23].

2.2.11 Summary

There are many benefits and a few costs associated with the introduction of a reuse programme into an organisation. There are the obvious direct costs for the set-up and maintenance of the software repository as well as the increased development costs to initially develop reusable components. However, these are far outweighed by the benefits which include enhanced productivity of developers leading to reduced development costs, reduced maintenance and testing costs due to the increased reliability of the software components, and therefore better quality software. The business benefits include reduced time-to-market, increased software quality and therefore a better customer image and improved service to consumers.

The majority of components from the system lifecycle can be reused, some more easily and more successfully than others.

2.3 Software Process Improvement

2.3.1 Introduction

The act of Process Improvement is "understanding existing processes and changing these processes to improve product quality and/or reduce costs and development time" [5]. This means that organisations need to define their current processes and then review them in working practice using measurements and metrics. These metrics are then used to propose improvements to the current processes, and, as a result of the metrics, a new or improved process is put into practice. The same metrics can then be used to determine the level (if any) of improvement after the changes [24]. Process Improvement can be applied to any type of process in order to make it more efficient, from making a cup of tea to building complex software. There has been much work done on process improvement activities in the world of hardware and physical engineering. In this domain, there are only finite solutions, which makes it much easier to measure and to define, whereas software is more difficult as it is linked to the skills and imagination of individual programmers [25].

The need for software process development was first identified in the mid 1980s when the Software Engineering Institute (SEI) was established by the US Department of Defence as an attempt to reduce the amount of software that was developed over-budget and over-schedule [26].

There is a strong relationship between the quality of a software process, and the quality of the software produced by the use of that process [5] and this makes process improvement particularly important.

Software Process Improvement (SPI) is lagging behind the rest of the engineering disciplines such as mechanical or civil engineering because although many organisations have spent time and money improving their hardware engineering, few have put the same effort into software [25]. There are several models that have been proposed to help organisations to formulate methods for Software Process Improvement. These have been described in Section 2.3.4.

2.3.2 Why Do We Need Formalised Processes?

Processes and models are needed in the Software Domain in order to allow developers to understand development activities, and to manage the development of systems so that they are more comprehensible, easier to maintain, and more effective and efficient

for organisations [27]. Organisations also need to be able to manage the design and development of systems in a systematic and repeatable manner as this will aid in, for instance, the reuse of components as well as allowing developers to follow a clearly defined series of steps [28], leading to better products and more efficient working practices. Many companies are having problems trying to apply process improvement techniques to the very ad-hoc and informal methods used for software development, particularly in the web domain [25] as the current procedures cannot be formalised enough to apply the models to them.

Sometimes it can seem that small companies have less need for sophisticated processes than the larger companies, but such projects often have more specific requirements and closer contact with their customers [29]. Some of the SPI methodologies can be tailored and/or simplified to take into account the different requirements of small companies [30].

2.3.3 How Is SPI Carried Out?

There are a number of activities and stages involved in a generic SPI model. It starts with *Process Analysis*. This involves a thorough examination of the existing process in order to determine what is happening at the moment. Next is the *Improvement Identification* stage where the results of the analysis are used to identify potential improvements to be made to the process. These changes should focus on bottlenecks in the system. The third step is the *Process Change Introduction*. Here, the new procedures and tools are put into place and integrated with the existing procedures. Fourth is the *Process Change Training* where the managers and engineers are trained in the new procedures and tools in order to maximise their chances of being accepted with maximum effect. Finally, there is the *Change Tuning* where minor problems in the new procedures are ironed out [5]. The process can be iterated in order to make it easy to assess the effect of each change on the system, rather than making many changes at once making assessment of the effect of individual changes impossible [31].

Figure 3 shows a generic process for improving a procedure or process [5]. The boxes identify documentation that should be produced during the process, and the ovals show the activities that should be carried out. All documents and activities should be closely controlled and should be under Configuration Management in order for the organisation to be able to implement quality and other standards [32].

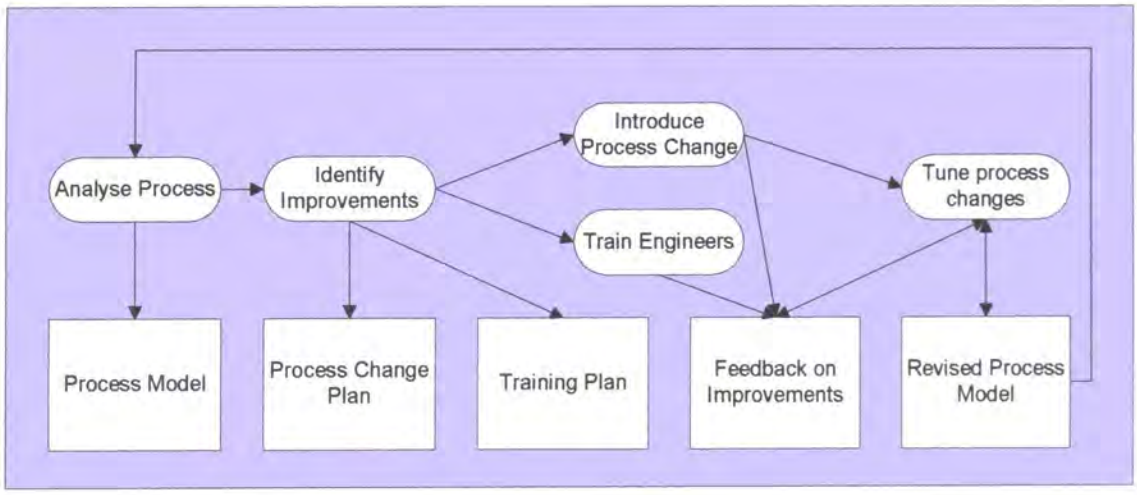


Figure 3 - Process Improvement Process

Figure 4 shows a different process improvement cycle that makes use of a generic improvement model and a specific process model inside it [6]. This model is cyclic and should be repeated continuously, hopefully leading to a continuous improvement in standards. As before, the process improvement is iterative.

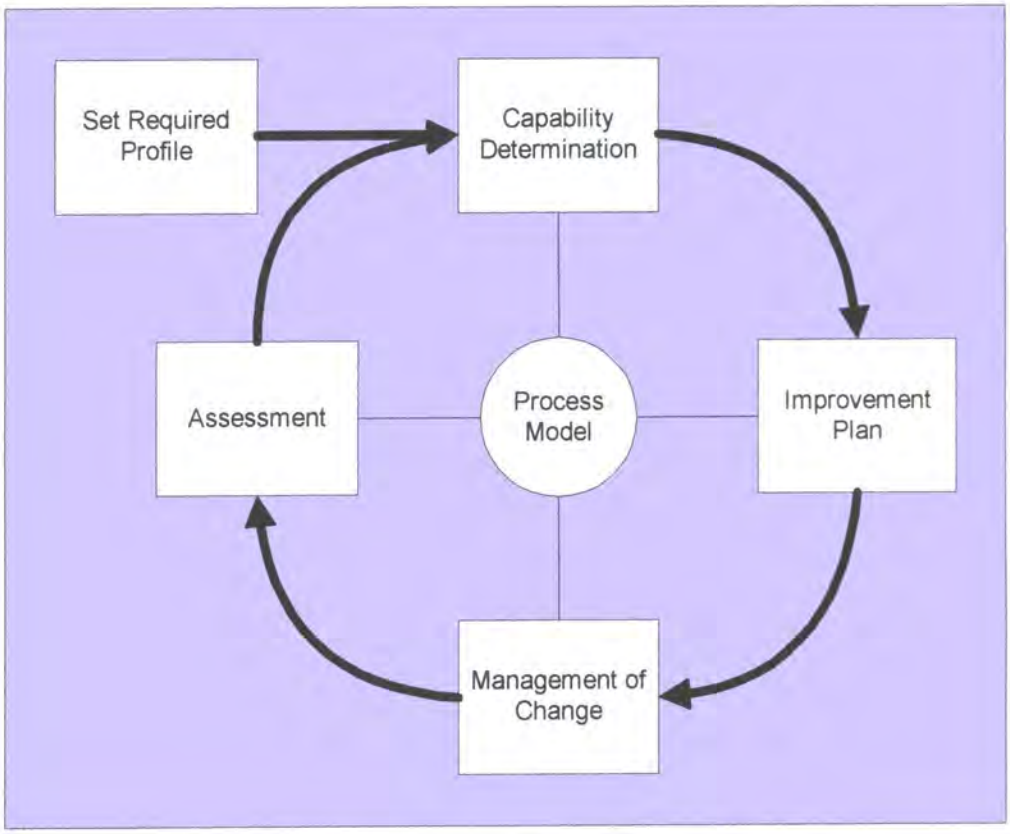


Figure 4 - Process Improvement Cycle

These two models can both be reduced, in the simplest terms to a cycle of analysis and change, followed by further analysis and change, similar to that shown in Figure 5 [24]. It should be noted "*Process Improvement is a long term, iterative process*" [5].

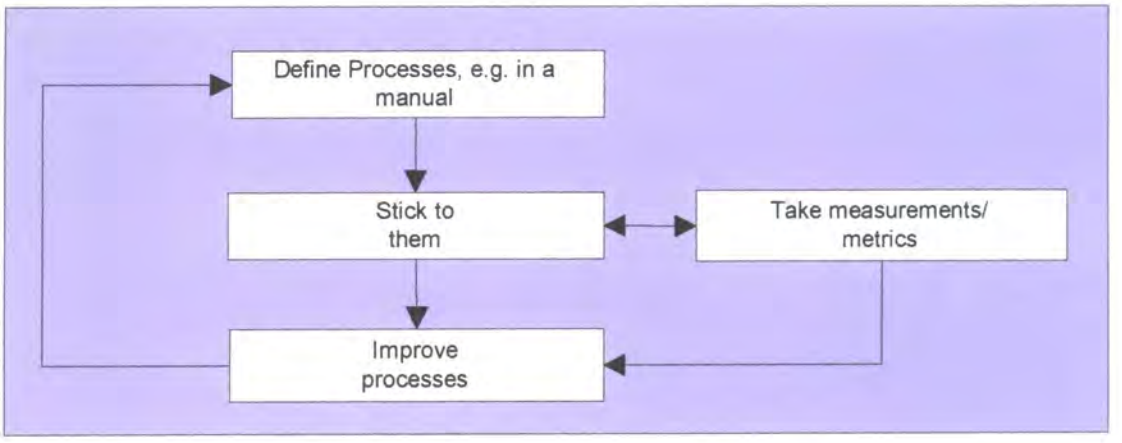


Figure 5 - A Generic Model of SPI

2.3.4 Specific SPI Processes and Models

Below are brief descriptions of several well-known process improvement models and methods. These are different ways that an organisation can carry out the SPI programme. All of the methods have stages in common, and these match up to the generic models. All methods have an *Analysis* phase, an *Improvement* phase, and a *Feedback* phase. All the methods iterate or repeat to allow continuous improvement.

2.3.4.1 The Capability Maturity Model

The Capability Maturity Model (CMM) was devised by the Software Engineering Institute (SEI) in response to the software crisis in the late 1970's [33]. It has been widely adopted amongst larger businesses including Boeing, IBM, Motorola, Siemens, the US Army, Navy and Air Force, and the United Space Alliance Space Shuttle Onboard Software Project. In a list held by the SEI in May 2002, there were 72 level 4 organisations, and 69 level 5 organisations [34].

The CMM identifies which of five hierarchical Maturity Levels a company fits into, with an ultimate aim of reaching level 5 for larger organisations. Some smaller organisations aim lower as it can be very expensive to attempt to reach the higher levels. The five levels are shown in Figure 6. These levels consist of a set of process goals or *Key Process Areas* (KPAs) which, when achieved, indicate that the organisation has stabilised one component of the Maturity Framework. The levels each establish a different component of this framework [35], and lay down a foundation for continuous process improvement [36]. The CMM focuses less on individual practices, and more on the organisational activities as a whole [37].

The five maturity levels are [5, 35]:

1. *Initial Level* – At this level an organisation has few if any formal procedures or plans, and no mechanisms to ensure that procedures that do exist are carried out properly. Both the software process and the software product are unpredictable.
2. *Repeatable Level* – At this level the organisation can successfully repeat projects, but still lacks a formal process model. Motivation is based on individuals and on organisational culture.
3. *Defined Level* – The organisation has a formal process defined, and has put into place procedures to ensure that this is followed in all software projects.
4. *Managed Level* – Metrics are introduced into the process and a formal method of data collection is defined. These measurements are used to determine process improvement activities.
5. *Optimising Level* – An organisation has committed to a formal process improvement schedule, and this is considered to be a part of the overall organisational culture.

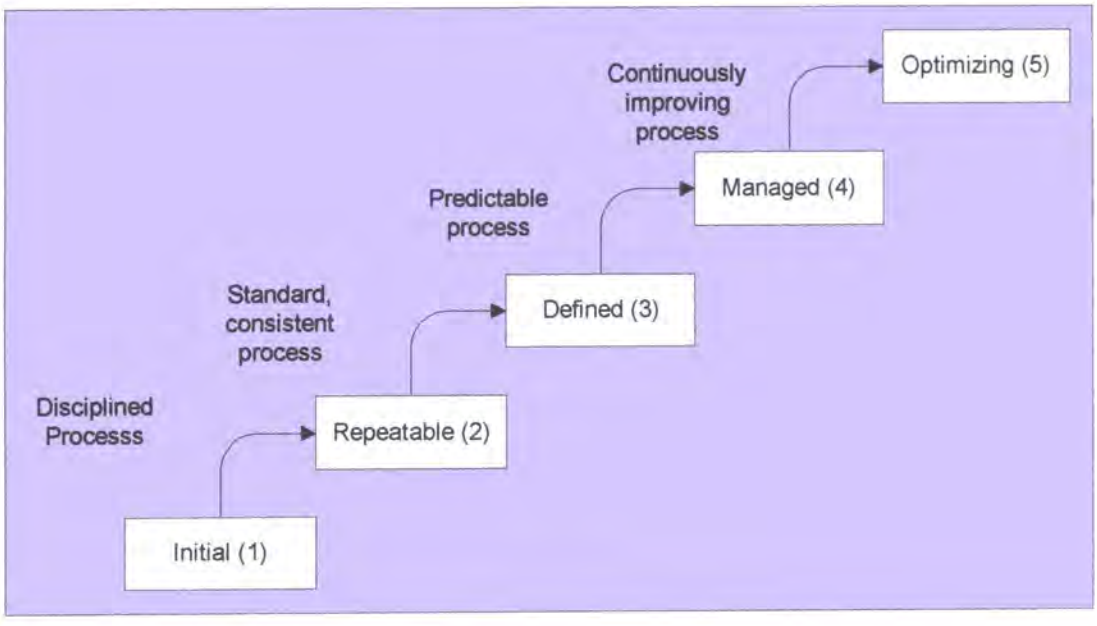


Figure 6 - The Five Levels of Software Process Maturity

CMM is based on the concept of mature and immature software organisations. The immature organisation works on a reactionary basis, dealing with immediate crisis; regularly exceeds budgets and schedules due to uninformed estimates; and often compromises quality and functionality in order to meet a deadline. In contrast, mature organisations have planned software processes which are communicated organisation-wide, and which are updated regularly. Cost and schedule analysis take place throughout the project and all roles and responsibilities are well defined [35].

2.3.4.2 ISO 9001 - Standards for Quality Management Systems

The ISO 9000 series are documents that deal with the standards for quality systems. Organisations can use these documents to make an assessment of the quality standards of their processes and products. They can also be used as certification that a process or product meets certain criteria [36]. There is no procedure diagram to follow, simply a set of documents to complete, as in the CMM, but an organisation will meet the standard or not, rather than having levels to work through.

2.3.4.3 The SPICE Project

SPICE stands for Software Process Improvement and Capability dEtermination. The SPICE project was created as a result of two key initiatives undertaken by the UK and US military in the 1980s, to improve the selection mechanism for software contractors in order to reduce the risks associated with software projects, and to improve the quality of the software produced [26]. The SPICE project was born out of a need for a generic industry wide solution to this problem, rather than a defence-specific solution.

The goals of the SPICE project were to develop an assessment standard that would [26]:

- Be applicable to both process improvement and capability determination
- Be applicable to different application domains, business needs, and sizes of organisations
- Not presume specific organisational structures, management philosophies, software life cycle models, software technologies, or software development methods
- Use objective and, where possible, quantitative criteria
- Produce output in the form of profiles rather than a single number or pass/fail result and support comparisons with outputs from other, similar assessments

2.3.4.4 Summary of Methods

These methods tend to fall into two categories – those that are measurement based such as the CMM or the ISO 9001, and those that involve carrying out a process such as the SPICE Project. All of the methods are quite complex, and require technical knowledge and a good understanding of the procedures before they can be carried out.

2.3.5 Benefits of Software Process Improvement

The benefits of introducing an SPI project into an organisation include improved quality of both the process and the product or service, as these are closely linked [5].

Some of the benefits reported by organisations in studies carried out on SPI include reduced cost-to-deliver, improved ability to predict cost and quality of software being produced, reduced life cycle time, reduction in number of defects after release of product, increase in productivity, better product management and significant cost reduction [7].

Improved quality of product will lead to increased profits and to a better reputation for the organisation. A more efficient production process will create shorter time-to-market – something that is very important in the software development domain as the market changes rapidly. The process may also require fewer resources, which will make financial savings for the organisation.

2.3.6 Barriers to Software Process Improvement

There are costs associated with introducing SPI into an organisation, and there are also barriers that may prevent an organisation from even attempting to introduce SPI. These costs range from the financial costs of setting up such a programme, to the organisational costs of convincing management and engineers that it is a good idea. Costs of such a programme are usually measured in one of two ways. Government organisations tend to think in terms of dollars, whereas industry tends to think in terms of effort expended [7]. Both these measurements should be taken into account as in any organisation, the project will be implemented based on it's proposed financial costs, but it will succeed or fail based on the effort required by the developers. Thus a programme that involves the developers making much more effort than they are currently expending will generally fail as they will not see any benefit to themselves. Other financial costs include set up and running costs associated with an SPI project. These come from increasing or re-allocating personnel to take charge of the project, as well as putting measurement processes into place, and the time taken to act on these measurements.

Some companies can take a long time to achieve observable results when using a process improvement model. Studies of the CMM show that it takes an average of 25 months to move from Level One to Level Two, but that this can take over six years to achieve in some cases [32]. This can be disappointing and can lead to a company giving up the programme before results are shown, thus wasting resources and reducing the chances that another scheme will be given a chance later on.

A lack of resources allocated to SPI activities is a barrier that is repeatedly reported by those responsible for carrying out the work [6]. This can mean that the organisation is

not fully committed to the idea of SPI, or it can mean that the process model chosen is not specifically geared towards the business goals, or it could be that there is a lack of communication between managers and those controlling the SPI projects [6].

Managers can be afraid that rather than bringing "continuous improvement" a SPI project will bring "never-ending change and upheaval". The incremental aspect of such models as the CMM and the ISO9000 can also imply that improvements will become more and more difficult as the organisation progresses and this will require more and more effort [31]. This can be handled by ensuring that the project is carefully planned and paced rather than taking it too quickly. Culture change in the organisation must be taken into account and handled correctly as the impact of this culture change can be initially detrimental to an organisation if people do not understand or welcome it [31]. Managers and engineers alike may feel that SPI work may get in the way of so-called "real work" and they may not wish to devote time to it. Organisations with excessive "organisational politics", or those which require a lot of paperwork to cause things to happen, seem to have more trouble succeeding with SPI than those that don't [38].

There may be concerns that an organisation may not continue a SPI programme once they have achieved a certificate, such as is gained for compliance with the ISO 9001 standard. The organisation may not strive for continuous improvement, or may concentrate solely on gaining certification rather than organisational improvement [24].

2.3.7 Why Is It So Difficult To Introduce SPI?

IT managers often find it very difficult to get resources and time allocated for SPI. Frequently the management of the organisation is not interested in investing in SPI as they do not understand the meaning of the measurements and how the process will work [6]. There may also be a reluctance to allocate time for SPI as other projects always seem to be more important and managers may not want to risk a potential reduction in customer satisfaction [31].

Some process improvement programmes may take a long time to produce results, and management may start to lose patience. The solution for this is to not look too far ahead, and to plan up to 3 months in advance rather than trying to plan, for example, two years into the future [39], or to keep the focus of the planning to a narrow field.

There are difficulties in managing the software development process that are not found in the more physical engineering disciplines. Software developers must rely more on creative instincts than hardware engineers, the processes are more difficult to

measure, and there are few development standards. Therefore it is more difficult to produce a development process for software than it is for hardware and thus more difficult to improve that process [25].

There are available process improvement models, for example the Capability Maturity Model (CMM) (see section 2.3.4.1 above for more details of the CMM). However, these tend to be aimed at large organisations, for example *small* in the CMM describes companies with less than 70 employees. As there are many software firms that have less than 10 employees [30], these software models are not really appropriate for many of the smaller software producing organisations [33]. Small companies are becoming aware that SPI is very important, and thus are requiring procedures which are tailored to suit the relevant organisation [30].

2.3.8 SPI In Small Companies

There are issues that need to be taken into account when introducing SPI into small companies. These include a high dependency on individuals who have become experts in the area, a small number of employees leading to one individual carrying out several roles in the process, dependence on a small number of projects, the importance of communication with customers, and a difficulty with finding the time, personnel or money to invest in SPI. [30]. All these problems can be surmounted provided the model chosen considers the characteristics of small companies.

2.3.9 Summary

This section has provided a basic introduction to the topic of Software Process Improvement (SPI) and a review of some of the specific models and methods used to achieve improvements to software processes, particularly those of small companies. It has shown that all the process improvement models follow a very basic formula, that of determining what the current process is, identifying potential improvements, carrying these out and then measuring the results. This is then repeated as required to gain continuous improvement.

2.4 Web Development

2.4.1 Introduction

Web Engineering can be said to "*deal with the process of developing Web-based systems and applications*" [28]. A more formal definition of Hypermedia Engineering is "The Employment of a systematic approach to the development, operation and

maintenance of hypermedia applications" [2]. This is based on the IEEE definition for Software Engineering [40]. Web engineering was first established in 1998 as a new discipline [41, 42] when it was determined that there were new elements of web-based applications that were not covered by any aspects of traditional computer science [41].

When discussing "web-based" systems, this also includes the development of Hypermedia Systems as they use similar or the same technology and are developed in a related way. Web applications can be divided into two categories, Web Hypermedia Applications and Web Software Applications [4]. A Web Hypermedia Application can be defined as "the structuring of an information space in concepts of *nodes* (chunks of information), *links* (relations among nodes), *anchors*, *access structures* and the delivery of this structure over the Web" [4]. A Web Software Application is defined as "any 'classic' software application that depends on the Web, or uses the Web infrastructure, for it's correct execution [4]. This includes such applications such as databases, booking systems, numerical software etc.

2.4.2 The Differences Between Web Engineering and Traditional Software Engineering

There are two schools of thought about whether web development is the same as software development. One believes that it is a completely different discipline that requires new processes, and the other believes that the two are very similar and that conventional engineering methodologies can be applied [43]. Most people believe that the details of web site development are different from traditional software engineering in many ways, but that the general principles remain similar [43]. Generally, web development has a much smaller timescale, and methods and technologies change very rapidly, making it very hard to successfully develop systems that remain maintainable over a long period of time [44]. The shorter development time usually allocated to web projects can also lead to a 'get it on the web' attitude, with developers reacting to complaints, comments and pressures (both external and internal), or work done by competitors, rather than being proactive about development [45]. There is also a problem with trying to migrate legacy systems to work in web-based environments [42]. This is related to the change in pace of technology and the rapid uptake of web-based systems [46, 47]. This can lead to a lack of planning and design, as clients want to see results more quickly, leaving less time for these crucial activities.

Originally, the web was designed to be an information medium rather than an application medium, so development is considered by many to be an 'authoring' activity rather than a development activity [42]. Currently, web development is "*a mixture*

between print publishing and software development, between marketing and computing, between internal communications, and between art and technology" [42].

A different type of problem is that business clients tend to be very uncertain as to their requirements for the website. This is not unusual in conventional systems, but further to this, they are also not certain about the business problems that a web application may be solving [46]. Therefore, the use of a traditional software engineering method, which relies on the production of solid requirements, is not necessarily very suitable in this case. Many web projects tend to be driven by a vision of what can be done, rather than what is needed [47], and this can lead to a lack of clarity in the requirements, and may also lead to requirements creep where the user increases the requirements as they see what can be done [46]. The lack of understanding can also lead to poor quality and un-maintainable applications because of the lack of planning [48].

Most large web sites are generally in a state of ongoing maintenance. There is never a finished product and thus it is a "process, without a well-defined ending point" [49]. This is different to traditional software where there is a final product to be deployed to the customer. Web developers can make immediate changes to their product, where the system is modified for all customers, without the difficulties of marketing, distribution etc [43].

The ramifications for businesses that have problems with their websites can be far more wide reaching than software problems. For many businesses, their website may be the direct link between them and their customers and business partners so a failure in the system is not only inconvenient, it may also be very public [46].

There have been many different models and processes proposed over the last few years to help improve web and hypermedia development. These all have their individual strengths and weaknesses. Some processes deal solely with an individual section of the lifecycle e.g. the Relationship Management Model (RMM) deals with the design phase, and others attempt to cover the whole lifecycle e.g. the Web Development Cycle.

There are also differences between traditional software applications and hypermedia applications. Generally, hypermedia systems designers start with complex information domains, and have to break them down into concise and small domains that can be presented to the end user [50]. Traditional software domains tend to be much smaller and less complex.

Sometimes the development processes of Web Applications and Software Applications converge with the inclusion of dynamic content such as CGI scripts or Java Applets [51].

2.4.3 A History of Web Development Processes

Recently, there has been work on development processes for both web-based applications and hypermedia applications. Some of these are described below. There are two main categories of processes – those which attempt to prescribe the whole development cycle, and those that concentrate on a specific stage of the development cycle. Early processes were aimed at the development of hypermedia and hypertext systems rather than web-based systems, because the web had not yet become such a major force. However, the theories are similar and the processes can easily be applied to web-based systems.

In 1990 the Dexter Model was proposed [52]. This was an attempt to bring together the current experts in the field, and produce a new model for the abstraction of hypertext applications, based on their experiences of hypermedia design [2]. This was the first model that dealt specifically with hypertext rather than traditional software engineering techniques. The Dexter Model was an attempt to document the characteristics of hypertext systems to allow for comparison between systems, and a basis on which to develop standards for future systems [52]. It divides a hypertext system into three layers, called the *storage layer*, the *run-time layer* and the *within-component*. The internal structures of both the storage and run-time layers are defined, but the within-component layer is left open to allow for any type of media to be used [53]. These layers are shown in Figure 7 [54].

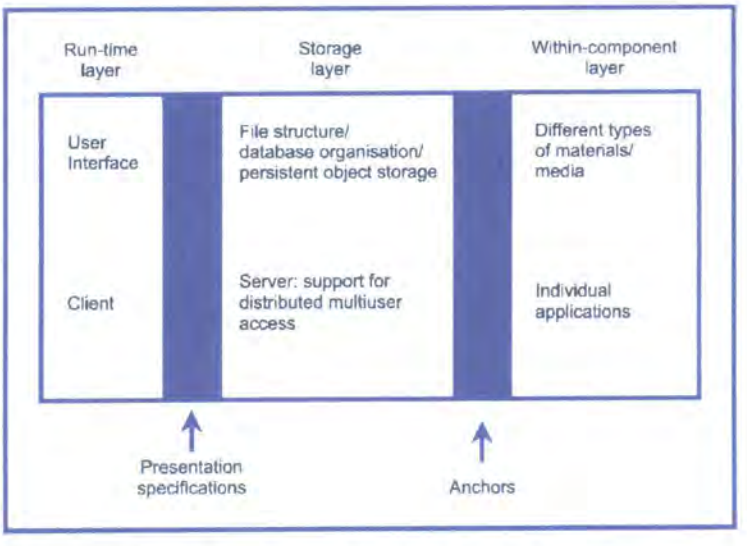


Figure 7 - Layers in the Dexter Model

In 1993 the Hypertext Design Model was proposed [55]. This attempted to present a model using the notion of *authoring-in-the-large* which allowed the description of elements and navigational aspects without concern over the implementation details [55].

The Amsterdam Hypermedia Model was proposed in 1994 [56] and was based on the earlier Dexter Model combined with the CWI Multimedia Interchange Format multimedia document model. It extended Dexter so that it included support for dynamic content.

Isakowitz proposed the Relationship Management Model (RMM) in 1995 [57]. The proposal for the design pre-dates work on website design, claiming to provide a structured design for hypermedia, rather than web based, applications [47]. RMM is based on the Entity Relationship Model (ERM), which describes the relationships between information objects. RMM extends the ERM by including navigational constructs. This allows the information to be displayed graphically and breaks down an information domain into smaller entities [50]. It has seven stages:

1. ER Design
2. Slice Design
3. Navigational Design
4. Conversion Protocol Design
5. User-Interface Design
6. Runtime Behaviour Design
7. Construction and Testing

The Web Development Cycle [58] consists of four fluid steps which describe the general actions to take when creating a new website. These four steps are:

- Information triage
- Evaluation
- Implementation
- Re-evaluation

These steps feed into one another as shown in Figure 8 [58].

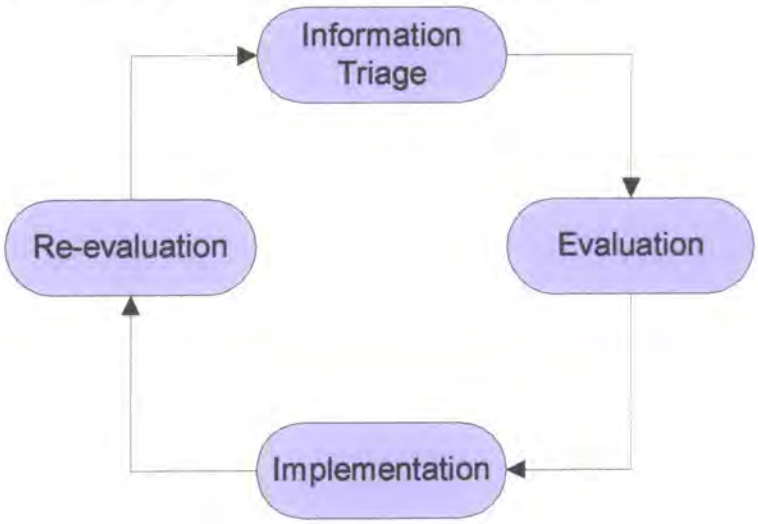


Figure 8 – Web Development Cycle

This cycle assumes that the concept for the site is clear before the process begins, which can be difficult for many clients. This cycle can be put together with the Benchmarks of Software Development Process [58] to form a process suitable for development of new applications. These benchmarks are:

- Conception
- Design
- Implementation
- Testing
- Documentation
- Release

More recently, focus has been on a more general process for overall development, which incorporates the specific stages. For example, the Web Development Framework proposed in 2000 [4] acts as "a malleable development support environment" and contains interoperable modules that can be put together to produce a development process. This method consists of a "generic framework with a modular architecture, based on the synthesis of interoperable modules, in order to satisfy the stated requirements" [4]. These modules are defined as sets of tools that can be used to support the functions of the modules. This has the advantage that new tools can be integrated into the framework as they are developed, and older ones can be phased out. A developer must select the tools he requires for the task in hand, and can benefit from any advances in technology. The RDF (Resource Description Framework) ensures that the data resources such as databases, video/audio files, HTML pages etc can all be interconnected. RDF incorporates XML as its common syntax.

The framework has four layers

1. Implementation Layer
2. Application Layer
3. Information Layer
4. Data Layer

Each layer has a number of Modules – for example, the Information Layer has a Conceptual Model Design Module, and an Access Module. The framework takes into account scalability and technology independence [4].

The most recent development process to be proposed is the Web Modeling Language (WebML), which was proposed in 2000 [59]. This is a model for specifying complex web sites and web applications using distinct orthogonal perspectives. These are:

- The Structural Model (the data content),
- The Composition Model (the pages that compose the site)
- The Navigation Model (the links between pages)
- The Presentation Model (the layout and graphics requirements)
- The Personalization Model (the customisation features)

In 2001, Ginige and Murugesan proposed ten steps for successful web development [28]. These are:

1. Understand the system's overall function and operational environment including the business objectives and requirements.
2. Clearly identify the stakeholders – that is, the system's main users, the organisation that needs the system, and those who fund the system's development.
3. Specify the technical and non-technical requirements of the stakeholders and the overall system.
4. Develop an overall architecture of the Web-based system that meets the technical and non-technical requirements.
5. Identify sub-projects or sub-processes to implement the architecture. If the sub-projects are too complex to manage, further divide them until they become a set of manageable tasks.
6. Develop and implement the sub-projects.
7. Incorporate effective mechanisms to manage the Web system's evolution, change, and maintenance. As the system evolves, repeat the overall process or some parts of it as required.

8. Address the non-technical issues such as revised business processes; organisational and management policies; human resources development; and legal, cultural, and social aspects.
9. Measure the system's performance.
10. Refine and update the system.

Studies have been undertaken of the various development methods, and the common elements have been abstracted out into a high level model [44]. This is shown in Figure 9.

This is very similar to the Waterfall Model, or System Life Cycle for Software Engineering [60-62] first described in 1977. This is shown in Figure 10. The Waterfall Model is a very simple model for software production that is quoted in many textbooks and courses. This model is primarily for traditional software development but the principles can be applied to web development as well. Some of the weaknesses associated with the Waterfall model include customer disassociation, design inflexibility and documentation problems [60, 61]. However, many of today's successful development processes are still based on the Waterfall Model [60].

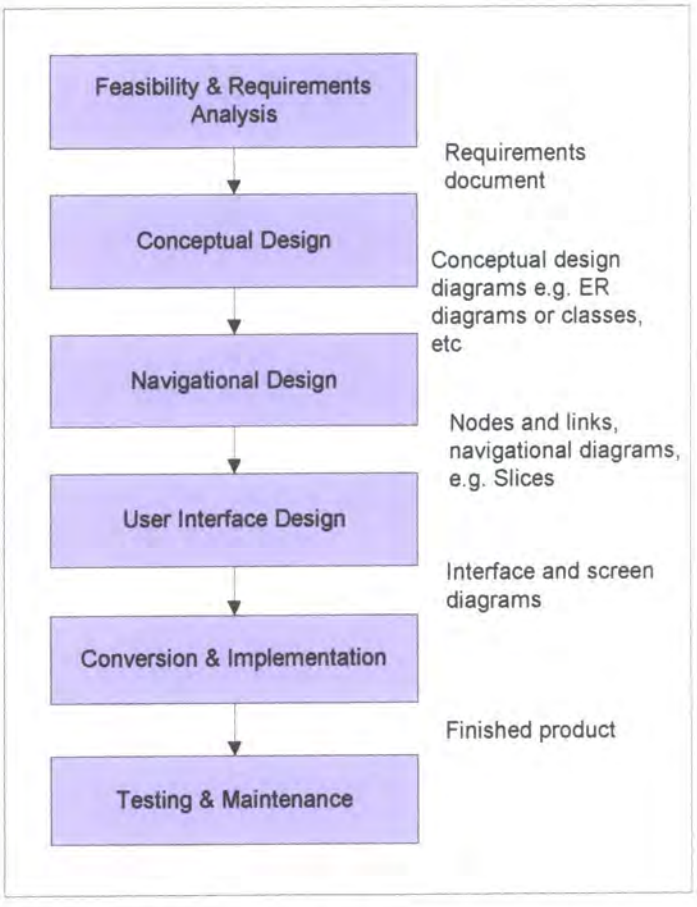


Figure 9 - Hypermedia Development Stages

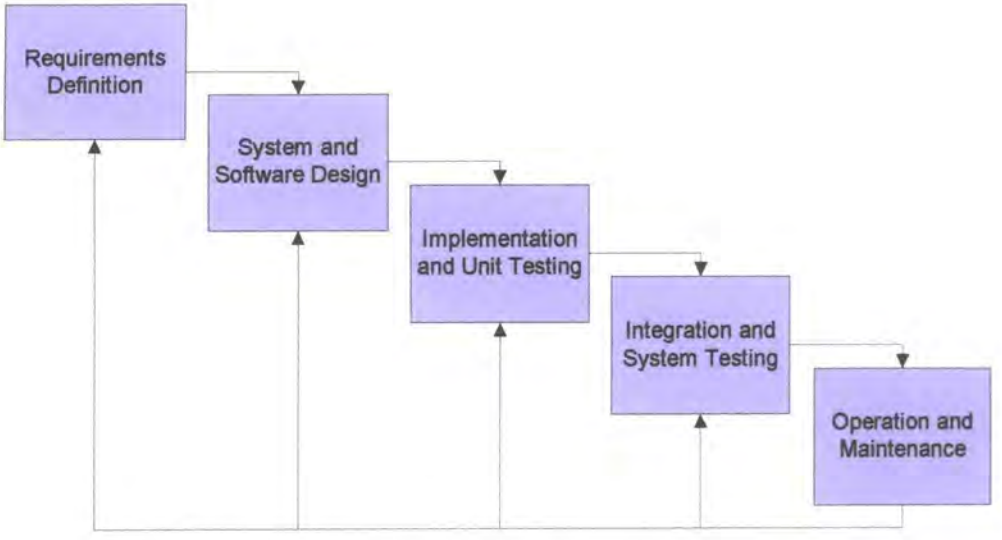


Figure 10 - Waterfall Model for Software Engineering

A similar, simple, development process is the Prototyping Model [62]. This incorporates the ideas from the Waterfall model and includes a prototyping stage that is aimed at solving the problem of the inability to adjust to change that the Waterfall model suffers from. Including a prototype will ensure that the project is feasible and will also act as a blueprint for the final production. Prototypes are generally a live implementation of a system that can be modified to explore changes to demands. It is generally not fully tested, nor will it meet all the speed and storage restrictions that the final system will. Some prototypes are purely facades of the system, that is they may look like they are carrying out the functions but they may have a modified database, or may be using primary storage instead of secondary. An example of the prototyping process is shown in Figure 11.

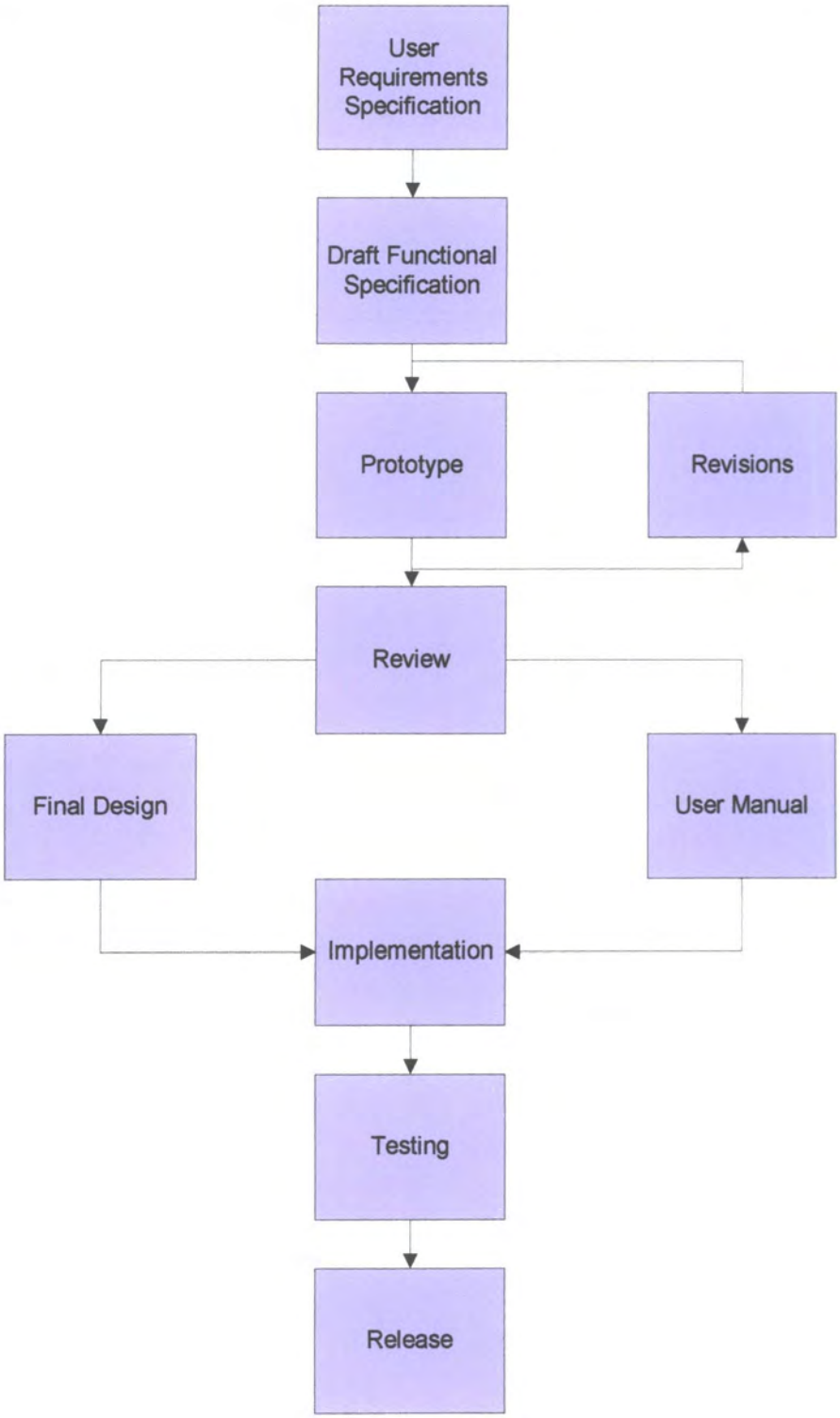


Figure 11 - Example of Prototyping Process

2.4.4 Why Do We Need Processes in Software Development?

Processes and models are needed in order to allow developers to understand development activities, and to manage the development of systems so that they are more comprehensible, easier to maintain and generally better for organisations as they have the ability to update and improve their processes [27].

Organisations need to be able to manage the design and development of systems in a systematic and repeatable manner as this will aid in the reuse of components as well as allowing developers to follow a clearly defined series of steps [28]. Some of the highly skilled developers argue that web applications cannot or do not need to be engineered like traditional software, and that trying to introduce processes will not work [63]. Indeed, one of the attractions of the web and hypermedia is that relatively untrained developers can produce applications very quickly and easily, but this leads to poor quality and difficulty when trying to perform maintenance activities [64].

It is important to document the current web development process for several reasons. It is important that development processes can be compared over time to show improved capability. The current development process status will give the developers information about the point at which they are starting and will give a comparison level so that the business can measure how their processes have improved over time. There are three reasons for documenting a process [65]:

1. To communicate – to yourself and others
2. To understand – if you can write it down then you understand it
3. To encourage consistency – repeatability can be identified

2.4.5 Problems with Web-based Development and Web-Based Systems

Hypermedia and web development processes are following the same line of problems as traditional software engineering went through thirty years earlier [2]. The industry is facing a problem with the maintenance and evolution of hypermedia and web based systems as they have grown and grown, and the systems are now experiencing similar problems to legacy software systems did when they became too difficult to maintain and to upgrade [63, 66]. Both types of systems tend to be very difficult to maintain, and evolve. They have a lack of formal engineering techniques and often a lack of accurate documentation [67].

In addition to the similarities in the systems themselves, it has been noted that the types of people who are active in web engineering at the moment, are similar to those who were active in computer programming in the 1960s, and that those in charge of the old legacy systems are similar to those who were using record files systems [41].

Another problem with many current web development activities is that, due to the rapid growth of Internet and Web-Based applications and technologies, most development takes place on a very ad-hoc basis [68], or is based on rough guidelines and the intuition and judgment of the developer or development team [69]. Many designers start work straight into the implementation phase without paying any attention to requirements or a design phase [51]. Web development tends to lack a systematic approach, and quality control procedures [3]. It has been described as "*more of an art than a science*" [57]. This leads to applications that can be very successful, given an appropriate amount of expertise and luck [2], but in general have problems with poor quality, lack of structure, difficult navigation and maintenance issues [44].

Although amateur designers can have highly developed skills, and much artistic ability, their sites can be constrained by a lack of technical knowledge and structure [70]. These problems can be exacerbated by distributed development over time or distance by several different people or teams [44]. This type of development technique lacks the use of metrics to determine quality, and a failure in one system can lead to propagation across others, leading to widespread chaos [68].

The basic problems with the various development methods for hypermedia include the fact that the methods only incorporate one model and cannot support others; most methods have a poor development environment; there is no environment which can support more than one method; development environments are not easily extensible; and most environments do not support efficient importing and exporting for content or design [4].

Much of the material used in website development, e.g. the data resources such as databases, HTML pages and text, video and audio files etc, have been designed for other applications and as such can be difficult to reuse within a web-based environment [4].

2.4.6 Summary

In summary, web engineering is taking a systematic and structured approach to the development of web based and hypermedia applications.
There are many differences between web engineering and traditional software engineering. These range from the time allowed and the technologies used, to the people involved. There is enough similarity, however, for some of the software engineering processes to be modified to become suitable for web engineering.

There are many problems with a large proportion of the web applications in use today. These problems are similar to those observed 30 years ago in the software engineering discipline. The problems are mainly with maintenance and extension of sites, as they have been developed without using a systematic approach, and have little or no documentation.

Chapter 3: Company Analysis

3.1 Chapter Abstract

The aim of this chapter is to introduce the case study company. It will give a short introduction to CACDP, and it's main aims and objectives. Then their current development processes will be described, including the current state of their web site. This has been determined from a combination of interviews with staff involved, and reverse engineering on the current site files using a variety of tools and methods.

3.2 Introduction to CACDP

The Council for the Advancement of Communication with Deaf People (CACDP) is a small organisation consisting of less than 50 staff in four locations within the United Kingdom. CACDP is the national examining board for qualifications in British Sign Language (BSL), Irish Sign Language (ISL), and Deaf and Deafblind Awareness in the UK. As well as organising and administering examinations, they also develop standards in Lipspeaking, Communicating and Guiding Skills with Deafblind People, Interpreting with Deafblind People, Speech-to-Text Reporting (STTR), Note-taking and Electronic Note-taking.

Currently, CACDP has four offices in the UK - the head office in Durham and branch offices in London, Scotland and Northern Ireland. The organisation has been in existence for just under 12 years. The London Office is due to close at the end of August 2002 and the remaining staff will be relocated to the head office in Durham.

CACDP was formed as a response to a need for minimum standards of competency for teaching and assessment of Sign Language Interpreting in the UK. The Deaf Welfare Examining Board (DWEB) previously carried this out for about 50 years, up until the 1970's. CACDP was first based in the Headquarters of the British Deaf Association (BDA) until it became independent, and after a series of temporary addresses it moved to its present location at the University of Durham in 1985. By 1984, CACDP had 18 member organisations, not all of which preferred sign language as a method of communication, so their focus shifted to include provision for Lip Speaking and Deafblind communication as well.

CACDP arrange and provide examinations in BSL, Deaf Awareness, Deafblind Awareness and other subjects throughout the UK. BSL examinations have

approximately 25,000 candidates per year and are the most popular qualification. Most of the examinations take place in Colleges of Further or Higher Education. CACDP do not provide training courses for these skills, as colleges run these, but they provide the curriculum and the examination and they also provide training for tutors.

CACDPs other main activity is the maintenance of the *Register of Interpreters*. This is comprised of those who have reached the highest skill in both BSL and English and who have completed the requisite qualifications and training. CACDP publish this register as their Directory of Registered Sign Language Interpreters, Interpreters for Deafblind People, Speech to Text Reporters and Agencies that are able to provide an Interpreter. CACDP also produce a wide range of training materials and fact sheets. Their main mission statement is "CACDP is committed to the raising of standards and to the development of all forms of communication used by deaf people".

3.3 Introduction to the Project

CACDP make minimal use of the potential of the Internet as a marketing and distribution medium. Currently, they have a simple site that allows the user to gain information and download a few fact sheets. CACDP wish to make more use of the Internet as a further outlet for their services. They are aware that it is a suitable medium for information distribution and already use their web site to provide fact sheets and other information to customers. They would like to increase the services they can offer by producing three Online Learning Packages and also by putting their Register of Interpreters online. Their current site consists of a mixture of text, pictures and pdf files. It is used for information distribution to customers and for various forms and fact sheets.

3.4 The Domain

Ginige and Murugesan [28] recommend that a developer must understand the operational environment, and the business objectives before they can produce a successful web application. This is known as understanding the *domain*. A domain is an area of knowledge. In this case, the domain is Deaf Awareness and related disciplines.

Deaf Awareness covers aspects of deafness from hearing aids, to subtitles, to the Disability Discrimination Act. The course is aimed at hearing people who come into contact with deaf people and who may need to make adjustments, or simply be aware of any issues. Sign Language is not covered in the Deaf Awareness course, other than

as a concept, but it does include Finger Spelling. This is different to Sign Language in the same way that *letters* are different to *words* in English. Finger Spelling in BSL consists of a 26-letter alphabet of signs made with the hands as shown below.

Figure 12 – "J"

Figure 13 – "M"

Figure 14 – "D"

These can be used in sequence to make a word, but this is not Sign Language, although it is used sometimes to spell out nouns and a few other words. Sign Language, in contrast, is a visual language where words mean nothing, and concepts are portrayed instead. Sign Language does not translate word-for-sign into English; for example, the word "sit" has different signs depending on where you are sitting, and on what – a horse, a chair, a wall etc. Deafblind Awareness is a similar domain to Deaf Awareness, but obviously covers different material as it relates to Deafblind people and their problems and issues. Deaf Community and Culture is a qualification that covers topics such as Deaf History, and Social Issues.

3.5 Current Software Development Processes

CACDP currently do not produce any software other than their web site, which is produced in-house by the IT Department. They use off-the-shelf products for general administration and accounting tasks, and are quite proficient as an organisation at using them. There is a culture of using Information Technology in the organisation but this has only really been developed in the last few years as they have introduced e-mail and web access to all staff. Therefore they do not have any current software development processes with which to work. The IT department is responsible for the installation and maintenance of all hardware and software used within the company, as well as the upkeep of the web site. This department consists of the Senior Admin Officer – Information Technology (SAO-IT) who is full-time but does not work exclusively on IT related jobs, and an external consultant. The SAO-IT has had training in appropriate skills required for the job. These are mainly related to specific software packages, and it is the consultant, who comes in when required, who carries out the majority of the more complex tasks with regard to hardware, servers and installation of software.

3.6 Description of Current Web Development Processes

The CACDP web development process is no different to that used by many small companies. The site was designed and created by the SAO-IT after a short technical course, and some discussion with colleagues. This is a very ad-hoc procedure and relies on the competence and presence of an individual. There are no formal written procedures and the design is not diagrammed or explained anywhere. They use standard, graphical, off the shelf software (NetObject Fusion) and the site is hosted by Demon.

Currently, CACDP have a very immature maintenance process for their web site. This is detailed in Table 1. Although it covers all the major points i.e. it details what should happen and how often and who should carry out each task, it is not a formal process and does not give any of the technical information which would be required. A new member of staff taking over the work would not be able to carry out the process given just this information. They would require more information about the programmes used, the current content of the pages, and how to carry out the actions. The procedure in Table 1 was defined by the SAO-IT in September 2001.

Table 1 - Current CACDP Web Maintenance Routine

Date	**Activity**
On-going	Collect information to be added to web site Collect amendments to any items contained on web site Staff pass any amendments/information to SAO (IT) for web site
Daily	Updating of urgent information e.g. HAC registration, new level 3/4 BSL Centres, removing of job adverts past the deadline date etc
Monthly	Updating of non urgent information Amend "What's New" page to include details of changes
Quarterly	Staff are asked to check relevant sections of web site to ensure they are up to date. Staff are sent a copy of the relevant pages as email and check them manually before sending acceptance or comments.

Link checking, e.g. ensuring that there are no broken links, and that those that are working are linked to appropriate pages, is done as the pages are edited and an informal record is kept by memory. No documentation is recorded and no automatic updating takes place. The 'What's New' section on the front page is added to as new

items are included, and items are removed if they are considered to be no longer valid but this is done on a monthly basis, and will therefore be out-of-date quite a lot of the time.

The development process for the site is very ad-hoc and informal. There are no procedures to ensure that updating takes place in a logical or systematic way, and there is no management of information. However, CACDP does make an effort to ensure their site contains up-to-date information and that anything out-of-date is removed.

3.7 Formal Assessment of Development Process

In order to formally assess the development process, a Capability Maturity Model (CMM) questionnaire was applied to the current process. The detailed results of this survey can be found in Appendix A – Results of CMM Questionnaire. The CMM is a method that identifies the maturity levels of an organisation within a hierarchy, as described earlier in section 2.3.4.1. It was decided to attempt to apply this in order to highlight the difficulties that a small company would have in making use of a method of this type. The CMM was difficult to apply to this organisation, as so many categories did not have any suitable match with their activities. For example, out of the twelve areas that the CMM looks at, two are wholly inapplicable to CACDP as they cover such topics as *Software Subcontract Management* and *Peer Reviews*. So much of the questionnaire could not be answered convincingly because it relates to the "*Software Engineering Group*", or the "*Software Development Process*". As the company has no formal process, and no Software Engineering Group, it was very difficult to directly apply the questionnaire. Had the organisation attempted to use the CMM to improve and develop their development methods, they would not have managed it. However, it does give an idea of the type of criteria that an organisation should be aiming at in order to fulfill the requirements. A second CMM survey will be carried out after the experimentation phase in order to gauge progress made.

3.8 Problems with Current Development Processes

There are some very obvious problems with the current process. It is a very labour intensive and manual procedure, and although this is sufficient for a small static site, this would cause problems when the site gets larger and more complex, or introduces dynamic elements. There are no automated procedures for link checking or for creating the "What's New" page, and either of these could be out of date very easily.

During the course of this project, the SAO-IT left the company, and there was a delay before her successor started work. This caused problems with maintenance and lack of written documentation or well-formed processes was a major factor. The staff detailed with taking over the web site temporarily, did not have the requisite skills or instructions required to continue maintenance of the site. Small changes were causing large problems, and major changes could not take place at all during the handover period. No other member of staff had been involved in the design or development of the site and there was no documentation to help them understand the process. As part of work before the SAO-IT left, all procedures that she carried out had to be documented. The web development procedure is shown below:

Procedure as documented by SAO-IT

1. Log out of all software
2. Go to start menu and choose NetObjects Fusion 4
3. The 'Welcome to NetObjects Fusion' screen will appear. It is set up to automatically open the correct web site, so click on the OK button.
4. On the toolbar click on the site icon. (This shows the contents of the web site). A list will appear down the left hand side of the screen. These are the pages on the web site. To get to a page double click on the page you want.
5. Text is either typed into a text box or table. To change any text double click in the box until a cursor appears. You are then able to edit text. On the right hand side of the screen is a text properties box. Use this to format the text. As in Word highlight text you want to bold etc and change the format as normal.
6. When using tables rows can be added to a table in the same way as in Word. Go into the Object menu and choose Table from the list. Options to format the table will appear.

The difficulties in just writing this process showed immediately how much knowledge the individual concerned implicitly held. In the end, a member of staff from another office was drafted in to take over as he had 'home-grown' web skills. This caused problems when work needed doing to the site, and as a result it was not as up-to-date as it had been previously. Even once the new SAO started work, she did not have the skills necessary to take over the work immediately, despite previous experience with web development. It was some time before the site was once again being maintained from the main office.

3.9 CACDP Web Site at Start of Project

The CACDP web site documents at the start of this project have been evaluated using various freely available tools in order to give a comparison for the final development process. The site was run through various checkers including the W3C HTML Validation Service (http://validator.w3.org/), the W3C Link Checker (http://validator.w3.org/checklink) and the Bobby Program which checks sites for accessibility problems (http://www.cast.org/bobby/). These are all freely available on the Internet. In general, the site passed the HTML checker tests, as one would expect since the web development software generated the HTML automatically. The Bobby program produced a report showing some accessibility problems, but as the site is not very complex, neither were the problems it reported. They were mainly a lack of any ALT (alternative) text for images, and use of graphics as links without, again, any ALT text. ALT text is used by screen readers, which are used by those who are partially sighted or blind, to produce an audio description of the graphic. Thus, if there is no ALT text for a link, they cannot tell where the link will take them, and the site becomes unusable. These are issues that standard HTML packages do not always highlight.

The site is hosted, free of charge, by Demon.Net at www.cacdp.demon.co.uk. This offers web space and email forwarding but no domain names, and no facilities for scripting. Thus a new host will need to be found which offers more services. CACDP have expressed a desire for their own domain name – www.cacdp.org.uk and for their email addresses to follow the same format. They feel this will enhance their professional image, and offer more confidence in their facilities.

3.10 Action List

As a result of the investigation into the requirements of the company, and of their current processes and procedures, an Action List was drawn up. The Action List consists of the tasks that need to be completed in order to achieve the objectives:

3.10.1 Determine a Formal Development Process

Theory states that having a formalised development process will help with maintenance, and with the quality and costs of system development [7]. CACDP currently do not have a formalised development process, and are also experiencing the problems associated with not having one such as inability to hand over maintenance and development activities from one staff member to another. Thus we must identify a formal development process suitable for the organisation to put into place immediately.

This should not be too complicated at first as CACDP have never had a formal process for this and the natural resistance to change within the organisation may hinder attempts to introduce a complex process that requires a lot of learning and effort to ensure that it works. A simple process will be introduced, and once it is in place and has been accepted, process improvement activities will be carried out to improve the process so that they can gain as many benefits as possible.

3.10.2 Simplify Maintenance Procedures

The maintenance procedures for the new systems need to be as simple and as quick as possible as CACDP do not have the appropriate skill level to make complex alterations to the systems. This can be done by the use of templates in web design, and by the use of forms for database maintenance.

3.10.3 Reduce Development Time

Development time needs to be as minimal as possible, as CACDP do not have dedicated personnel to allocate to this project. This will be done in this case by introducing the concept of reuse of previous work to the organisation. Currently, if their web site needs anything changing that affects every page, it will need to be changed by hand. This can be very time-consuming and can lead to errors where instances are overlooked. This prevents CACDP from attempting to make any major changes in the aesthetics of their site.

3.10.4 Introduce Accessibility Issues

Accessibility issues need to be addressed as an integral part of the development process as this is a very important aspect for the organisation. These are not addressed in any of the development processes discussed in Chapter 3, so provision will need to be made in the development process that is selected. A set of guidelines will be defined and all software developed, including the organisation's current web site, should be subject to these guidelines.

3.10.5 Simplify Development Environment and Introduce New Facilities

The current web development software used by CACDP is complicated and does not offer any tools for making the development easier, by using, for example, templates. It introduces a lot of "unclean code" into the HTML, which does not help to make maintenance straightforward. Introduction of a new software package will make development more straightforward, allow development of more complicated systems,

and move CACDP forward in their aims for the organisation to become more technically able. CACDP also require the somewhere to host their new services, as their current facilities will not be suitable.

3.11 Summary of Chapter

This chapter described the company being used in the case study – CACDP. It also described the aim of the project and what the company is trying to achieve from it. Finally, it described their current development process, and what the problems are with it, leading to an Action List of activities that need to be included in the proposed solution to the problem.

Chapter 4: Experimentation

4.1 Chapter Abstract

This chapter will describe the experimentation, that is, the process of carrying out the proposed development methods and the review processes used to determine potential improvements in them for the next project. It will describe each individual sub-project in turn and the process used to implement it. Then it covers the problems encountered each time and the proposed improvements to the process for the next project. Finally, there is a summary of the whole experimentation phase of the project.

4.2 Introduction

The overall aim of this project is to study reuse and process improvement in the domain of web development by producing four web-based products. As noted in the previous chapter, many of the available Process Improvement Procedures are not suitable for this type of organisation as they are too complicated and would require far more mature organisation before they would be of any use. The commitment to, and understanding of, process improvement within CACDP is not such that they would manage to carry out the complex procedures described previously. CACDP are not familiar with any type of formal process at this stage and therefore it is required that a much simpler and more comprehensible method of introducing Process Improvement be determined.

A further aspect is to introduce to concept of web accessibility to the organisation, and ensure that all software produced meets with a set of defined guidelines where possible.

The plan for this project is to start by designing and implementing one product, and then develop others using the processes and technology designed in the first. This will allow research into the reuse of processes and procedures, as well as the actual software itself. The processes used will be reviewed after each project and improvements built into the process for the next project. This method is based on the iteration model described in Chapter 2.

In this project, CACDP and the developer will work within two main domains of interest. The first is "Online Learning" and the second is "Online Directories".

The first product to be produced will be an Online Learning Package for Deaf Awareness (DA). This will be designed and implemented and the processes documented. From the basis of this original product, two more products will be produced. The second product will be an Online Directory of Interpreters, which will allow research into whether parts of the system can be reused outside the 'Learning' domain. The third will be another Online Learning Package, this time for Deafblind Awareness (DBA). This will be very similar to the first product, but with enhancements, and will allow the investigation of reuse within the same domain, and a chance to see whether the reusable aspect of the DA package was correctly designed and implemented.

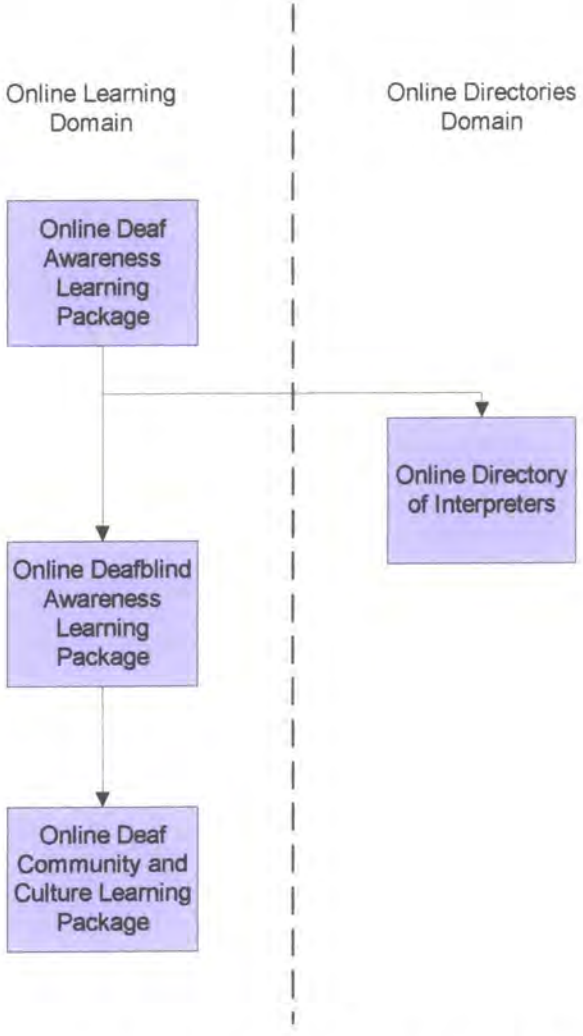


Figure 15 – Packages to be produced

Finally, another Learning Package, Deaf Community and Culture will be developed which will allow for fine-tuning of the design and production process with the final aim of describing a well defined and documented process, which the CACDP will be able to use to develop more online learning packages in the future. This project will investigate the area of reuse both within the initial domain (Online Learning), and within a second domain (Online Directories).

4.3 General Process Description

The method of process improvement chosen for this project is *iteration*, as described in the Literature Survey. This method was chosen because it is very simple, and will be easily understood by the organisation. Figure 16 shows how this will be carried out for this project.

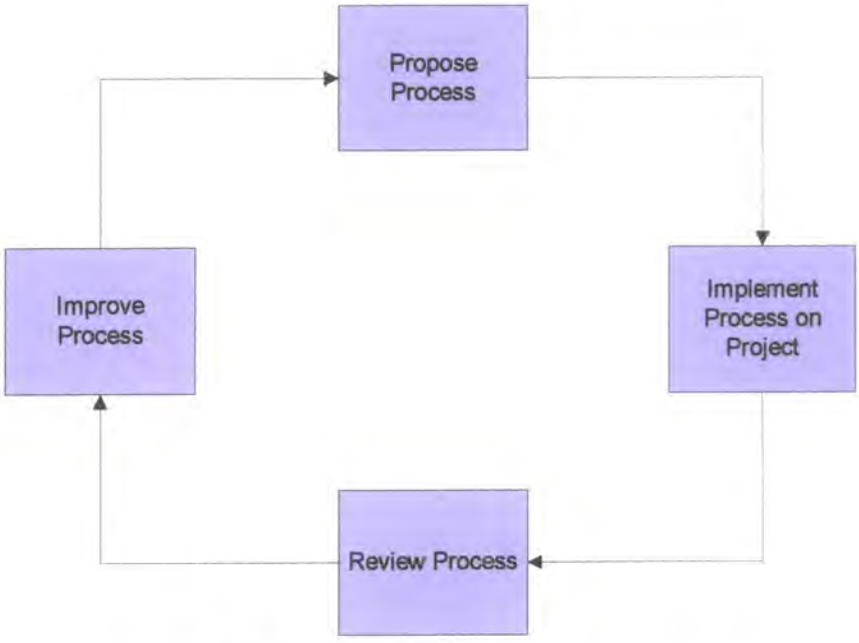


Figure 16 - Process Improvement Flowchart

Each individual package will be implemented and then the process used will be reviewed and analysed, and potential improvements identified. These improvements will be put into place in the next project, finally ending up with a design and implementation process that CACDP can use to produce further web-based packages in the future, and simultaneously increasing their technical knowledge in the process. This method was chosen because it is simple to execute whilst products are being produced and can be understood by the company staff. This also meant that it was likely that they would continue using this method to improve other processes after the end of the project. The use of the CMM Questionnaire, as described in the previous chapter showed that it is important to define a very simple process for use with this type of company, otherwise they will not embrace the change process, nor will they been enthusiastic to assist in the implementation.

4.4 Proposed Product Development Process

The proposed product development process is based on the standard software development process, The Waterfall Model as shown in Figure 17 [5], and described in Chapter 2. This was selected so that feedback could easily be given to the company. It is simple so they will be able to understand the stages, and this will give them more

motivation to accept the process and ensure that it is carried out successfully. There are documented problems but these can be identified through the use of the model and solutions put into place that are suitable for the demands of this particular company.

The Waterfall Model begins with the gathering of the customer requirements and production of a system design. From the design, the system is implemented and then tested, first as individual units, and then as a united system. The software system is then considered to be complete. After completion of the system, maintenance activities are required. This stage is generally ongoing in a web-based product as the content changes rapidly and development activities are always occurring. However, due to the nature of the products in this project, it will be only the content that changes in this case, and therefore a simple maintenance step can be included. If, at any point, further development of the application is required, then a more complex maintenance procedure should be defined based on web maintenance principles.

The Waterfall Model allows the stages to be iterated continually until it is possible to move on. Therefore, if an issue is causing problems at a later stage, it is possible, and indeed desirable, to go back to an earlier stage to correct the problem. In general, the later on in the process a problem arises, the more difficult it will be to solve.

The Waterfall Model was selected, as, even though it is not specifically intended as a web development process, it is simple to understand and carry out. The stages can be individually planned to suit the organisation, and thus it was decided that it was more suitable for CACDP than a more complex web-specific process that they would not understand and would not continue to use. While, based on the theory [61], it is known that there are problems with the adoption of this model; its simplicity can be utilised to help the company to understand better where the problems lie and hence provide the justification and motivation to correct them during later iterations.

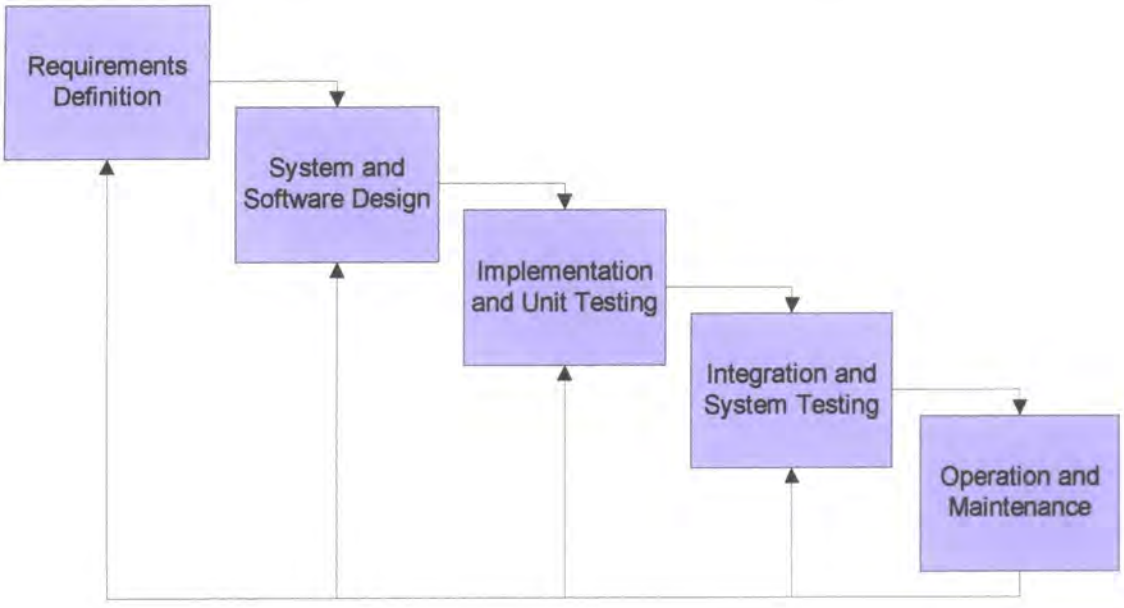


Figure 17 - The Waterfall Model

4.5 Project One: Deaf Awareness Learning Package

4.5.1 Description of Project

The aim of this sub-project is to design and implement an Online Learning Package for a Deaf Awareness Course. This will be based on material already available in a paper-based form. The course is currently delivered by a tutor and takes approximately two days to complete. The course is assessed by a written paper that takes 30 minutes to complete and consists of 20 multiple-choice questions and 8 longer answer questions.

The online version will be used through a standard web browser. The package is aimed at individuals and commercial organisations that may not be able or may wish not to learn from a tutor. It should take approximately 10 hours to complete. The course will consist of a mixture of text and pictures and there should be a Mock Examination at the end of the course.

As part of this project, it will be necessary to find a new hosting company that can offer the required services for web hosting and for email. This will enable CACDP to move their current site over to it and start to publicise their new domain name and email addresses before the new systems are completed.

4.5.2 Location of Hosting and Email Facilities

During the early stages of Project One, investigations were carried out into a number of potential web hosting companies. The requirements were:

- Local company if possible – the organisation likes to deal with small, local companies if possible as they find this works well in terms of building a long-term business relationship.
- Able to offer more assistance than just provision of the hardware required – CACDP wanted to find a company who would be able to solve problems if they arose as they do not have the in-house skills for this.
- Inexpensive – CACDP are a non-profit organisation and thus do not have the funds for expensive facilities
- Able to provide web and email hosting, domain name purchasing, and scripting facilities.

The company found was *The Web Works*. This is a small company based just outside Durham. They can offer all the facilities needed and are not too expensive. An account was set up, the new domain name purchased, and new email accounts set up that used the new domain name. Web space was provided that allowed scripting using Microsoft® Active Server Pages (ASP).

4.5.3 Definition of Web Accessibility Guidelines

As stated in the action list, a set of guidelines for web accessibility must be defined so that all software products developed can follow them. The following guidelines are compiled by the WAI [71]. CACDP aim to conform to Level A of the Web Content Accessibility Guidelines where applicable, taking special consideration of issues that will affect deaf users. They intend to ensure that all their sites are written in Plain English [72].

4.5.4 Development Process

The development process used for Project One is based on the standard "Waterfall Model" as described in Sommerville [5] and shown in Figure 17.

In the spirit of the model, the processes for each stage will be planned before the stage commences, based on the current information and what needs to be done.

4.5.5 Requirements Definition Stage

The following section outlines the plan for the requirements section, followed by the domain investigation, and issues resulting from the requirements gathering.

4.5.5.1 Plan for Requirements Definition

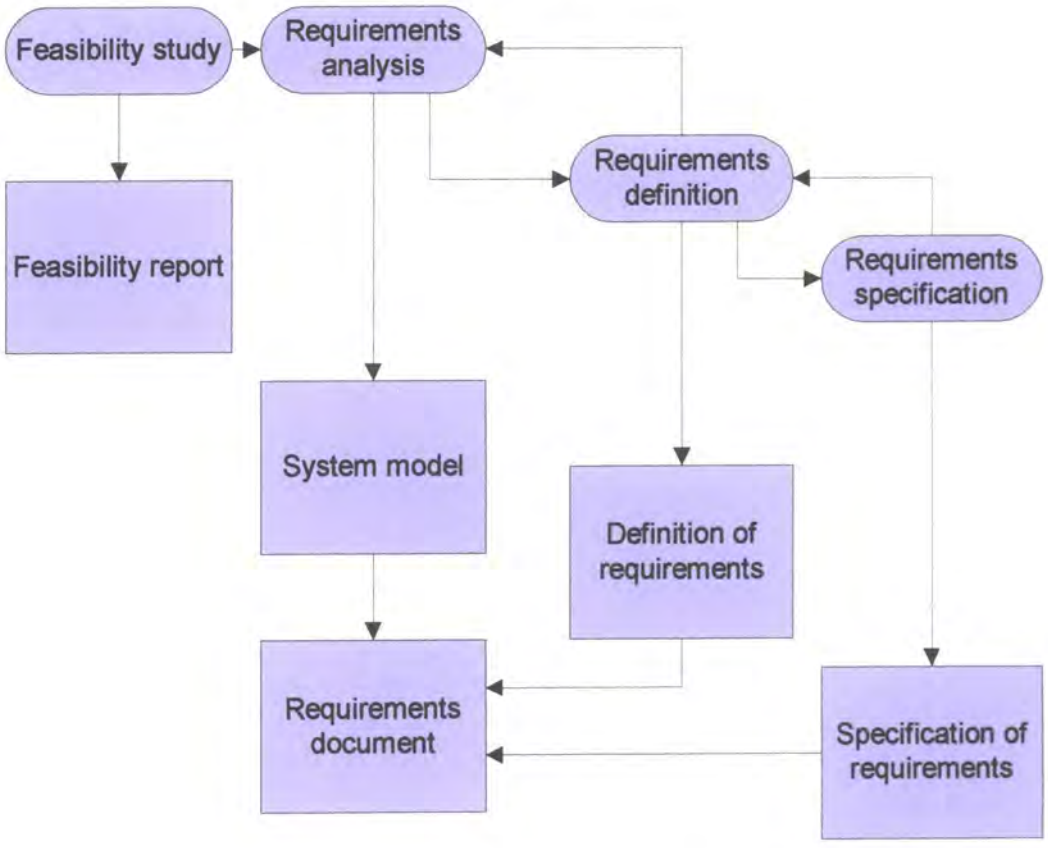


Figure 18 - Requirements Process

The process for the Requirements Definition stage found in Figure 18 [5] shows the various activities that should be carried out during this phase.

To fulfil the *Feasibility Study* phase it was decided to carry out a domain investigation to see what other similar packages were around as no feasibility study had been carried out by the organisation. After this it was decided to carry out a market survey in order to gather information about the general user that the system was aimed at. The findings from both the domain investigation and the market survey would be written up and presented to the stakeholders. This will be the *Feasibility Report* as identified in the left hand box in Figure 18.

The *Requirements Analysis* phases will incorporate interviews with the stakeholders in order for them to define their requirements based on the findings from the Feasibility

Report. This will result in the production of the *Requirements Document*. This will be used to create a *Requirements Definition* and then a *Requirements Specification*.

4.5.5.2 Domain Investigation

The domain investigation for this project consisted of research into existing similar packages which teach Deaf Awareness (DA) or Sign Language (SL) using multimedia (either web-based or CD-ROM). It was decided that packages presented on CD-ROM would be included in the material as they use a similar technology to online work, and they will give other ideas as to what can be done. This survey was to allow the developers and the client to see what is possible with the technology available, and also to carry out a study into the strengths and weaknesses of each of the packages.

Four software packages were studied, all of which were delivered on CD-ROM, as there were no equivalent online packages available at the time. The systems that are available online in this domain taught just finger spelling rather than full sign language or any other information about the subject, and thus were not considered to be comparable systems. Of the four systems surveyed, one taught Deaf Awareness (including a small amount of British Sign Language (BSL)), one taught BSL and two taught American Sign Language (ASL). The ASL systems were included for two reasons:

a) There are not many systems in this domain so they were required for a more complete survey
b) The survey was looking at the way the content was taught, rather than the content itself, and therefore it was of no consequence that the language used was ASL rather than BSL as they use similar teaching methods and would encounter similar issues during development.

In theory, other language systems, and indeed any other online learning systems, could have been surveyed for their teaching methods, but it was decided that Sign Language is visual rather than aural and has different demands to other languages. Therefore it was decided to concentrate on the systems than taught Sign Language as these offered domain familiarity, where the clients were familiar with the material contained within the systems.

The systems were evaluated on eight specific criteria:

1. *System Usability Characteristics* – this includes the presentation of the information on the computer screen, its layout and the general 'first impressions' felt by users.

2. *Language Specifics* – this evaluation criterion investigates if the software is composed for a specific deaf communication language and what, if any, implications this support will have for other deaf communication.
3. *Pedagogical Nature of the Application* – including the training approaches used and whether this appropriately supports the learning process.
4. *Navigation Capabilities* – this criterion evaluates if learners are able to appropriately navigate around the software, whether they become lost and disorientated and the overall effect this has on the training.
5. *Use of Technology* – this aspect evaluates the types of technologies that are used and how effective each type is in terms of supporting learning.
6. *Real Time Capabilities* – what speeds are demanded by the software and whether technology is able to deliver the required expectations and overall how this affects the learning process.
7. *Entertainment* – since many of the users will be using these courses as non-vocational the entertainment factor is also considered an important aspect in the process of maintaining the motivation of the learner.
8. *Accuracy* – the information provided by the system must be accurate and correct.

The first seven criteria were taken into account in the survey, but Accuracy was not covered due to a lack of availability of experts in ASL at the time.

4.5.5.3 Survey Results

The results of the survey were in the form of recommendations for the new system. The full report of the survey results is shown in Appendix B. These were:

- *Screen Size* – The system should be designed so that it takes up the whole screen, otherwise it is too small to read the text properly, and the background can be distracting. Several of the systems looked at displayed their software in very small windows. This was deemed to be difficult to see, and users always attempted to make the window larger immediately and were disappointed when this proved to be impossible.
- *Navigation* – The navigation system should be clear, simple, consistent and obvious. Ensure that there are textual alternatives if icons or graphics are used. One system looked at used only faint graphics as a navigation system, so not only were they difficult to find and see on the page, they had no clues as to which icon had which function.

- *Approach* – Use a lesson based approach, which progresses from level to level so that users interest and attention is kept, and they can build on the information they gained in previous levels.
- *Logical* – The lessons should be arranged in a logical order rather than randomly. Users should feel that they are building on previous learning rather than approaching each topic from new.
- *Assessments* – Use short tests in order to challenge the user, but make them realistic, i.e. in terms of timing and length.
- *Video* – Video clips are essential when attempting to teach sign language. Textual descriptions in addition are also desirable as they offer another way for the learner to grasp the concept.
- *Interactivity* – Systems should be interactive, and should keep prompting the user to take part and take in the information
- *Text* – Ensure that the text is large enough and clear enough to read, especially if partially sighted people may use the system.
- *Replay* – Let the user select when and how many times they wish to view the video clips.
- *Placement* – Topics should be reasonably short in length otherwise users may get bored. If possible, an indicator should tell the user how far through the particular module or topic they are, so that they can judge how long they have to go.

These recommendations were taken into account when devising the requirements for the Learning Packages. Some of the recommendations can also be applied to the Directory, for example those concerning the screen size or the text recommendations, thus identifying reuse of domain knowledge between domains.

The domain investigation was useful as it allowed the developer to get an idea of the types of packages already available, and how the project could best be tackled. It also helped the clients realize what was already available and assisted in narrowing down some of the requirements. Full details of the survey can be found in [73].

4.5.5.4 Market Survey of Potential Users

It was decided to carry out a market survey as part of the requirements gathering and domain analysis process. This took place in early 2001. Ginige and Murugesan [28] recommend that the stakeholders of a system be clearly identified in order to aid requirements gathering. This identification process was carried out partly by interviews

with CACDP staff members and partly by carrying out a market survey of potential users.

Two sets of potential users were identified. The first group were those people who subscribe to the CACDP quarterly newsletter, *The Standard*, as they were easy to access and already have an interest in the organisation and the services offered. The second group consisted of candidates for the Deaf Awareness Examination during January and February 2001. These groups covered the main interest groups – i.e. examination candidates and tutors/assessors/interested parties who are most likely to reply as they have a vested interest in new services offered by CACDP.

The survey asked questions about subscriber's current use of computers and the Internet, and about their views on a new web-based system for online learning, and also about their language use e.g. English, BSL or other languages, their hearing status, age group etc.

The survey was designed and produced by the developers, and sent out as an insert in the January 2001 issue of *The Standard*, and also given to candidates after they had completed their Deaf Awareness Examinations in January and February 2001. In total, 432 questionnaires were sent to Deaf Awareness Candidates, and 101 replies received (23.4%). 1911 questionnaires were sent out with *The Standard*, and 382 replies received (20%). This questionnaire can be found in Appendix C – Market Survey.

The results of the market survey were in the form of graphs showing percentages of respondents. These results can be found in Appendix D. These were divided between the two categories of people surveyed. Recommendations were drawn up based on these results.

The recommendations arising from the market survey were as follows:

- Timings are crucial; people will not spend a great deal more time than they are used to on the Internet.
- The predomination of dial-up users (over 50%) indicates that timings are also important due to the cost of connecting to the Internet for a long time.
- BSL users must be considered when designing the course, as they comprised about 16% of the respondents. The course should be written in Plain English to increase accessibility. Sound and video clips should have transcripts provided if possible, and no unnecessary audio should be included, nor should any be crucial to the course outcome.

- The course navigation and operational structure must be very simple and obvious, using graphics as much as possible to aid with accessibility.
- Comments indicated that users were most concerned about the lack of interaction when using a web-based course. This needs to be addressed in the design of the course content.

4.5.5.5 Issues Identified in the Requirements Gathering Stage

The requirements were gathered using a mixture of informal interviews with the stakeholders, the results of the market survey and domain study, and brainstorming by the developers. Stakeholders were identified, and then asked questions as and when the answers were required. Stakeholders were interviewed individually and informally. This caused a lack of communication and therefore a lack of discussion between the stakeholders and led to confusion and confliction over the requirements.

The information gained from the Domain Investigation phase was very useful as it allowed the stakeholders to see what other types of software were already in existence, and it gave them ideas about which functionality they did and did not want in their product. It also gave the developers ideas about how some of the functions could be implemented and what was possible. Another benefit of the domain investigation was to indicate the functionality that was not required, and what did not work well in this format, for example, sign language tuition was determined to be too difficult to reproduce successfully in this media, so it was decided not to include it in the requirements for this course.

The interviewing process also introduced Requirements Creep as different stakeholders suggested more and more ideas and features, without anyone to play Devil's Advocate and refute their necessity. Requirements Creep is a colloquial term for the situation where users gradually increase the requirements with no consideration given for subsequent increases in the resources or time allocated to the project. The requirements gathering process was carried out in a very informal way. From this a requirements specification was produced. Despite the fact that there was a formal requirements document produced to IEEE standards and using UML this proved to be unsuitable for the client's level of technical comprehension and was re-written using plain English, and non-technical jargon.

However, some significant errors were made in the requirements gathering activities. The questions asked of the stakeholders were open-ended to try to ensure that they were able to specify exactly what the client required. This proved to be a mistake as

either the client did not know what was possible and therefore tended to not be sure what they wanted the package to do, or they specified so many features and functions that made it difficult to reduce them to the base requirements.

The Requirements Document produced was too complicated and formal for the organisation and had to be re-written to suit their level of understanding. This added extra time and effort to this stage.

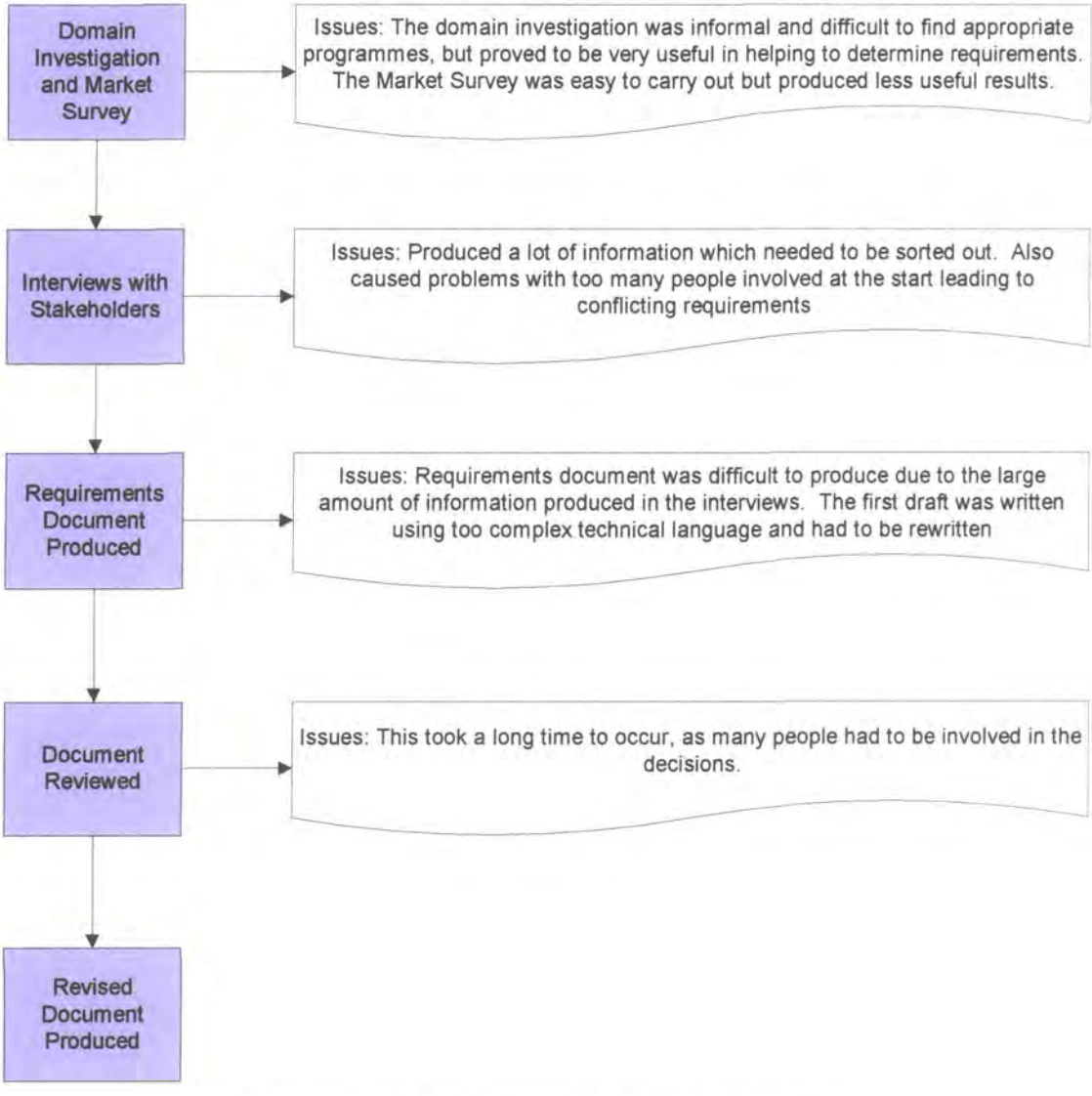


Figure 19 - Requirements Definition Process

Figure 19 shows the process that was used for the Requirements Definition section, and some of the issues that arose in this stage of the work.

4.5.6 Prototyping

During the Requirements Definition stage, it was determined that the concept of prototyping was very popular with the client, as they generally had no idea what the

technology is capable of. Showing them an example meant they could refine their requirements accordingly. Prototyping in this case may have been done too late as it was used when the majority of the requirements had been gathered, and possibly would have been of more use at an earlier stage. It was also difficult to define how complete the prototype should be, taking into consideration time and effort to create it, against it's usefulness as an aid to requirements gathering. In this project it was decided that just the interface would be designed in the prototype system as the client was not interested in how things were done, but was more interested in what the system would look like to the user. Thus an impromptu prototyping stage was introduced into the process model. This is shown in Figure 20 where the added stage is shown in red. It was included in the Requirements Gathering stage rather than as a separate stage as this was needed to aid in the gathering of the requirements.

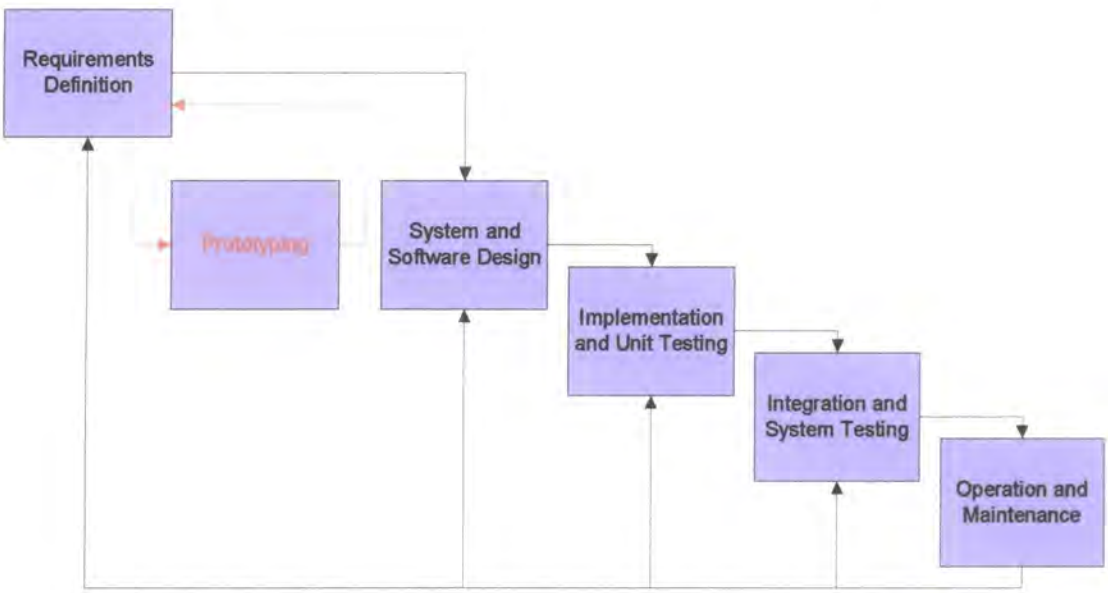


Figure 20 - Waterfall Model with added Prototyping Phase

4.5.7 Further Requirements

After the prototype was developed, the requirements were refined to take into account new and altered demands made by the client. Unfortunately this increased the amount of requirements creep as a result of seeing the prototype. This was later identified as 'vision-driven' rather than 'needs-driven' [47] as the client could now see what was possible using the technology and was basing their requirements on this fact rather than on what was required. Unfortunately this was not identified at the time due to inexperience of the developers, and the gradual nature of the creep. The client was identifying functions that they 'liked the sound of' or thought the customers 'would like'. No actions could reasonably be taken to take the new requirements into account as the

allocation of time and resources stayed the same for the project. A further problem identified was that the client is not the end-user and had done no research into what the end-user actually wanted, other than the market survey, which was carried out at the insistence of, and by, the developer. It was difficult at the time to control the additions to the requirements, and it was noted at the end of the stage that this needed to be taken into account in future projects.

4.5.8 Design of System

The system was designed in two sections, the databases required for holding the content and the user options information, and the interface between the user and the databases. The design process was carried out based on the requirements identified in the earlier stages, and based on client feedback about the prototype system. The design was started by setting out the Course Framework. This identified the aspects of the database that would be required and also set various aspects of the user interface such as the order in which sections of the course would appear to the user.

4.5.8.1 Course Framework

The course consists of a number of objects, which are defined (in terms of each other) below. These will be stored in a database. The course framework is shown in Figure 21

- *Course* – Each *course* consists of a number of *sections*
- *Section* – Each *section* consists of a number of *pages* or *activity pages*, and finishes with a *test*
- *Page* – Each *page* consists of a *title* (compulsory), some *text* (compulsory), one *picture* (optional) and one *video clip* (optional)
- *Test* – Each *test* consists of 3 *questions* in any combination of types.
- *Question* – Each question can be one of three types: *Text question*, *Checkbox question* or *radio question*
- *Text Question* – Each *Text Question* consists of the text of the question, and the text of a correct answer
- *Checkbox Question* – Each *Checkbox Question* consists of the *text of the question*, and one or more *answers*, of which one or more must be designated as 'correct'
- *Radio Question* – Each *Radio Question* consists of the *text of the question*, and one or more *answers*, of which one (and only one) must be designated as correct.
- *Activity Page* – Each *activity page* consists of one or more *activities*
- *Activity* – Each *Activity* consists of the text of the *activity*

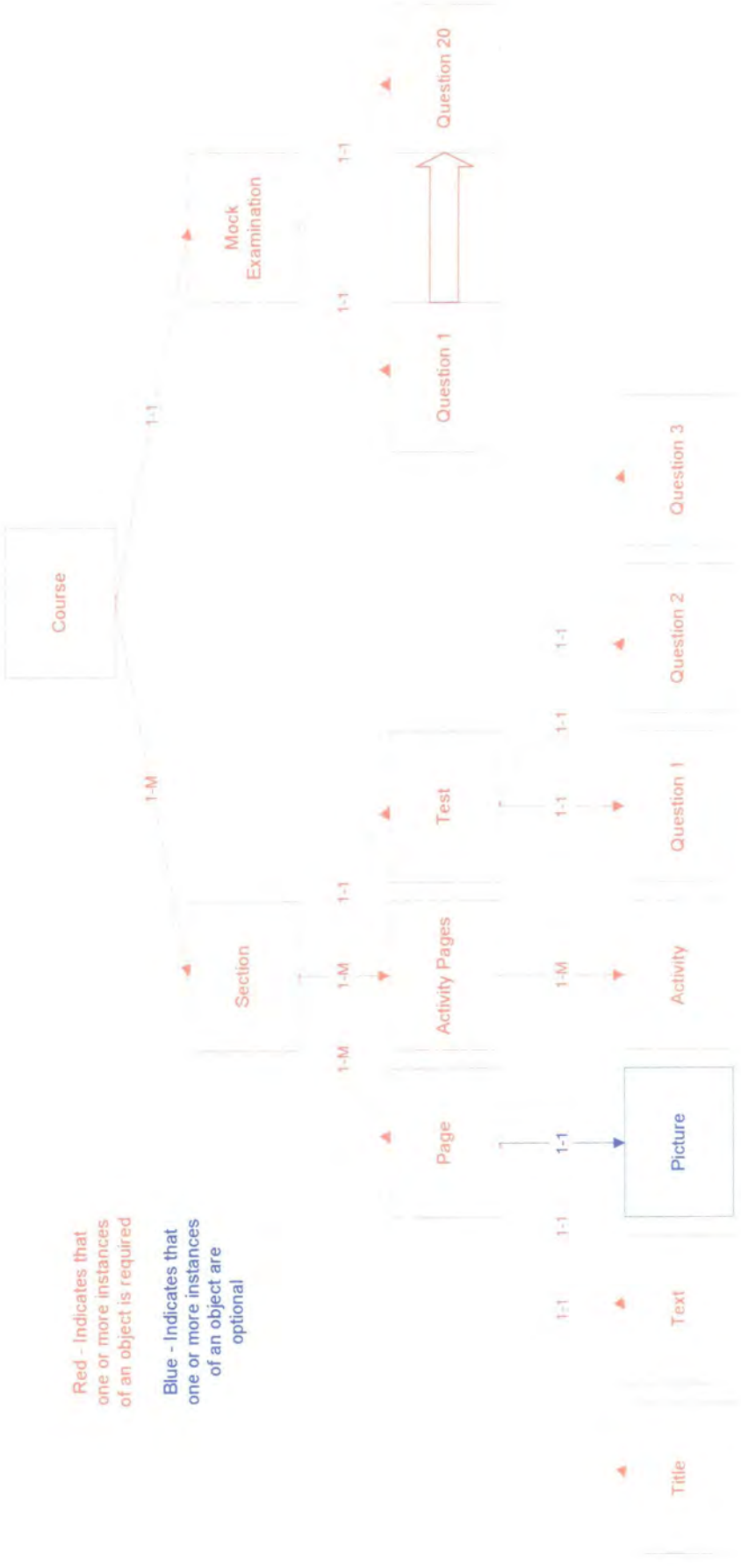


Figure 21 - Course Framework for Deaf Awareness Course

The course framework is shown in Figure 21. The red boxes indicate where at least one instance of an object is required; blue boxes indicate where one or more instances of an object are optional. 1-M indicates a *one-to-many* relationship, for example, a section can have many pages. 1-1 indicates a *one-to-one* relationship, for example, a page has one title.

4.5.8.2 Database Design

The databases were designed fully as these proved to be somewhat complicated to visualise. There are two aspects to the design of the databases – the tables and the forms. The database tables are where the information is held. For example, the Deaf Awareness database had the following tables:

- Activity
- Activity_Answers
- A_Options
- Correct_Answers
- Last_page
- Page
- Question
- Question_Answers
- Q_Options
- Section
- Text_Answers

Each table was designed in terms of the data that needed to be contained in it and the format that data needed to take. Relationships between the tables were established, and forms to fill in the tables were designed. Forms were needed because CACDP do not have the skills to deal with the tables directly and it was part of the requirements that the maintenance be simple.

Three databases were required for Project One – the subscription database holding the information about all the subscribers; a User Options database which holds all the information about which colour the user selected etc; and the content database that contains the course content.

4.5.8.2.1 Design of the "Page" Table

The Page table contains the following items of data. The format for each item was also determined:

- Page_ID (numeric, unique identifier)

- Section (numeric)
- Title (Text, max 250 characters)
- Text (Memo)
- Image (Text, max 50 characters)

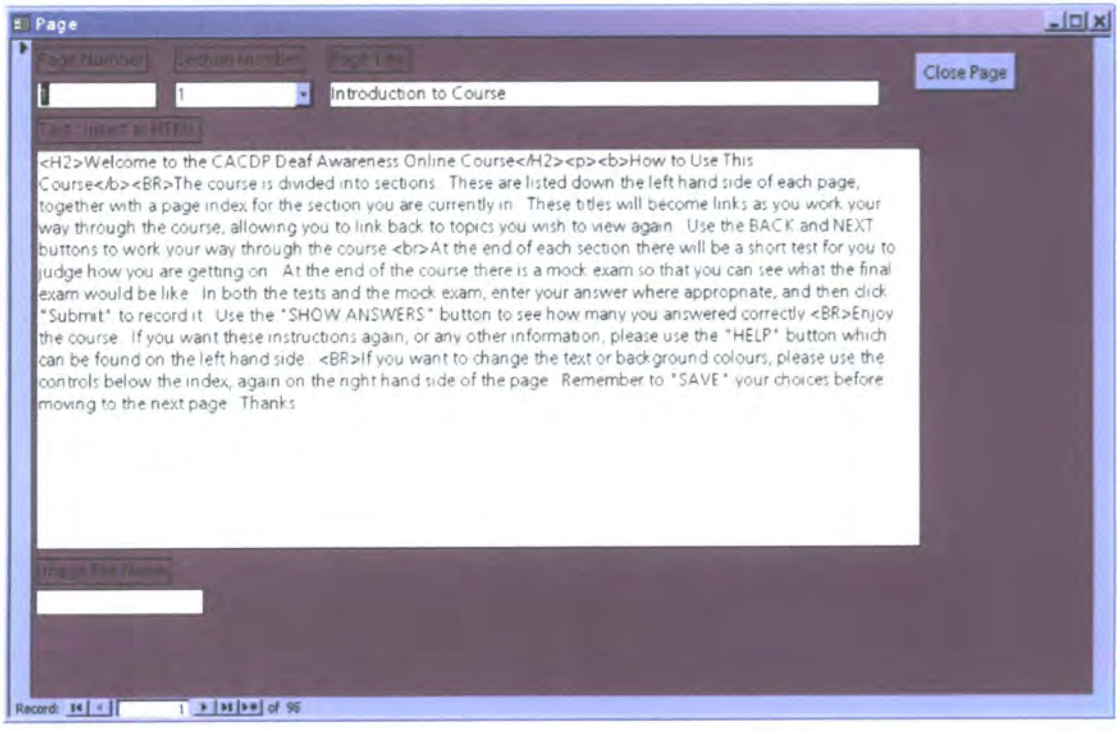


Figure 22 - Form Design for "Page" Table

Each table was designed in a similar manner, specifying the data formats and the design of the input form – an example of which is shown in Figure 22.

4.5.8.3 User Interface Design

The user interface also received considerable design effort. The requirements for this were that it must be simple and "clutter-free" so that it was accessible for as many people as possible. Standard user interface guidelines were followed to ensure that the design was suitable. The accessibility guidelines set out in section 4.5.3 need to be met to ensure that CACDP are achieving their objectives with regard to accessibility. Figure 23 shows the initial page for the Learning Package.

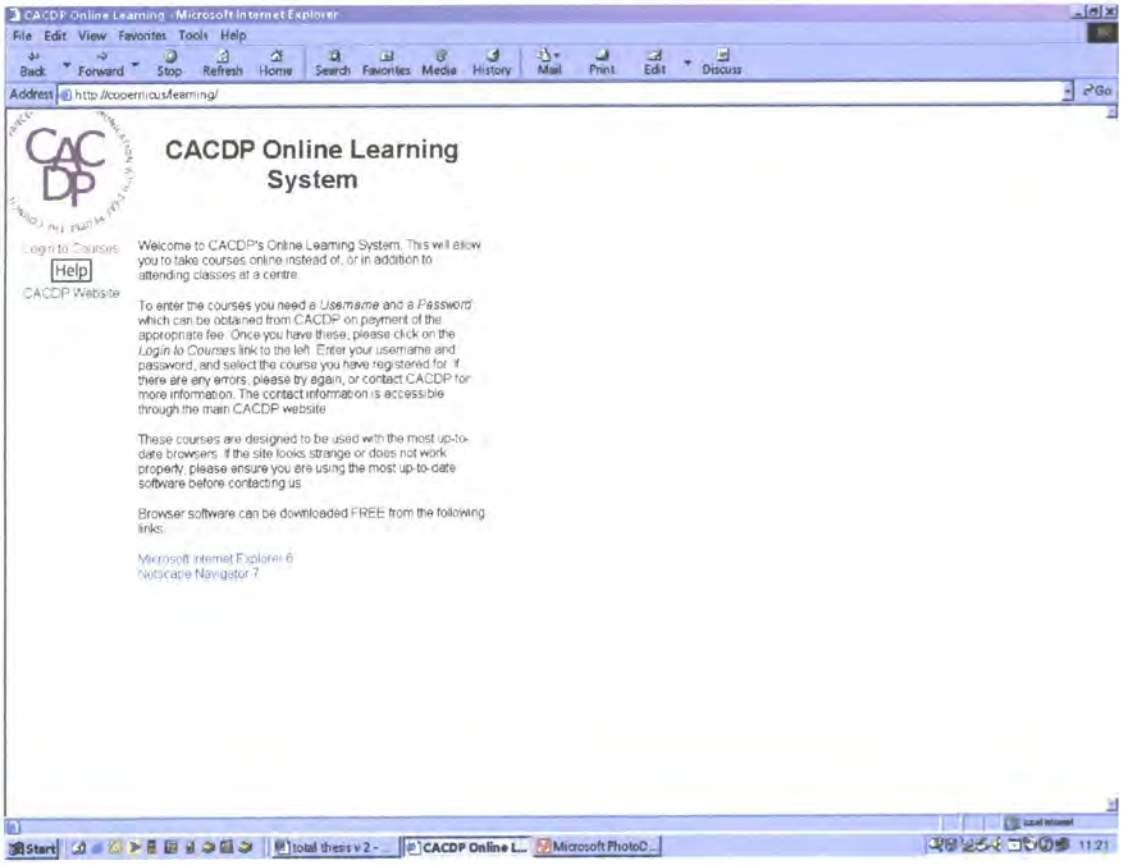


Figure 23 – Initial Directory Page

4.5.9 Software

As part of the overall project requirements, CACDP wished to improve both their development software, and their internal skills in web development. It was decided to use Macromedia Dreamweaver for the development of the systems. This package was chosen because it is easy to obtain, there is a lot of support for it in terms of both manufacturers support, and also in terms of books and reference materials. Dreamweaver is suitable for use with ASP, which is provided by the web host. CACDP will also be able to import their existing web site into the package. The author is familiar with it so the learning process will be less than with a different package.

Microsoft Access 2000 was chosen as the software for the databases. This was chosen because CACDP already make extensive use of it within their other administration and examination systems, and can therefore maintain and understand the databases created using it. It is suitable for use with ASP and HTML.

4.5.10 Implementation

An external developer was employed to implement the DA package due to time restraints. The implementation was based on the requirements defined earlier and the

design produced in the previous stage. The implementation stage identified problems in the requirements and thus the design, and highlighted other issues where decisions had either not been made, or which later proved to be impossible to implement given the time, skills and resources available. For example, CACDP wished that the tests and the mock examination be marked as to whether the answers were correct or not. This proved to be impossible as some of the questions had text-based answers and there was not the technology, time nor skills available to complete this requirements within this project. Thus, during the implementation stage it was decided to change the requirements and provide sample answers for the text-based questions, and show whether the multiple choice questions were correct, or if not, to show the correct answer(s). This was acceptable to CACDP and the system was changed to meet the new requirements.

A top-down approach was used to develop the system. The general functions and pages were mapped out, and tested as a framework, and then gradually the details of the individual actions were filled in. This allowed mistakes to be identified very quickly as functions were not considered complete until they were bug-free. As the system increased in size, so more testing was required to ensure that the functions worked together correctly. This work was carried out successfully, as each new piece of code was tested in the main system as it was developed, thus reducing the number of errors and reducing the time required for system testing later on.

4.5.11 Testing

The system was tested as it was developed, thus reducing the need for testing at this stage. System testing was carried out at various points during the development phase, and then again at the end of the development stage to ensure that there were no bugs in the system. This was white-box testing as the testers understood the system and were testing the internal structures and processes.

Field-testing was carried out on a number of sets of people. These tests were carried out under field conditions, as testers were given a username and password and asked to simply try the system, as would any standard user. This was the black-box testing, as the testers had no information about how the system should work. They were asked to report any problems that they came across. They were given no more and no less information than any standard user. This proved to be a success as testers reported back any problems, and any comments or criticisms that they had about the system.

The sets of users tested ranged from Academic Staff, to CACDP Staff members, to those who work in the examination centres. Both hearing and deaf users also tested the system to ensure that it fulfilled the requirement that it be suitable for all users. The results of the testing phase determined that various aesthetic aspects of the course needed to be altered, but technically it worked successfully, and the aesthetics could be changed quite easily.

Some specific problems were:

- Students identified questions that did not relate to the current topics – this proved to be an error in the original course text. It identified the need for testers who are not specialists in the domain.
- Students identified a need for navigation buttons at the bottom of each page as well as at the top – this was rectified, and identified a need for non-technical users to test the product.
- Students identified occasional problems with the layout on different operating systems – this shows the need to test a system on different platforms and operating systems before release.

4.5.12 Maintenance

As the maintenance stage is still ongoing, and always will be whilst the products are still in use, it is difficult to say that this stage was a success or not. The system was beginning to require maintenance to alter the content, even before it went live. This was being carried out by the staff member who will be responsible for the software once it is launched. These maintenance activities are the simplest set of tasks. The more complex ones involving modifications to the code or the interface to the database will require CACDP to acquire new personnel with the appropriate skill level. After some consultation, it was determined that it would not be appropriate to attempt to train the existing staff members to maintain the code as they did not have the appropriate skills, and would not have time to carry out any work required. They would not be using the skills often enough to make it worthwhile providing training.

4.5.13 Review of Product

The Online Deaf Awareness Package was completed successfully. It was slightly over the time scale set at the start of the project, but as it contained increased functionality to that determined in the requirements, this was deemed to be satisfactory. In relation to the recommendations made as a result of the domain survey, the results are shown in Table 2:

Table 2 - Product Criteria

Criteria	Notes	Met?
Screen Size	The package was designed to take up the whole screen.	✓
Navigation	Text based links were used to prevent any misunderstanding of icons, and also to meet the accessibility criteria.	✓
Approach	A lesson based approach was used where user progress from one topic to another. They have no choice about the order in which they cover the topics.	✓
Logical	The topics build on information gained earlier in the course.	✓
Assessments	There are short tests at the end of each unit, and a mock examination at the end of the course. Users are given sample answers to all questions so that they may determine how well they are taking in the information.	✓
Video	Video clips are not used in this course, however they will be introduced in later courses as they were felt to not be necessary due to the content of the course.	
Interactivity	Users are prompted to enter answers to tests and also to activities throughout the course.	✓
Text	The user can alter the text size and colour, and the background colour of the package to suit their own preferences.	✓
Replay	N/A in this case as no video clips used.	
Placement	Users can see an index of all the topics within a section so they can tell how far through they currently are.	✓

The system works on the new web host, and contains all the elements of the course that were required. The accessibility guidelines set out at the start of the project were followed and met with regard to the user interface. The end of module tests and the mock examination both work successfully and the correct answers are displayed when the user is incorrect, or the sample answers shown if appropriate. The content can easily be changed due to the use of database forms. The system was put through rigorous testing, both in the implementation stage, and after completion. It was tested by both the developer and by members of the public. Amendments were made to the system as a result of the testing and the system re-tested to ensure it still worked correctly. In general, feedback was positive, with the large majority of testers finding it easy to use and straightforward to work through. CACDP are pleased with the result.

4.5.14 Review of Process

There were problems with the requirements gathering stage of the process as it allowed requirements creep and was allowed to become vision-driven rather than needs-driven. It was too informal and the stakeholders did not have enough communication between themselves to make the decisions required. This lead to confusion and, sometimes, conflicting requirements from the stakeholders. This was partly due to the clients not really knowing what they wanted themselves, and waiting for the developers to tell them what was possible, and to show them possible ideas. The formal Requirements Specification was not well received and had to be re-written in a more suitable manner for the customer. This identifies a problem with the Waterfall model that the documentation is usually aimed at the technicians rather than the clients and therefore may not be well received [60]. The re-production of the requirements specification took extra time and effort and led to slippage of the timescales by several weeks.

Sommerville [5] identifies five reasons why requirements analysis can be difficult, mainly linked to stakeholders. All of these factors were experienced during the Requirements Definition phase of Project One. The factors identified by Sommerville are:

- Stakeholders often don't know what they really want a system to do, and even if they do, they may find this difficult to state. Stakeholders may be unaware of the costs of the system and therefore may make unrealistic demands.
- Stakeholders express their demands in relation to their own business knowledge. This can be difficult for a developer who is unfamiliar with the domain to understand.
- Stakeholders may have conflicting requirements.
- Political factors may affect the requirements, especially internal factors.
- The business environment may change during the course of the project, leading to changes in the requirements and new stakeholders who have different ideas.

This identifies a documented weakness in the Waterfall Model. The users are often dissatisfied as they have no input into the system between the requirements definition, and the implementation stage [60]. Therefore, it is often at the end of the implementation stage where they have their first chance to view the system and make any comments about it. By then it is often too late to make any changes and they will end up with a system that does not fit their requirements. In this case, this did not happen, but a further problem is that customers are often required to "sign-off" on

requirements before they really know what they want, and before they have seen what the technology is capable of [60]. This can be identified in this case as CACDP were not sure of their requirements at all, and this caused the requirements creep. An attempt was made to solve this by introducing prototyping to aid the clients in their requirements definition. This was a partial success, although the prototype took a long time to produce because it was added to the plan, which added to the workload. Prototyping is noted as a solution to some of the problems in the Waterfall Model [62] because it allows the users more interaction with the system before they have to complete the requirements.

Stakeholders at CACDP had not thought through their proposals completely and thus were not sure exactly what they wanted their new system to do. They were also unaware of what the technology was capable of. The domain investigation was proposed as a way of showing them other systems that are available so that they had a starting point for their own ideas.

During the implementation phase, the requirements, and subsequently the design, proved to be incomplete and this caused further problems later in the development procedure. A lack of communication, between all involved, during the implementation phase compounded these problems, as the decisions were not made by the stakeholders, as they should have been, but by the developer who possibly made too many assumptions without consulting the clients. This was difficult to surmount as any decisions required from the clients had to be made by management so a quick decision was difficult to attain. Therefore it was far easier for the developers to make an assumption, than to ask the client. Part of the problem in getting the requirements completed was due to internal communication and decision-making policies within the company. A lot more time was required for a decision to be made than was originally allowed for in the planning for the project. This was a combination of the developer not being familiar with the policies within the organisation, and the organisation having a significant number of immature processes that required time to complete.

The development stage for this package was divided between two developers. The first section of the development was contracted out to an external developer. After this it was passed back to the author, who had gathered the requirements and written the design, to complete and ensure that it worked. This was difficult as there was little communication between the two teams. The main problem was, as with the large majority of software, the code was not commented or documented which led to a loss of development time while the code was being analysed. The system had become

legacy code before it even went live. This is a common problem that highlights the necessity for small companies to improve their development processes. This two-stage development also identified the problem that the design document was not complete and had a number of omissions and conflictions. This was a problem as several design decisions were made based on the information given, which meant that some requirements could not be fulfilled as a result, or others had problems that surfaced later on.

The implementation stage was eventually successful after a number of problems. The development techniques of incremental development and top-down development were put into practice, and these proved to be very suitable for this type of application. They allowed the system to be tested as it was developed, thus ensuring that the code was complete, and also reducing the time required for testing later on. As the majority of the code could be written in small, independent sections, this was a very successful way to develop it.

The testing stage was considered a success, as the system testing and field-testing were both carried out, and feedback was obtained that enabled detailed improvements to be made to the system. These improvements were carried out and the system retested.

The general process worked well, but some of the specific stages contained problems that should be addressed in the suggested improvements.

4.5.15 Suggestions for Improvement

Stakeholders should be identified at a very early point in the process, and it should be ensured that they are useful stakeholders rather than simply those who are interested or who think they should have an opinion. This is very difficult for external developers to determine. Provision should be made for the stakeholders to have formal communication with each other; most usefully this should be in the presence of the developer, as this will allow a more structured and formal requirements gathering process. Documents should be developed that allow the stakeholders a chance to review what they have already decided or stated, and what there is left to decide. This is important, as in many commercial organisations, people not directly involved in the development make these decisions, and thus they will need reminding, as this will not be their sole project. More time should be allowed for decisions to be made and they should be set out in the most simple way possible so that a minimum amount of time

can be taken to get to a decision, even by those not directly involved in the project on a day-to-day basis.

Questions should be asked which provide choices for the client, rather than leaving them open-ended. This provides a structure for the client with which to start and should produce a more specific requirements specification. This requires that the developer has some knowledge about the domain, or that they work closely with one or two stakeholders who have this knowledge. This can cause problems if those few stakeholders start to push development in the way they wish it to happen rather than taking all views into account, however, it is the responsibility of the organisation to provide suitable stakeholders for this activity.

The prototyping should be incorporated into the requirements phase if possible or between the requirements and the design or implementation phases as this allows problems to be identified and solved much earlier in the process.

More time should be spent on the design stage of the process. Due to limited time in this project, the design was not completed as fully as it should have been, and issued were raised during the implementation phase that should have been decided upon during the design phase.

In summary, the product was a success, but the process needs to be improved before it will be satisfactory for CACDP to use. The product met the requirements, and fulfilled the guidelines set out for accessibility. It was developed slightly over schedule so more control is needed to ensure this does not happen in future.

4.6 Project Two: Directory of Human Aids to Communication

4.6.1 Description of Project

The aim of Project Two is to design and implement a web-based directory of BSL Interpreters and other Human Aids to Communication (HACs). Users should be able to subscribe to the Directory, and the subscribers' information should be held on record. The Directory should only be accessible to people who have paid to access the information.

Currently, the Directory is produced on an annual basis in book form. This contains approximately 350 HACs in four categories, and also information on Agencies who can put users in contact with HACs. Information held about each HAC includes name, address and contact details as well as data on when they are available for assignments, and what types of work they are and are not available for. For example, some HACs do not want to be contacted about assignments that are religion related, or court work, or work with children. These preferences must be included in the data.

4.6.2 Process

The process used to design and implement the Directory Package was based on that used for Project One, incorporating the improvements suggested by the analysis of the procedure. A more formal prototyping section was incorporated, and the project was implemented in stages. Changes were also made to the requirements gathering process in order to reduce the aspect of requirements creep. The diagram in Figure 24 shows the changes made to the process used in Project One, the changes are shown in red.

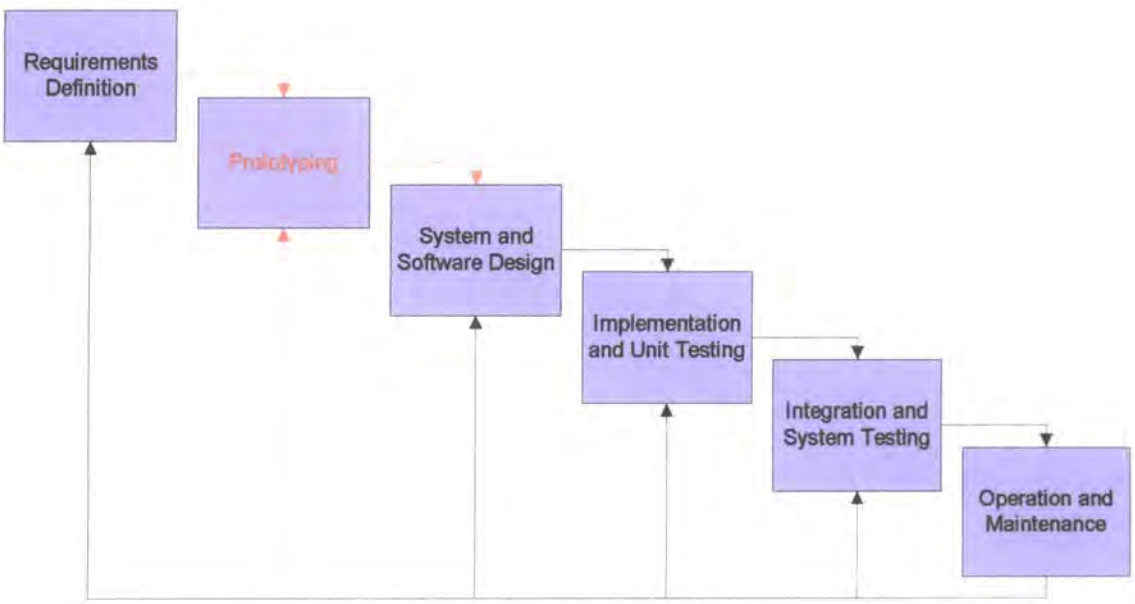


Figure 24 - Improved Waterfall Process

4.6.3 Requirements Gathering

As part of the review of the process used in the requirements gathering for Project One, it was determined that the domain investigation was very useful and that it should be carried out again in relation to Project Two. This should be carried out in conjunction with interviews or brainstorming sessions with a limited number of

stakeholders in order to reduce the communication problems noted in Project One. As a result of these sessions, documentation should be produced that would allow the management to make the appropriate decisions. This time, the documentation should, where possible, contain various options that the stakeholders can consider when making decisions, rather than leaving the questions too open. This will hopefully reduce the requirements creep noted in Project One, and should make the decision-making process easier and more efficient.

4.6.3.1 Domain Investigation

In Project One the domain investigation was found to be very useful. While much of the information gleaned from the surveys could be directly applied to Project Two, it was felt that further information was required about the new domain. An interview was conducted with the developers of a similar web-based directory, that of the Institute of Linguists (IoL). This allowed the client to foresee some of the potential problems and to ask about solutions before they arose. The interview proved to be of more use to the client in terms of organisational process changes that may be required, than the developer, as the IoL had contracted out the development of their directory and thus did not have any technical knowledge to impart.

In addition, other Online Directories were surveyed in order to give an idea of the type of services available and how they worked.

4.6.3.1.1 The Institute of Linguists Directory

This can be found at http://www.iol.org.uk/ under their 'Find a Linguist' service. It offers users a service for finding an Interpreter as shown in Figure 25. Users are asked to enter the details of the assignment, and they are presented with the details of appropriate interpreters. The Institute of Linguists deals with interpreters of many different languages hence they have an option for selecting which languages interpretation is required to and from. This form offers many options to select from and has a simple user interface that clearly shows to the user, which sections they need to fill in.

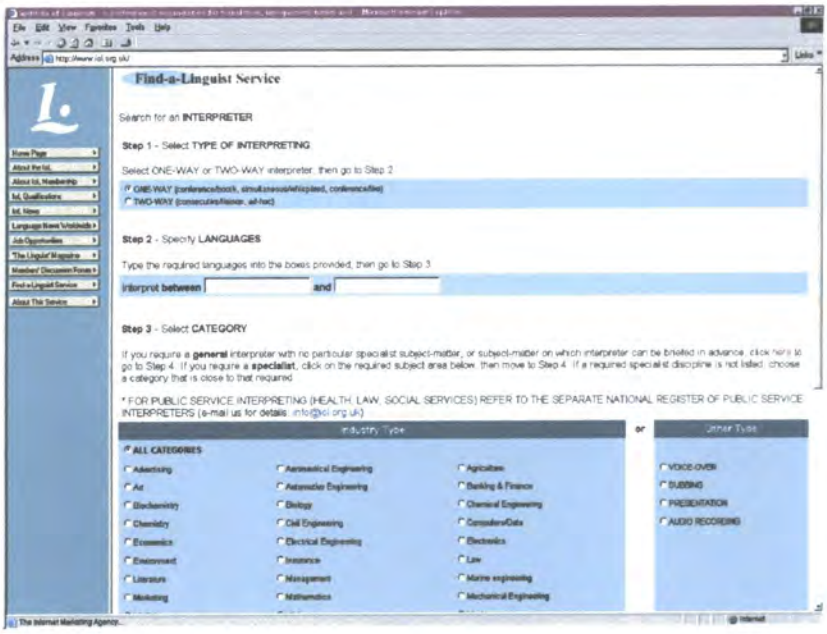


Figure 25 - Find a Linguist Service

4.6.3.1.2 Registry of Interpreters for the Deaf

This can be found at http://www.rid.org/ and offers a service for locating RID members as shown in Figure 26. Users are asked to provide information as shown, and are presented with the details of interpreters that match the criteria. This directory is not an advertising service, and therefore only lists information about the members rather than offering a service. It has a very simple interface with no extraneous details to confuse the user.

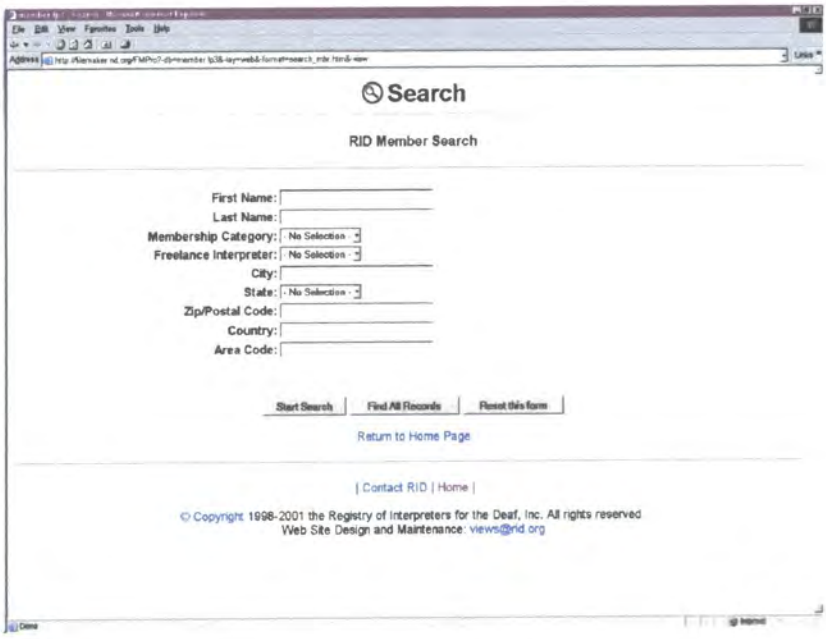


Figure 26 - Search for RID members

4.6.3.1.3 National Association of Judiciary Interpreters and Translators

This can be found at http://www.najit.org/. Users are asked to enter their requirements for a translator, including the language, location and credentials needed. The user is offered the chance to search by language, or by name if they know the name of the interpreter they are looking for.

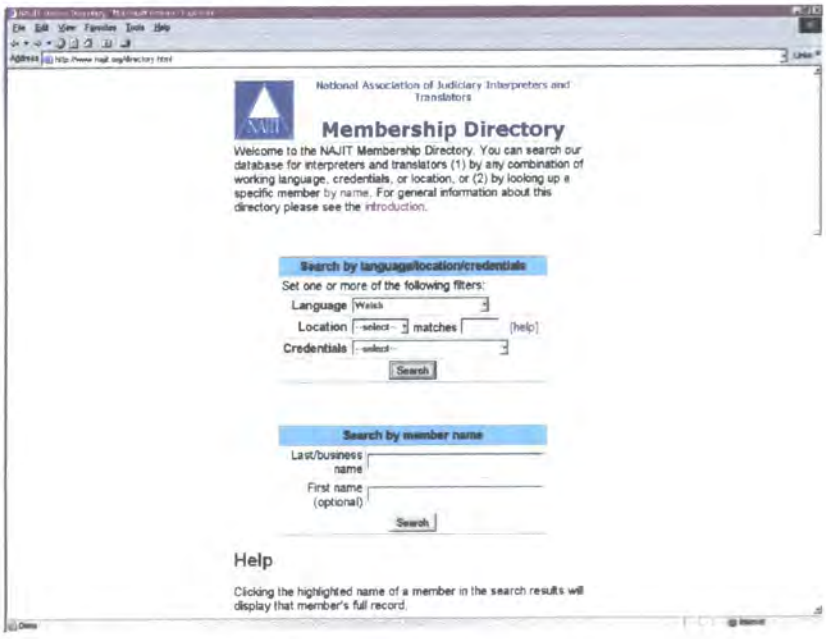


Figure 27 - NAJIT Search Form

As a result of the problems in the first project, the requirements gathering process was more closely controlled in this project and the new process was carried out as shown in Figure 28.

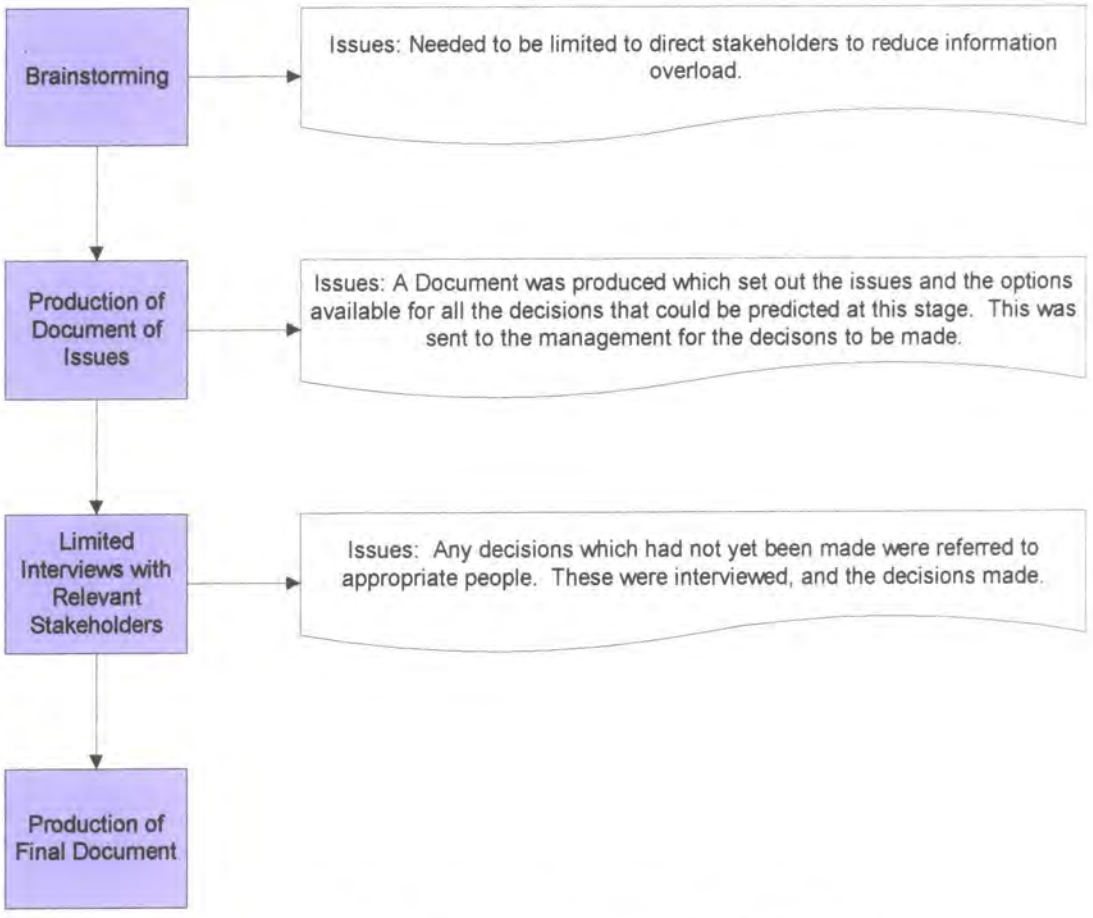


Figure 28 - Requirements Process for Project Two

First, a structured brainstorming session was carried out between the developer and the major stakeholder. This resulted in the production of the "Report on Functionality from the User Point of View", which raised many of the issues that would later need a decision made on them, for example, which data would be displayed and how. The report set out the functionality of the system from the point of view of a user, as the majority of stakeholders in this case would be involved in the system from this angle rather than from a developers point of view, and would not have understood the technical language and decisions required for this work, and were not interested in how the requirements were to be carried out.

The decisions that needed to be made were included in the document, along with options where appropriate. This document was then the subject of a formal meeting between the developer and those members of staff with the most knowledge about the existing procedures. This meeting, along with other less formal discussions, resulted in a formal "Software Requirements Specification" document. This method reduced the amount of communication required as the stakeholders were interviewed together, and also reduced the problems caused by a lack of communication between stakeholders. The Functionality Report raised many issues early on in the process and this was an

improvement on the requirements gathering stage in Project One as the issues could be identified and solved much earlier in the process. Less time was wasted on waiting for decisions, and the stakeholders were more satisfied with their earlier involvement in the decision-making.

4.6.4 Prototyping

As a result of the identified need for prototyping in Project One, it was included in the process model for Project Two. This will help to reduce the problem of the users not being able to see what the system can do. This is an identified problem with the Waterfall Model [60] and caused problems in Project One, thus it needed addressing in Project Two. Five prototype user interfaces were produced and shown to the client in order to allow them to see what the system could look like before any decisions were made. It was decided to concentrate on the interface for the prototyping, as the client was only interested in the aesthetics of the site, and not in the technical aspects of the code. Thus, as long as it fulfilled the requirements, they would be satisfied. The five interfaces showed different ways of arranging the contents on the page and gave the client ideas about what their site should look like. The clients found this very useful, as they had not previously been able to imagine what the Directory might look like.

4.6.5 System Design

A system design was produced in a similar manner to that produced in Project One but in more detail. The design was divided into two strands – the databases required, and the interface between them. Due to the limited time spent on design in Project One, it was determined that a greater amount of effort should be allocated to design in Project Two in order to reduce the mistakes made and the issues raised during the implementation stage. A formal design specification was drawn up using UML. This was most useful for the developer as the organisation found it was too complicated for them, and were not concerned with the more technical aspects of the development, although the pictorial aspect of UML appealed to them rather than the more textual design methods employed in Project One.

4.6.5.1 Formal Design Specification

A formal Design Document was produced using Unified Modeling Language (UML). This ensured that more time and effort was put into the design document, in order to reduce problems during the implementation phase. The UML diagrams produced showed the relationships between the various entities identified as part of the system. For example, as shown in Figure 29, an Interpreter will submit a registration, which will

be processed by a Staff member. A Person is a member of the general public, those who will use the Directory.

4.6.5.2 Definition of Entities

Person – someone who will use the Directory

Interpreter – someone who has information within the Directory

Staff – someone who processes any information within the Directory

Subscriber – a Person who has paid money to gain access to the Directory

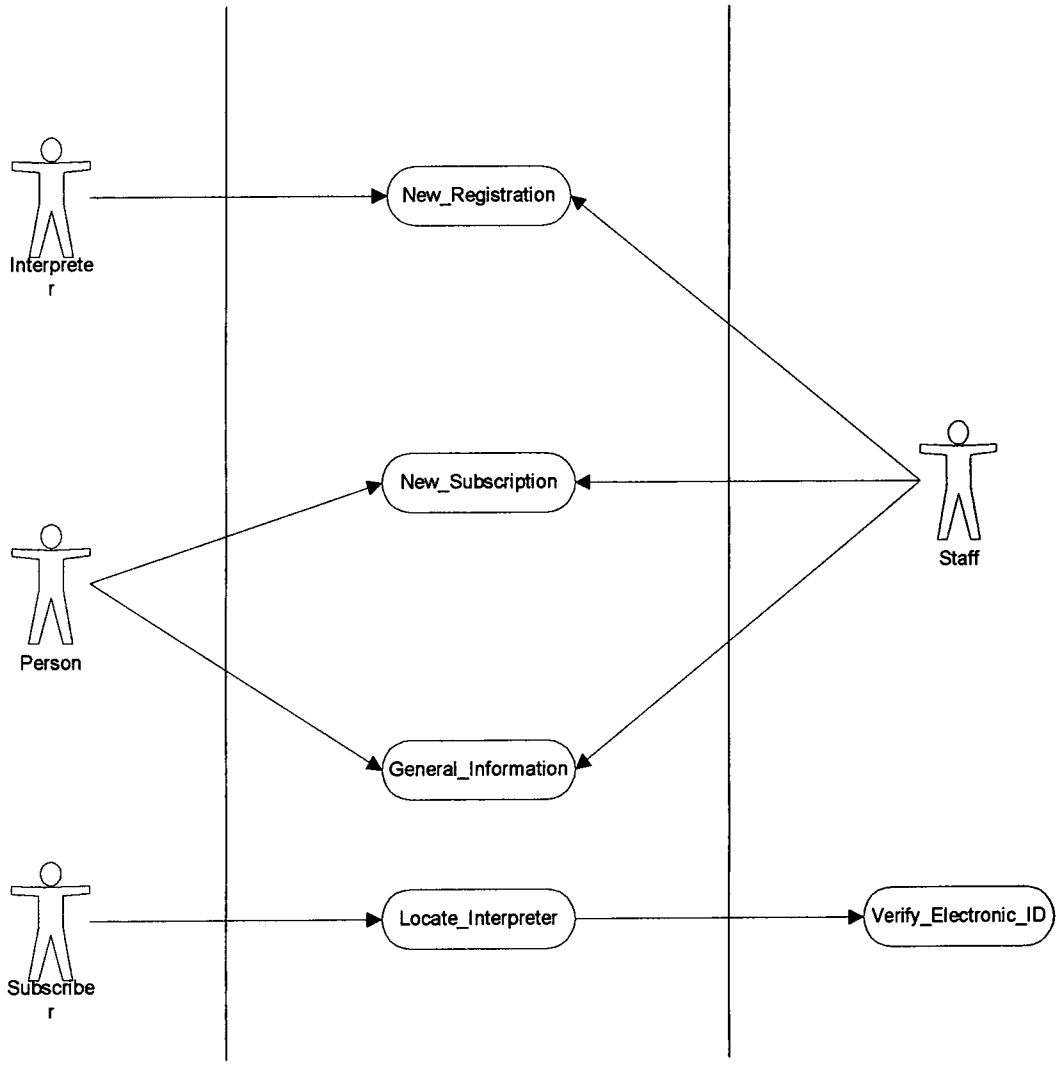


Figure 29 - UML Diagram for Directory

4.6.5.3 Activity Diagrams

These show a simplified look at what happens in a process. For example, as shown in Figure 30, a form is sent into CACDP, where the Admin Staff and the Accounts Staff deal with it. The data is entered into the system and the money paid in to the Accounts Department. A receipt is issued and a password sent to the new Subscriber. This diagram shows an interaction between a Person and two of the departments at CACDP.

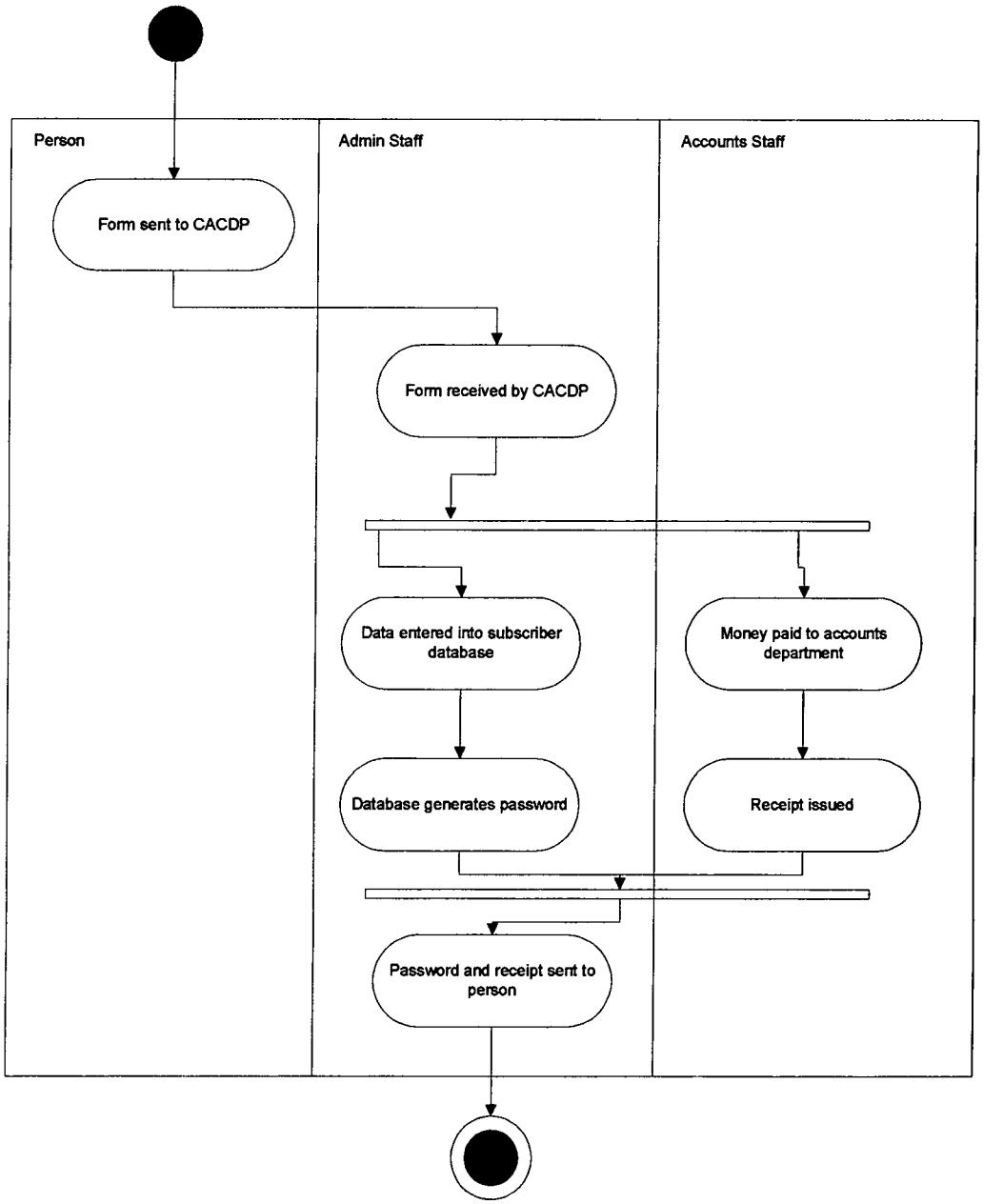


Figure 30 – Activity Diagram for the "New Registration" Process

Figure 31 shows the interaction between a Subscriber and the System when the Locate_Interpreter activity is occurring. UML diagrams help to ensure that the system will achieve its objectives, and will also make implementation more straightforward as the procedures have already been prepared and can be used as a framework for the code functions.

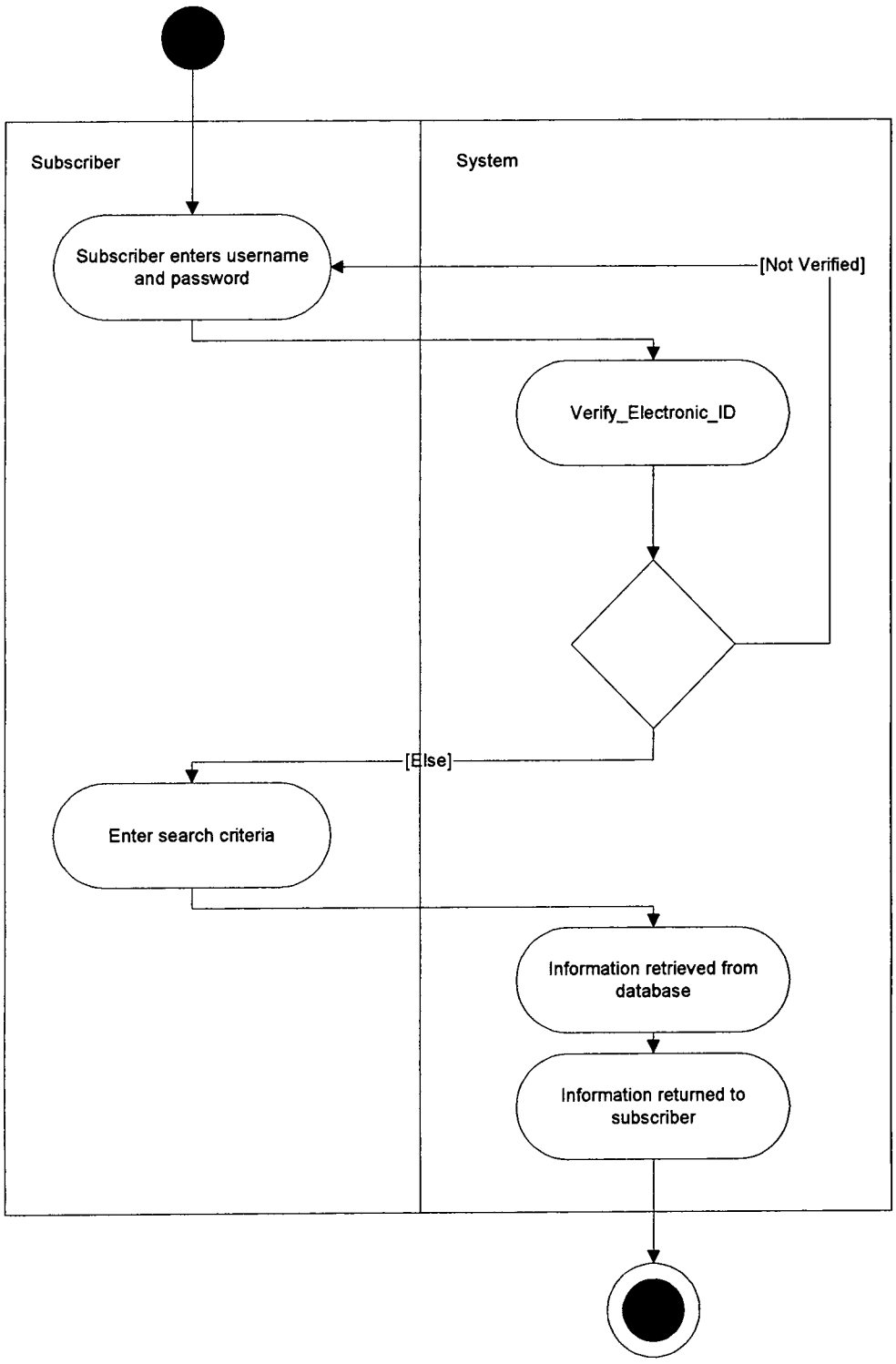


Figure 31 - Activity Diagram for the "Locate_Interpreter" Activity

Activity diagrams were produced for all the processes carried out within the system – not just within the computer system, but also within the organisation where processes could be affected by the new system. This helps to identify which parts of the process can be improved by the use of a new system.

4.6.5.4 Version Description Document

A Version Description Document was also produced during the Design Stage. This described four incremental versions of the Directory that were to be developed. This would allow the client to ensure that their requirements were being met, and would allow the developers to make any changes to the requirements as the client decided upon them. This would help to reduce the effect of the problems identified in Project One and documented in [60] where the user does not have any input into the system between the requirements phase and the end of the Implementation phase. The introduction of incremental development should allow them to see the system at several pre-agreed points in the process and ensure that the development is proceeding upon mutually agreeable lines. The four versions decided upon were:

Version One (V1)

- Scripts to retrieve options from the database for input forms created and working correctly.
- Templates created for the user interface and a user interface created for development purposes. This should contain the CACDP logo on every page, and should allow movement between appropriate pages.

Version Two (V2)

- All functionality for the site should be completed and working correctly both for agencies and for individual interpreters.
- The system should retrieve information as required for the input options, and should produce appropriate results to search actions.
- Error handling for all forms should be correctly implemented in order to reduce the chance of mistakes being made by the users.
- Login system should be completed and tested.
- All final design issues for the interface should be decided by now so they can be incorporated into Version 3.

Version Three (V3)

- There should be no broken links.
- Accessibility aspects should be correctly implemented and the system should pass the Bobby level one standard – this will have to be tested again as the final included text will not be ready until 28th February.
- All graphics should include an ALT text description – this is for screen readers to help describe graphics.
- All links should have appropriate ALT text descriptions.
- Help files should be written and implemented.

Final Version (V4)

- All included text should be completed and properly formatted.
- There should be no spelling mistakes.
- Testing completed on finished database.
- System ready for users.
- Staff should be familiar with the new system and be able to use it appropriately.

4.6.6 Implementation of System

In order to maximise reuse, and thus reduce the development time, it was determined that the login and the subscription system developed for Project One could, in addition to the domain knowledge, be reused within Project Two with some amendments.

The system was implemented in stages using incremental development in a more formal way than in Project One. Four proposed versions of the system were specified at the start of this phase and deadlines set for the production of each version. These were set out in the Version Description Document during the design stage. The four versions were each described in terms of the functionality required, the interface and the deadline date. This implementation method allowed the clients to view the Directory at each stage, and to make changes to their requirements if necessary. A process model for incremental development is shown in Figure 32 [2].

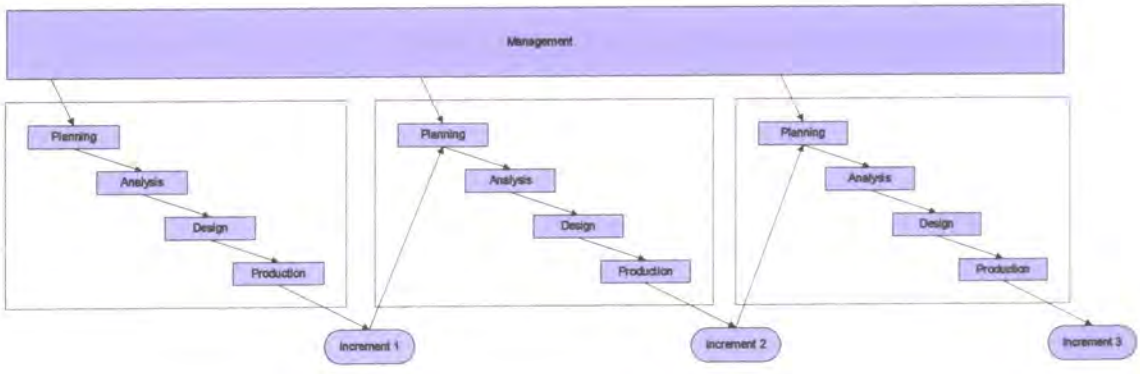


Figure 32 - Incremental Development Model

The development for this Directory was completed at the same time that the registration process for the Interpreters, whose information was contained within it, was undergoing major changes, and therefore the changes sometimes affected the requirements for the Directory, hence the need for regular checking. This took into account the need for prototyping activities without needing to rewrite the system, and introduced the ability to develop it further once the client had approved the work.

It is very difficult to implement a system at the same time as the business environment is changing as this can lead to changes in the requirements, and to the system becoming "out-of-date" before it has even been released. The incremental development approach gives more flexibility to the implementation, but can extend the time required to produce a system, as the intermediate versions are required to operate fully.

This approach was very successful from the viewpoint of the customer as they were able to see the progress of the work, and ensure that it was progressing as they wished. They were also able to make changes as needed. This approach helped to solve the problems with the Waterfall Model where users can feel disassociated with the system while it is being developed [60].

4.6.7 Screen Shots of the Final Product

Figure 33 shows the final screen design for the main index page of the Directory. As with Project One, the requirements were that the user interface design be simple and easy to understand. No special effects such as JavaScript could be used because they make a site unusable with screen readers as they hide the destination of hyperlinks. Each screen shows the CACDP logo and has a link back to the main index page. Users need to log into the site in order to gain access to the information stored in the database, but they have free access to all the General Information displayed on the site.

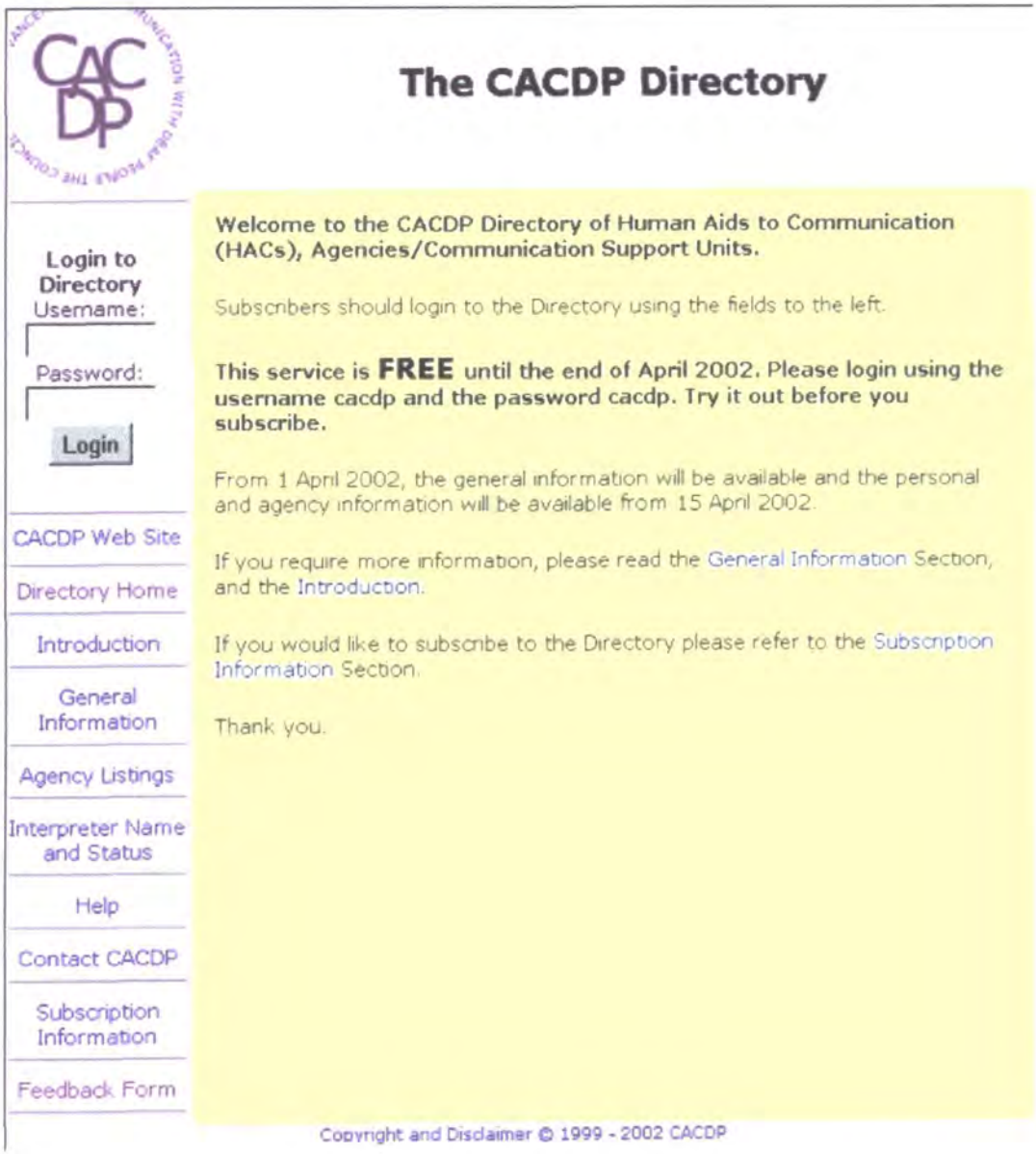


Figure 33 - Main Index Page

Figure 34 shows an Individual Interpreter Biography. This shows all the information about one of the interpreters who advertises on the system. In this case it gives their contact details, some information about the types of assignments they are willing or not willing to undertake, and some information about their training and qualifications.

Figure 34 - Individual Interpreter Biography

4.6.8 Maintenance of System

The system maintenance, as in Project One, will mainly be changing the content of the databases. The member of staff who edits the registration database also carries out the maintenance activities as these are linked to the existing database. This makes it very simple for the maintenance to be carried out and will ensure that the database and thus the system is kept up to date. It means that no specialist knowledge is required to be able to maintain the integrity of the system. However, if changes to the code or the

interface are required, CACDP will need to retain additional personnel with the appropriate abilities.

4.6.9 Testing

At this stage, staff training was undertaken to ensure that they could use the system easily as many of them would need to be able to use it in the general course of their work. This also formed part of the testing procedure. In addition to system testing by the developer, and in-house testing by staff members, the Directory was subject to field-testing by users. This was carried out in a slightly different manner to that of the Deaf Awareness Package as people are likely to use the Directory on a regular basis, rather than only once through like the learning packs.

The Directory was released onto the web site free of charge for the month of April 2001. Notices were sent to all readers of *The Standard* and to all current interpreters who have an entry in the Directory giving them the chance to try it out before they subscribed. All those with an entry were allocated a username and password as part of their package, mirroring the traditional free copy of the book.

Many people took up the offer, and feedback was obtained, much of it positive. The problems reported were mainly about the content, where interpreter's records were inaccurate due to inaccuracies in the database. These were easily rectified and users are being encouraged to use this method to continue to report any errors that they find or any changes to the data, e.g. a new address or contact number. This will allow CACDP to ensure that they can keep the Directory up-to-date and giving them a massive benefit over the previous paper versions. The only technical problem reported, other than odd reports of the site being inaccessible due to problems with the server, was that those with an apostrophe in their name could not access their records. This was rectified and a new version launched.

4.6.10 Review of Process

The development process was more effective during Project Two. The requirements gathering stage was more controlled as a result of the earlier production of documentation that detailed the issues involved, and also resulted in a smaller degree of "requirements creep" due to the limited options offered to the client. This approach would not be valid in all projects as many clients would need more input into the process and would have more knowledge of the product they want to produce. This approach meant that less time was lost on this stage, and also overall on the project. Problems with the requirements during this project were mainly as a result of the

changes being made to the overall business process at the same time, and these were reflected back into the software process. This led again to requirements creep but it was based more on the changes occurring in the system, and less on "spur-of-the-moment" changes from the clients.

The "staged implementation", with different versions of the software, worked very well as the client could view the software several times during the implementation stage, and make any changes they needed to. This method of development also served to ensure that the software was fully tested at each stage, thus reducing problems later on. The client was satisfied with being able to monitor progress in a way that they could understand, rather than being given technical reports, or progress reports.

The testing was very successful, with users responding positively and reporting errors in the data rather than technical problems. CACDP are extremely pleased with the result, and subscription orders are starting to increase.

4.6.11 Suggestions for Improvement

The process worked more successfully in Project Two. There was still a limited amount of requirements creep, and this could be reduced even more if the process was more closely controlled but this is very difficult as the client has little knowledge about the product they want, and tend to use the requirements stage as a feasibility study instead.

The requirements stage is very difficult to regulate as it relies so much on the company or organisation involved. For example, if the organisation knows what it wants, and what is possible and practical, then the requirements gathering will be much easier as there will be fewer decisions to be made, less explanations and more cohesion. However, in this case, the organisation was unsure about what they wanted, and had little knowledge about what was possible. This meant that they were learning as the product developed, and thus needed to be able to change their requirements as the project developed. This was good for the organisation but not a good idea from the viewpoint of the developers as it was difficult to plan ahead and make decisions. The version-related implementation allowed this more successfully than in Project One as the client could view the system at a defined time and make changes then to the requirements.

4.6.12 Analysis of Reuse Issues

As stated earlier, it was shown that the security systems could be reused from Project One, both the subscriptions database, and the code required for accessing it and ensuring that usernames and passwords are valid. In addition, some of the domain information gained in the domain survey could be applied to this system, particularly that relating to user interfaces, and recommendations for accessibility. The code was not all reused directly, but much of it could be modified and made use of. Also, the knowledge learnt from the implementation of Project One was a large area of reuse. Mistakes and errors made in Project One were not repeated in the implementation of Project Two, saving time and effort, and resulting in a more structured coding period. Both technical learning, and the knowledge gained from the use of the development process could be applied to this project. Due to the current informality of the reuse process, Project One needs to include a formalised reuse process especially for capturing (for later reuse) the software development process. In Project Two, the process itself was reused, with amendments, and some of the code fragments were certainly based on those used in Project One, as were the database access routines, the concepts behind the SQL statements, and the general arrangement of files and modules.

4.7 Project Three: Deafblind Awareness Learning Package

4.7.1 Description of Project

The aim of this project is to produce a similar package to that produced in Project One. It will be a package to teach and assess Deafblind Awareness (DBA). The material for this course will be produced especially for the online version, as currently it only exists as a curriculum and not as content material. The course will be similar to the Deaf Awareness (DA) Package but will extend it by the inclusion of video material, and by some other minor changes to the requirements from Project One. These are listed below in section 4.7.2.

4.7.2 Requirements Gathering

As a result of the testing and review of the DA Package that was produced in Project One, some changes were made to the requirements of Project Three. These were mainly aesthetic changes, based on differences in the content of the course. For example, the DA course contained a number of activities that appeared throughout the

text, whereas the DBA course does not contain these, so the code needs to be modified so that a text based review of the section can be shown rather than just the activities.

The number of questions for each section needed to be modified as it was originally set at three, and it was determined that future systems needed to be more flexible than that. This needed to be changed to allow any number of questions to be displayed.

The security system implemented in Project One, and amended in Project Two can be reused directly in Project Three. This included the subscriptions database, and the code needed to access it. Also, the database containing the user options could be reused within this project, even though it was not required for Project Two. There will need to be provision added for video clips to be shown on the course pages so that a video clip can be associated with a page in the course.

The course framework from Project One will be reused, but with the following amendments:

- The Activity Pages must be optional as the course used in Project Three does not use this format and other courses in the future may not either.
- There must be provision for a video clip to be included on each page
- There must be no limit on questions in each test.

The requirements gathering phase for Project Three was a much less complex task than either Project One or Project Two. Many of the requirements for Project One could be reused, with a few modifications and enhancements. This meant that the requirements were closely controlled, as there were not many decisions to be made as they had already been made in Project One. The requirements were gathered using a combination of those determined in Project One, and meetings with stakeholders.

4.7.3 Process

During this project, the development process was used to improve the existing code and reuse it for the new system. The main bulk of the code was reused as it already existed, with amendments made to improve the login and make it generic, to increase the number of questions allowed for each module, and also to add the provision for video clips to be played during the course. This involved changing both the database, and the interface code.

An optimal process for this is shown in Figure 35 [74]. Currently, the stages of 'certified' and 'classified' will not be included (thus following the dotted line) as CACDP do not have the resources to carry these out, however, in the future if they wish to build up a larger repository of artifacts they will need to put into the process some method of ensuring the objects meet certain specific standards.

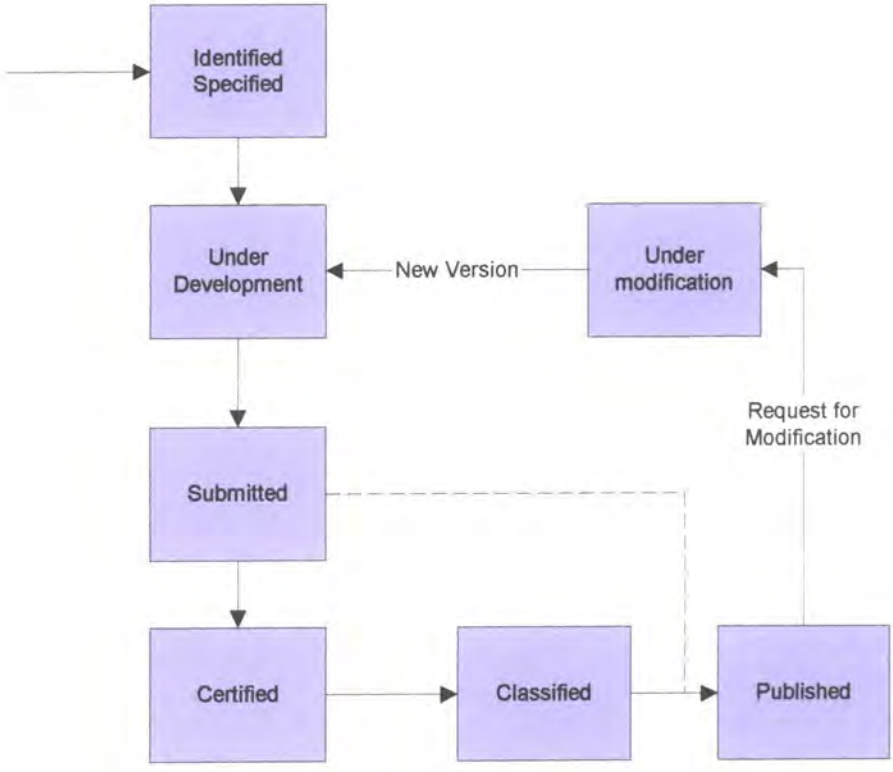


Figure 35 - Reuse Process for Software Assets

4.7.4 Design

As with the requirements, much of the design from Project One was reused in Project Three, as well as the design work from Project Two that amended the security system. The course framework used in Project One was amended as shown in Figure 36, this reused much of the design work from previous projects and saved a lot of time and effort.

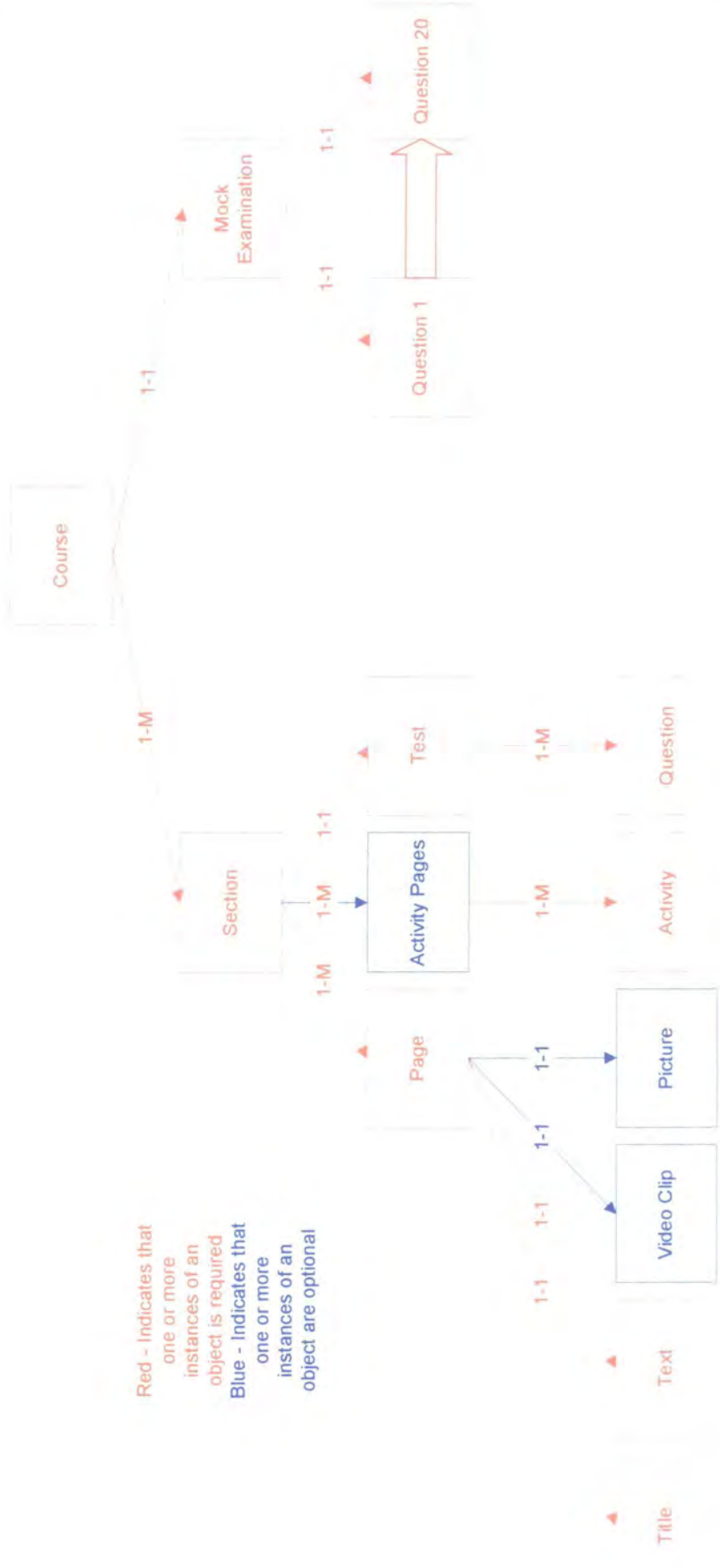


Figure 36 - Course Framework for Deafblind Awareness Course

4.7.5 Implementation

Project Three was implemented by reusing much of the code from Project One, with amendments to take account of the changes to the requirements. For example, the code that displays the mini-tests at the end of each section needed amending to take account of the new requirement that it be possible to have any number of questions, rather than 3 as was specified in Project One. Again, a top-down approach to implementation was used as the framework was put into place, and then the functions completed, and tested as they were finished. This reduced problems during the system testing phase.

4.7.6 Testing

The Deafblind Awareness Learning Package was tested in a similar way to the Deaf Awareness Learning Package. It was system-tested by the developer, and then field-tested by sample users. The testing phase for Project Three did not produce as many issues as in Project One as the amendments suggested for Project One had already been incorporated into Project Three during the implementation.

4.7.7 Maintenance

Project Three will be maintained in the same way as Projects One and Two. CACDP will be able to edit the contents of the databases using the forms provided, and if they need to make any changes to the code or the interface then they will need to use Dreamweaver to edit the code. CACDP are being encouraged to continue to make records of maintenance activities carried out as this will aid them if there are any problems, and also will allow them to have notes available for maintenance procedures that are not carried out very often. In addition to this, staff members other than those originally trained will be able to carry out maintenance procedures, as there will be documented procedures available for them to follow.

4.7.8 Review of Process

The process in Project Three worked more effectively than in Project Two. The requirements were more controlled and this led to less slippage in the time and resources required for the development of this project. The requirements for this project were based on those used for Project One, meaning that fewer decisions were required, as the organisation was not starting from nothing as they were with the first two projects. The decisions that were required were made more quickly as the organisation had the options produced for them again. This identifies an organisational benefit of reusing artifacts other than code.

A large proportion of the work done for Project One was reused in Project Three. This reduced the time needed to implement the system and was more efficient as many of the mistakes and errors had already been identified in previous projects. This indicates that knowledge is also being reused. Approximately 70% of the code was reused "as-is" and about 30% was modified to take into account the changes to the requirements.

4.7.9 Suggestions for Improvement

The main improvement possibilities highlighted in Project Three are that of a reuse programme. CACDP need to formalise their resources in this area and document them so they are certain of what is available, and how it can be used.

The development processes can still be improved through use, and the introduction of metrics to control time scales and development progress. In Project Four these can be introduced by making estimates of the time required for each phase, and then trying to identify how accurate the estimates were and which areas were wrong if the timings are inaccurate. This will allow CACDP to allocate resources more efficiently to their development processes.

The area of prototyping requires further work, as it was more successful in Project Two, but was not really needed in Project Three, as it was very similar to Project One. Likewise, the use of incremental development in Project Two. These were very useful tools when developing a new system in a new domain, but were found to be redundant when reusing much of the previous work. The customers did not need prototype models for Project Three, as they already knew what they wanted. The incremental development was not required as much of the development work had already been completed, and just needed modifications made to fit the new requirements determined for Project Three.

4.8 Project Four: Deaf Community and Culture Learning Package

4.8.1 Description of Project

The aim of this project is to produce a similar package to those produced in Projects One and Three. It will be a package to teach and assess Deaf Community and Culture.

Deaf Community and Culture covers topics about Deaf history and social issues. The course is in the process of being developed, which will allow the developers the chance to take online learning into account from the start.

4.8.2 Process

It was decided that, due to the analysis and implementation of the other two learning packs, this project was not suitable for the online media. Once the content had been designed, it was determined that the multimedia aspect of the pack would need more sophisticated resources than were available, and it was decided to develop the pack as a video rather than as an online pack.

This identifies one of the limitations of the Waterfall Model in that it does not identify risk, and as such cannot be used to state whether a project will be successful or not, but simply to implement a system [61]. A decision must be made by another method as to the feasibility of implementing a system in this manner. The introduction of the prototyping phase will help to solve this however, as developer can see when they get to this stage whether a system will work, rather than getting much further into the project and finding it won't.

Therefore the use of time metrics and further work on the prototyping phase cannot be further investigated.

4.9 Summary of Chapter

This chapter contained a description of the experimentation stage of the project. First it gave an overview of the plan of the experiment, and a description of the process to be carried out. Then, the four sub-projects were introduced. Each sub-project was covered individually, with each stage of the development described and commented upon. The sub-project was then reviewed and improvements noted that were carried through to the next sub-project.

Chapter 5: Results And Evaluation

5.1 Chapter Abstract

This chapter deals with the results of the experimentation, describes what observations were noted, and then describes the evaluation of the processes, and the evaluation of the project based on these results and observations.

5.2 Introduction

The results of the experimentation will be in the form of observations about the processes and the use of the processes, and an evaluation of the products produced. The results for each section are given and then the results are evaluated. The sections are:

- Product Evaluation
- Reuse Measurements
- Process Improvement Measurements
- Completion of Action List activities

5.3 Product Evaluation

The products produced must be evaluated in order to determine whether or not the project was a success in terms of production. The products will be evaluated in three categories:

1. How well the requirements were met
2. How many problems were shown up in the field-testing stages
3. How satisfied the company are with the finished product

5.3.1 Deaf Awareness Learning Package

The requirements were all met, although some were altered at later stages in the development.

During the testing stage, the users identified various problems. These problems were superficial however, i.e. they were not functional problems but aesthetical and could be rectified easily. The problems identified were:

- Necessity for two sets of navigational buttons, at the top and bottom of each screen, rather than just at the top

- Default colours should be a white background rather than a black background as it is easier to read
- Odd questions that were linked to the wrong topics
- Typing mistakes, punctuation etc
- Video clips would be a good addition
- Text boxes for question answers were too small
- CACDP logo too large on initial login page

These problems were all rectified before the package was released apart from the issue of the introduction of video clips. CACDP decided that they did not have any suitable video clips available at the time, and rather than delay release of the system, they would not include them at this time, but would look into the feasibility of adding this functionality. The user feedback was generally positive, and all users completed the courses. Some users even went on to complete the qualification. CACDP were very happy with the result.

5.3.2 Directory of Human Aids to Communication

This system was tested in a slightly different manner to Project One. Once the system had been tested thoroughly in-house it was released to the public on the CACDP web site. Users were offered a free months access and were asked to try it out and report any problems using the feedback form provided. A lot of feedback was received, the vast majority of it positive. A few problems were reported, but mainly these were due to inaccuracies in the database rather than problems with the code itself. The problems reported are listed below:

- Users with apostrophes in their surname e.g. O'Connor, were not able to access their records as the search routine could not deal with the punctuation
- A few people reported inaccuracies in the information displayed about them

Problems with the information held in the database were passed on to those responsible, and the problem with the apostrophes was corrected to the satisfaction of the user who reported the problem. The company were very pleased with the result, and found it to be "*better than expected*".

5.3.3 Deafblind Awareness Learning Package

The Deafblind Awareness Learning Package was tested in the same was as Project One. The system was fully tested internally, and then was released to a similar group of testers. Again, they were provided with usernames and passwords and asked to

work through the course in their own time. The errors reported are listed below. There were expected to be, and proved to be, fewer errors than in Project One, because those reported had been taken on board already and added to the design.

- Typing mistakes, punctuation etc
- Slow downloading of video clips

The Deafblind Awareness Package was well received, and the video clips proved to be a popular addition although they were slow to download over a dial-up connection. Again, the feedback was mostly positive, although some users did mention the slowness of the video clips. This was an issue raised with CACDP during the requirements stage and again after the feedback, but they felt that the positives of including video clips outweighed the problems of slow connections and decided to include them. Work was done, based on the feedback received, to make the clips as small as possible and to ensure that all material contained was relevant and that any extra material was removed to make the clips as short as possible. CACDP were again happy with the finished product, although a little disappointed about how much video they could include without it becoming too slow to view.

5.4 Reuse Measurements

In order to measure how effective the reuse procedures between the three projects have been, it was decided to measure what percentage of each system had been reused in the other two. The projects have been divided into six sections. Each section contains artifacts that can be reused, for example, documentation, code and knowledge. These sections are:

- Domain Knowledge
- Requirements
- Design
- Implementation, including code
- Testing Procedures
- Maintenance Procedures

Reuse took place between Projects One and Two, Projects One and Three, and Projects Two and Three. The graphs below show the percentages (approximate) of the sections that were reused in each project. These show the reused material divided into three categories. Some was reused "as-is", that is, it was not changed or modified before it was reused in another system. Some was reused with modifications, and some was new, developed especially for the system. For example, Project One

contained 100% of new objects. This is because CACDP had no repository of material from which to take reusable artifacts. Thus, this was to be expected at the start of the project.

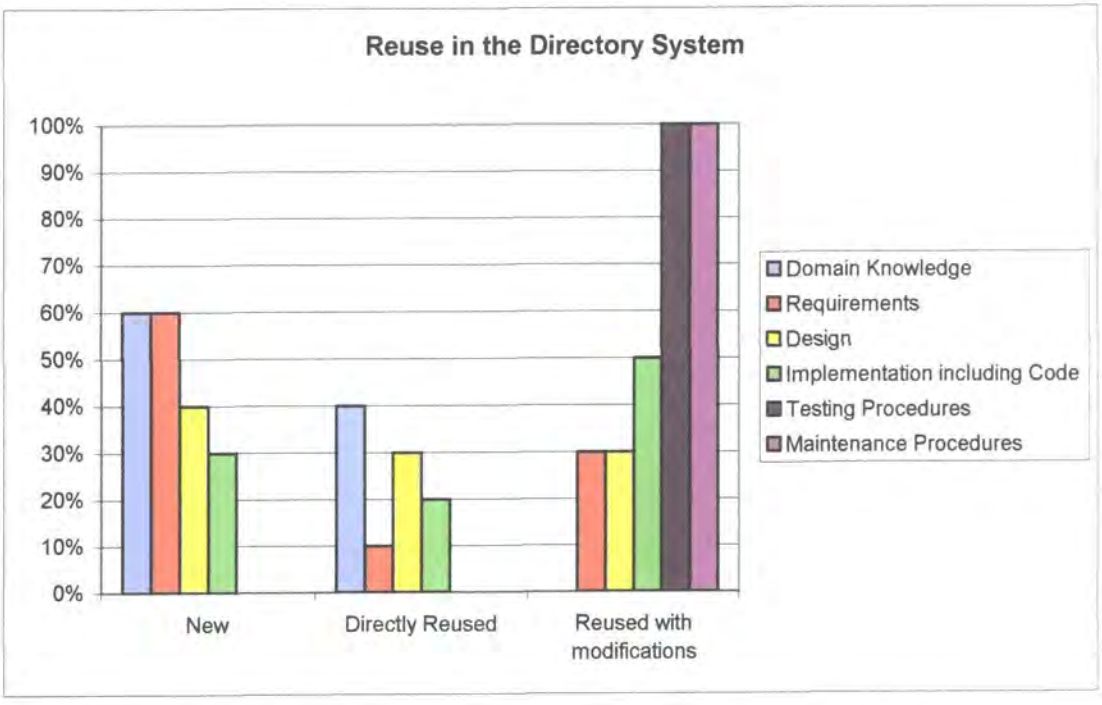


Figure 37 - Reuse in the Directory System (Project Two)

Figure 37 shows the amount of material that was reused in the Directory System. All the reused material was taken from the Deaf Awareness System. As these projects are in different domains, this explains the limited amount of reuse possible within the domain knowledge area. However, there was still reuse occurring within this area, as the knowledge gained about user interfaces would still be applicable. Much of the design and implementation was reusable with modifications as actions such as the database access routines and the database queries were very similar and were not reliant on the domain. This meant that reuse was possible in this area. Finally, the testing and maintenance procedures could be reused to a much greater extent as these were all carried out in the same way – system testing and then field testing. This did not vary from project to project, other than the instructions used for the testing, and the specifics of the maintenance procedures.

The concept of artifacts being reused "with modification" may be misleading as some were not modified a great deal, for example the maintenance procedures, and others, for example the specifics of the testing procedures were modified much more. The results show in Figure 37 show reuse from both of the earlier systems so some artifacts are taken from Project Two and some from Project One.

Figure 38 shows the material that was reused within the Deafblind Awareness System. This material was taken from both the Deaf Awareness System and the Directory System. This chart shows that a large amount of reuse was possible in all areas. The domain knowledge was mainly reused from Project One, with additions to take into account the slightly different topic. Again, the requirements could be reused almost completely from the Deaf Awareness Project, as the two are very similar. The small amount of new material required would have occurred as a result of the changes to the requirements made by CACDP, for example, the change to the number of questions allowed in the tests, and the removal of Activity Pages. Similarly, the majority of material from the design section could be reused directly as it was also nearly identical. Again, the testing and maintenance procedures could be reused directly from Project One, as the test subjects were the same, and the requirements for testing the same. Similarly, anticipated reuse levels for Project Four would also have been high as it would be very similar to Projects One and Three and would gain from the mistakes made in the development of those systems, and would also benefit from the work done.

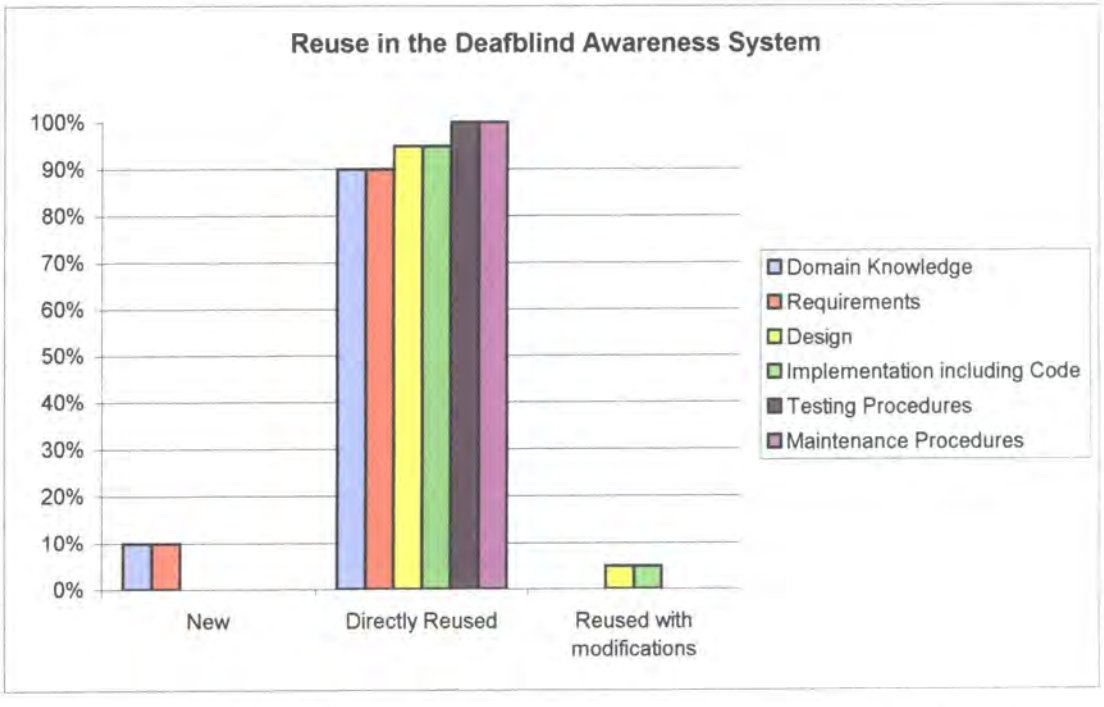


Figure 38 - Reuse in the Deafblind Awareness System (Project Three)

5.5 Evaluation of Reuse Aspects

The figures above show that the reuse procedures have made a significant impact on the development of the second and third packages. They also show that reuse is possible for many objects within the whole project, rather than simply the code.

The reuse of some artifacts is easier than others. For example, the code was easiest to reuse as it was straightforward to see whether a code fragment fulfilled the requirements, and if not, whether it could be modified or not. Other materials such as the design documents proved more difficult to reuse, especially across the domains. It was difficult to see the similarities on occasion.

CACDP now have a small repository of objects that they can reuse for any more systems that they implement, and this will increase the more systems they implement. As noted in Chapter 2, the benefits of reuse to CACDP include a reduction in development time, an improvement in standards and thus software quality, an increase in productivity and an increased competitiveness [20]. CACDP are most affected by the improvement in software quality and the reduction in development time. They are not so affected by the increase in competitiveness, as they are a monopoly in the area of Deaf Awareness Examinations. The improvements in standards are very important as they see professionalism and standards as very important in their area of work.

5.6 Process Improvement Measurements

The results of the Process Improvement activities are measured by carrying out a second CMM survey and comparing the results against those from the survey carried out during the Company Analysis. The results are shown in Figure 39, in terms of how many criteria the company meet, i.e. how many questions they could answer 'yes' to.

It can be seen that the number of criteria has improved dramatically. The graph shows that there are 49 sections where CACDP meet the criteria after the introduction of the new process, as opposed to just 9 beforehand. This shows a good improvement, although there is still work to be done. The 'Does Not Apply' category has reduced. This is because, previously, the questions related to procedures that the company did not have. Now they have a formal procedure so these questions now apply to them.

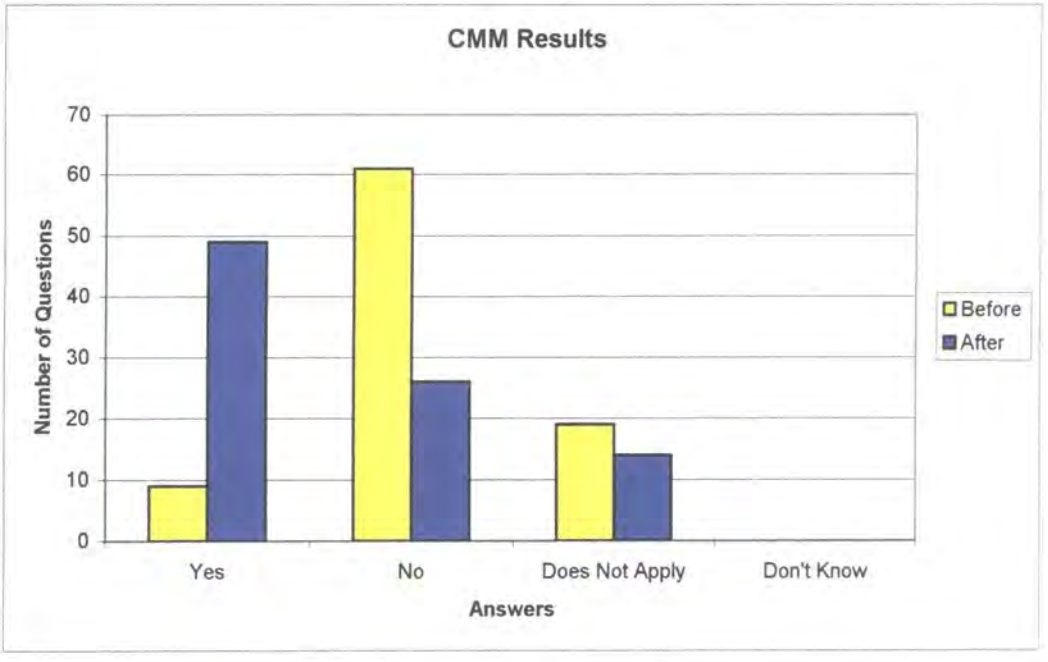


Figure 39 - Results of CMM Survey

5.7 Evaluation of Process Improvement Activities

The process improvement activities have identified aspects of the process that could be, and were, improved through the three iterations of the process. These improvements were put into place with varying levels of success. Some stages, such as the introduction of prototyping in Project Two were very successful and led to a reduction in the time needed for the requirements definition. Other improvements, for example, the control introduced over the requirements stage were not so successful the next time around, but did register a slight improvement, and then improved again in the third iteration.

Problems identified during the first iteration of the project were:

- Requirements creep during the requirements phase leading to an increase in work to be done, but no increase in time or resources allocated
- Offering open-ended questions in the requirements stage led to the requirements creep, and did not help the client to state specific requirements
- A need for prototyping to assist the client in determining their requirements
- Documents that were too technical for the client to understand
- Lack of communication between the stakeholders leading to a conflict of requirements on occasion
- Design phase was not completed fully
- Lack of documentation for the system after development by the external developer

Some of these problems are documented issues with the use of the Waterfall model.

In contrast, a number of positive aspects were also noted. These were:

- Useful domain investigation that gave the organisation many ideas before they defined their requirements.
- Successful testing period for both system testing and field testing where the system was tested by staff, and by external testers. Comments and feedback were received and acted upon leading to an improved product.
- Successful completion of the project, and satisfaction from CACDP with the result.

The problems were rectified in the second iteration with varying degrees of success. The open-ended questions were replaced with options, which helped the client to specify their requirements exactly rather than giving them too many options. This would not be necessary with clients who had more knowledge about what the technology is capable of, but was necessary in this case.

Prototyping was introduced and this helped the client, again, to see what the technology was capable of and what they could expect. This was a great success.

The documents produced in the first iteration were too complicated for the client to understand. They complied with the IEEE standards for documentation, but needed to be written in plain English and non-jargon to be useful to CACDP. The documents needed to be reproduced, extending the time required for each stage. In the second iteration, these were produced first time, saving time and effort.

Measuring Process Improvement can be very difficult. In this case, the CMM was used both at the start of the project, and again at the end to gain a measure of the improvements made to the process during the course of the project. This measurement can be misleading, as some of the results may not be completely straightforward. For example, in this case, the question about training – "*Do members of the software engineering group and other software-related groups receive the training necessary to perform their roles?*" In this project, the answer to this question went from "yes" at the start of the project, to "no" at the end. This was because, while the CACDP staff had the appropriate training for their activities before the new process was introduced, they did not at the end because the activities had changed and were more complicated. This gives a false idea as some things have actually failed to improve, and have in fact worsened. Therefore the CMM may not be an ideal way to

measure Process Improvement. However, if the responses to the questions are studied individually, rather than as statistics, the CMM will be very useful as the company can identify where they are lacking and can seek to rectify all those areas where they currently answer "no". This type of activity is more suited to companies who understand formal processes and are willing to put in time and effort understanding the criteria, and making sure they are met. CACDP do not have the time or skills for this type of activity.

5.8 Action List Revisited

As described in Chapter 3, an Action List was drawn up to address the problems identified in the Company Analysis. This contained the following activities:

- Determine a Formal Development Process
- Simplify Maintenance Procedures
- Reduce Development Time
- Introduce Accessibility Issues
- Simplify Development Environment and Introduce New Facilities

These actions were put into place in the experimentation phase of the project. Each activity is described below, and whether it was carried out successfully.

5.8.1 Determine a Formal Development Process

A formal development process was defined at the start of the project. This was put into place for Project One. It was relatively successful, although there were problems within the individual stages. For example, the requirements gathering stage created a large amount of requirements creep, which increased the amount of work without making any provision for increased time or resources.

A review of the process was carried out, and improvements proposed. These were integrated into the next project and the process carried out again. This was repeated for Project Three. In order to gauge the effect this has had on CACDP development processes a CMM survey was carried out again, as in Chapter 3. The results can be compared to determine the improvements made to the process. A graph of the results of both surveys is shown in Figure 39. This is a huge improvement for the organisation.

The results of the CMM survey and the working procedures in place at CACDP show that this activity has been carried out successfully.

5.8.2 Simplify Maintenance Procedures

As part of the aim to simplify the maintenance procedures, templates were introduced in all packages. This allows CACDP to alter the interface to each package very easily, without the need for time-consuming editing. Reuse of other artifacts was also put into place in the development. This included code sections, database access routines, requirements, frameworks, design and testing procedures.

Forms for the databases were designed and implemented in order to simplify maintenance of the databases.

5.8.3 Reduce Development Time

Development time was significantly reduced both by the introduction of formal processes, and by the reuse of assets throughout the different sub-projects. The development time was reduced by the greatest proportion for the two packages that were most similar – Projects One and Three. Project Two benefited from work done in Project One but not to such an extent as Project Three. This was due to the nature of the domains in which each package existed. Projects One and Three were both in the *Online Learning* domain and as such have more in common than with Project Two, which was in the *Online Directories* domain.

It has been shown that there is scope for reuse in each of the following ways:

- Within individual packages – for example page templates, and sections of code that can be used for similar functions
- Within each domain – for example, course frameworks, requirements
- Between domains – for example, security systems, database access routines, design concepts

Figure 40 shows the three packages divided into three sections – the Security System, the Interface and the Database. The lines indicate where reuse has taken place between them. A broken line shows where partial reuse has taken place, i.e. where the object was amended in some way, or where only part of the object was reused, for example, the Interface for Project One was reused in Project Two in terms of the use of templates, but the design was changed to suit the different systems. An unbroken line shows where an object has been directly reused completely; for example, the Login System developed in Project One was amended for Project Two, but then reused "as-is" in Project Three.

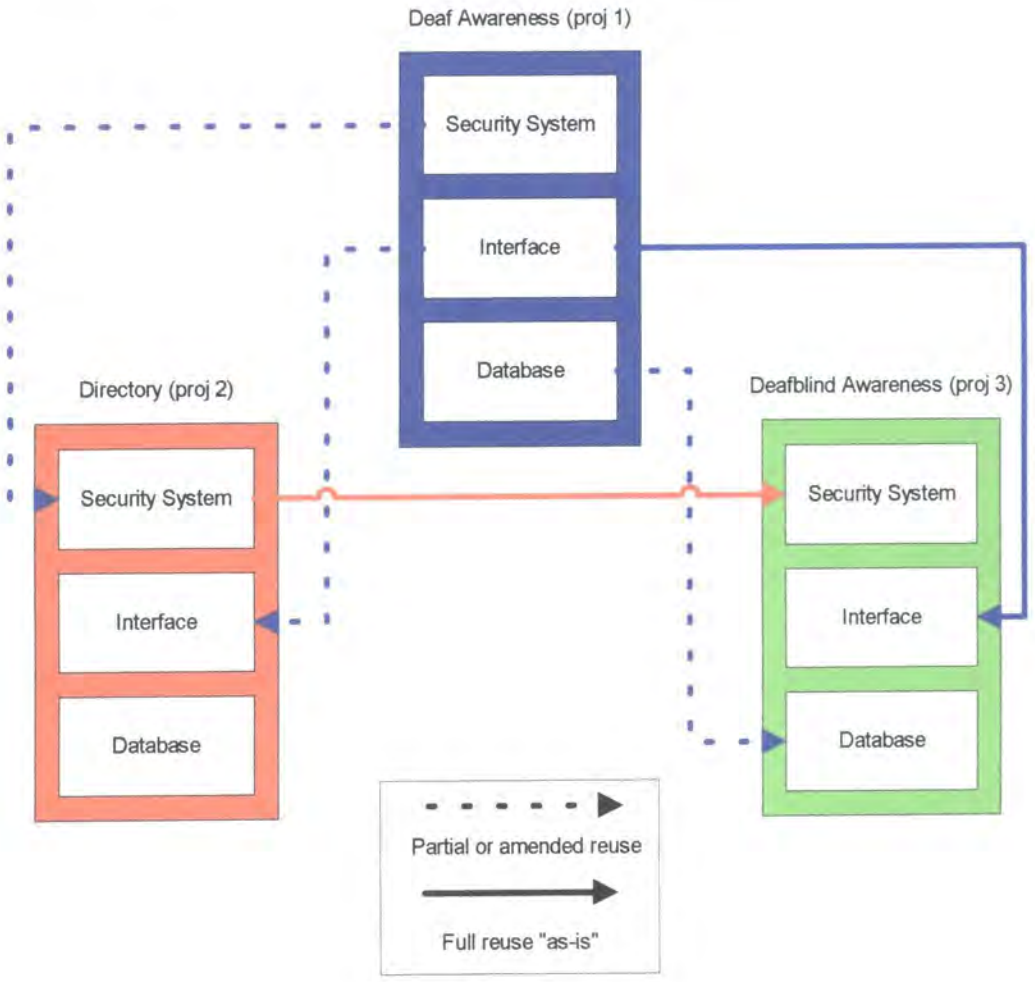


Figure 40 – Reuse Between Systems

Each time an item is reused, time is saved on the development of that object. Time is also saved on the testing process as items are assumed to be working correctly, unless they have been modified, in which case they must be retested.

5.8.4 Introduce Accessibility Issues

As accessibility is a major concern of the organisation, this was an important task. Accessibility issues were taken into account during the design phases to ensure that the interface met the guidelines set out by the W3C for web accessibility [71]. The W3C is the World Wide Web Consortium. They set standards and guidelines for accessibility and technology. CACDP have been introduced to the standards, and the challenge of meeting them with the software they produce. Guidelines were set out in section 4.5.3 and it was ensured that the systems designed met these guidelines. The CACDP current web site was also checked to ensure that it too met the guidelines.

5.8.5 Simplify Development Environment and Introduce New Facilities

The current software used by CACDP to develop their web site was not suitable for more complex systems so Macromedia Dreamweaver was used instead. This is a more complex program and is very suitable for dynamic code using scripts and database access. It is compatible with Microsoft Access 2000, which is used by the organisation already.

New hosting and email facilities were found for CACDP to host their new systems on. These were provided by *The Web Works*, which is a small company, located just outside Durham, who can offer a more personalised service than some of the larger companies. This should help CACDP to be able to maintain the system in the future.

5.9 Summary of Chapter

This chapter contained the results and the evaluation of the results from the experimentation phase. It covered the results in terms of Reuse, Process Improvement, and the Action List from Chapter Three. The results for each aspect were described, and then evaluated based on the theory described in Chapter Two

Chapter 6: Conclusions and Further Work

6.1 Conclusions

The conclusions for this project are in the form of improvements and benefits gained by the company, and any issues that were raised during the project. This chapter also refers back to the Criteria for Success (SC1-SC8) defined in section 1.5 in order to determine if and how these criteria were achieved. A summary of these criteria and how they are fulfilled is given at the end of this section in Table 3.

CACDP began this project with a very low level of knowledge about both formal processes and reuse. They were working at the Initial Level of Software Process Maturity where they had no formal processes and their software development process was unpredictable [5, 35]. The only software they produced was their own web site, but they wished to expand their online presence by producing other web-based services for their customers. They saw this as a way to improve their services to their customers, a way to increase their income and a way to promote their examinations. CACDP also wished to increase the accessibility of their services, as they have many links with disabled people. This is important to them and needed to be incorporated into the process. The packages they wished to develop were:

- Deaf Awareness Learning Package
- Deafblind Awareness Learning Package
- Deaf Community and Culture Learning Package
- Directory of Human Aids to Communication

In order to successfully produce these products, improvements had to be made to several factors. CACDP needed to change the development software they were using, as it would not be suitable for the complexity of the new systems. They needed to find a new web hosting company because their current host did not offer the facilities required for interactive services. CACDP also needed to implement a more formal software development process in order to ensure that standards were kept up and to make it more efficient to produce new software. Members of staff at CACDP do not have the skills for these tasks, so training would also be required.

A company analysis was carried out, and a formal assessment of their current development practices carried out, thus fulfilling SC2. This was done using a CMM survey, which indicated where their deficiencies lay and what needed improving. As a result of the company analysis, an action list was drawn up indicating what needed to be done to achieve the results required. The action list contained the following tasks:

- Determine a Formal Development Process
- Simplify Maintenance Procedures
- Reduce Development Time
- Introduce Accessibility Issues
- Simplify Development Environment and Introduce New Facilities

A number of potential development processes were researched, and their suitability for use within CACDP was noted. These development processes are described in Chapter 2 as required by SC1.

In order to introduce a formal development process, it was necessary first to determine a suitable process for the company's level of understanding and then to introduce it gradually in order to counteract the natural resistance to change of the organisation. It was decided that the Waterfall process would be simple enough for the company to understand and yet formal enough that it would give enough structure to the process, as set out in SC3. It was decided to introduce the process in an iterative manner, performing process improvement activities during each cycle in order to show CACDP how they could continue to improve their own processes after the project.

Each package to be developed would form a single iteration of the Waterfall process, allowing improvements to be identified during a review of the process, and the improvements put into place during the next iteration.

The concept of *reuse* was also introduced into the process, enabling CACDP to gain the benefits of more efficient production, quicker development and a higher quality product. CACDP were currently working at below the lowest level of reuse identified. The lowest level is "ad hoc" [13], where engineers are reusing their own work between projects in an informal manner. CACDP were not reusing any development work at all.

The process was put into place and the first package, the Deaf Awareness Learning Pack, produced (SC4). There were problems encountered during the requirements stage, leading to an increase in development time required, and inconsistency in the

requirements. These inconsistencies filtered through to the design and then the implementation stages, causing problems at these levels. This identifies a need to spend time ensuring that the requirements are consistent, unambiguous and clear. Sommerville [5] identifies three types of problems with requirements written in plain language:

- Lack of clarity – requirements should be precise and unambiguous
- Requirements confusion – functional and non-functional requirements should be distinguished from system goals and design information
- Requirements amalgamation – requirements should be expressed individually

These problems could be identified in the requirements documents for the first system. It was difficult to resolve these problems. The first set of requirements documents were written to adhere to the IEEE standards for requirements documents, yet CACDP could not understand them and requested that they be written in plain language. Therefore it was very difficult to achieve the criteria.

A further problem was that prototyping was clearly required and was introduced ad-hoc into the process during the first project. It was successful and was identified as a change to be made to the process the second time around.

The first package was successfully implemented, after a few problems due to the handover from an external developer to the author who completed the project, and the process reviewed. The changes made to the process were modifications to the requirements gathering stage in order to improve the quality of the requirements definition, and the introduction of a prototyping stage. Process Improvement techniques were applied, satisfying SC5.

The second package to be produced was the Directory of Human Aids to Communication. As this was not in the same domain, i.e. online learning, as the Deaf Awareness Package, a further domain survey had to be carried out. The requirements for the Directory were defined using the new recommendations for this stage. The process was more successful this time around, leading to less confusion and inconsistency, although it was still not perfect. The requirements document was more acceptable to CACDP as it was written in a way they could understand. Decision took less time to achieve, as the questions asked of the company were not as open-ended as previously.

As a result of the requirements being clearer, the design was easier to develop. Some of the design work from the Deaf Awareness Package could be reused in this project; for example, the database access routines were the same (SC6). More time was spent on the design in Project Two, ensuring that it was more complete than in Project One.

The Implementation stage of Project Two went more smoothly than in Project One. Partly this was due to having one developer rather than two, and partly it was due to the ability to reuse work from Project One. Much of the code needed to be modified before it was useful, but the experience from the development of Project One was invaluable. The security system from Project One was modified to make it suitable for an unlimited number of systems, thus anticipating its reuse in Project Three. This was development *for* reuse, rather than development *with* reuse, which was already occurring.

Project Two was tested in a similar way to Project One. As it is subscription based, it was decided to allow one month of free access so that the public could report any problems. A feedback system was implemented and the system publicised. Much of the feedback was positive and only one problem was reported. A user who had an apostrophe in their name, e.g. "O'Connor" was having trouble accessing their own records. This was rectified easily and no more problems were reported.

The third package to be implemented was the Deafblind Awareness Learning Package. This was able to reuse much of the material from earlier projects, mainly Project One due to the similarities in requirements. There were slight modifications to the requirements, but these were only minor, for example, there should be an unlimited amount of questions in a test, rather than 3, as specified in Project One. These changes were made, and the design altered accordingly. Again, much of the work could be reused directly from Project One, and the Security system that was modified in Project Two was reused without further modification in Project Three. This saved much time and effort and ensured that the learning packs were consistent in their development. Project Three was field-tested, and fewer problems returned than Project One. This was because many of the issues had already been identified, thus showing a practical example of one of the benefits of reuse.

During these projects, practical examples of the benefits of formal processes and reuse have been shown. The introduction of a formal process to CACDP will allow them to continue to produce further web-based products (SC7). They have increased the

scope of the company, and enhanced the services they offer. CACDP have made improvements in the following Key Process Areas:

- Requirements Management
- Project Planning
- Process and Product Quality Assurance

This is a marked improvement, and they can continue to work towards the fulfillment of all the Level 2 criteria. In terms of reuse, they were not operating any form of reuse programme before, but now they are aiming towards being able to operate at the level of Domain Orientated reuse [13] where they are focussing on creating a repository of assets that they can reuse within their own business domain (SC6).

Other benefits to the company of the project include increased professionalism as they now have their own domain name, www.cacdp.org.uk, and email addresses in the same format. They have a greater range of services to offer customers, and a wider range of customers to sell to as they have branched out worldwide. The Online Directory was a product that had been requested by several customers over several years, so CACDP could show that they were listening to their customers' desires, and responding.

In terms of accessibility, CACDP have achieved Level A of the Priority One guidelines set out by the WAI (SC8). This will gain them approval within their own sphere of influence and will prepare them for the imminent government rulings on accessibility of information.

With reference to the Criteria for Success set out in section 1.5, all the criteria have been met with some degree of success. SC7 and SC8 refer to the enhancement of knowledge within the company; obviously this is a matter that cannot be measured as such at this time and will be measured by how CACDP continue to work and produce products. SC1, to identify state of the art procedures and investigate the feasibility of adoption by CACDP, was carried out during the background research stage. Several processes were looked at and the Waterfall Model was chosen due to it's simplicity and structured approach. SC2 referred to a study of the existing state of CACDP's processes. This was incorporated into an overall Company Analysis. SC3 was to define a development process. This was completed as the basis for the project. The development of the products in the four mini projects met SC4, and between each project, improvement techniques were applied, thus meeting SC5. SC6 referred to the

introduction of reuse techniques into the company. This was achieved generally over the four projects as CACDP gained knowledge about reuse and how to apply it to their own development activities. Thus, it can be stated that all the Success Criteria have been met.

Table 3 - Summary of Research Success Criteria and Fulfillment

Criteria	Fulfillment	
SC1 – Identify state of the art procedures and investigate the feasibility of adoption for small companies	Detailed literature survey covering • Software Reuse • Software Process Improvement • Web Development For further details see Chapter 2.	✓
SC2 – Carry out a study of the software development processes and procedures of a small company and document the current activities with a view to improving the procedures	Formal assessment of current development practices using CMM survey For more details see Chapter 3.	✓
SC3 – Define a development process for web based products	Development process refined using Process Improvement techniques, the Waterfall Model and prototyping approaches. For more details see Chapter 4.	✓
SC4 – Develop commercial products using the process defined above	Products developed include • Deaf Awareness Learning Package • Directory of Human Aids to Communication For more details see Chapter 4.	✓
SC5 – Improve the defined process by the application of Process Improvement techniques	Changes to the development process were made at the start of each mini-project. This allowed continued Process Improvement throughout the development of the software packages. For more details see Chapter 4.	✓
SC6 – Introduce the concept of reuse to the company in order to make their procedures more efficient	Reuse achieved within the project included • Database access routines • Security System • Testing Procedures	✓

	• Development Processes For more details see Chapter 4.	
SC7 – Enhance the knowledge of the organisation with regard to web development, development procedures with reuse, and process improvement	New formal procedures have been designed and 'taught' to the company. CACDP have improved in 3 of the CMM key process areas. For more details see Chapter 5.	✓
SC8 – Introduce guidelines for web accessibility to ensure that all products, new and existing, are suitable for use by all potential users	Level A priority Guideline as defined by the WAI were introduced in all developed products For more details see Chapter 5.	✓

In summary, the project was a success in terms of the products produced, and the changes to the organisational structure. CACDP were very satisfied with their new systems and are intending to continue to produce further web-based products in the future.

6.2 Further Work

In the future, CACDP could extend their success by introducing a more web-specific process and making use of the more complex demands it would make. They could also strive to reach a higher level of process maturity by putting measurements and metrics into place and putting a continuous improvement schedule into place, thus reaching Level 2 for process maturity.

Equally, they could also seek to improve their reuse programme by extending it to their other documentation and components so that they can reuse these assets more easily. CACDP is a small company however, and their main focus is the advancement of communication with the deaf, so they may not wish to expend time and energy attempting to reach the higher levels of reuse and process maturity. This would involve a large financial and organisational commitment and would not necessarily benefit the company a great deal.

As the area is of particular importance to CACDP, accessibility issues were addressed throughout the project. Accessibility issues are not addressed directly in any of the current development processes, yet they are becoming vital to the success of any large company web application. In the USA and Australia, along with many other countries

[75], government legislation is being brought in to ensure that government departments and large companies comply with the standards set out. Some countries simply ascertain a "right to certain kinds of information" while others have laws stating that products sold, or information provided, meets accessibility standards. CACDP can work towards attaining the accessibility levels that are required of an organisation in their area.

6.3 Project Successes

The project should be considered a success. CACDP have gained the following benefits:

- A mature process that they understand and can use to produce more web based products in the future
- Three new online services for their customers
- Increased profitability
- Web products that conform to Level A of the WAI Web Content Accessibility Guidelines, that can be used by all their customers
- Maintenance routines that can be carried out more easily and more quickly, and should cause less hassle when maintenance is passed from one person to another
- Increased technical learning within the organisation
- A more professional approach to their image in terms of online services

This work has demonstrated the feasibility of introducing a formal process to a small company, which incorporates reuse and accessibility considerations by means of process improvement.

References

[1] J. Gillies and R. Cailliau, *How the web was born*. New York: Oxford University Press Inc, 2000.

[2] D. B. Lowe and W. Hall, *Hypermedia and the Web: An Engineering Approach*. Chichester, UK: John Wiley and Sons, Ltd, 1999.

[3] S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige, "Web Engineering: A New Discipline for Development of Web-based Systems," presented at First ICSE Workshop on Web Engineering, 1999.

[4] S. P. Christodoulou, P. A. Zafiris, and T. S. Papatheodorou, "Web Engineering: The Developers' View and a Practitioner's Approach," *Lecture Notes in Computer Science*, pp. 170-187, 2001.

[5] I. Sommerville, *Software Engineering*. Harlow, Essex: Addison Wesley Longman Limited, 1995.

[6] S. Reiblein and A. Symons, "SPI: 'I can't get no satisfaction' - directing process improvement to meet business needs," *Software Quality Journal*, vol. 6, pp. 89-98, 1997.

[7] K. E. Emam and L. Briand, "Costs and Benefits of Software Process Improvement," Fraunhofer IESE, Kaiserlautern, Technical Report ISERN-97-12, 31 December 1997 1997.

[8] M. Ezran, M. Morisio, and C. Tully, *Practical Software Reuse: the essential guide*: SURPRISE Project, 1999.

[9] C. W. Krueger, "Software Reuse," *ACM Computing Surveys*, vol. 24, 1992.

[10] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organisation for Business Success*. New York: ACM Press, 1997.

[11] E.-A. Karlsson, *Software Reuse: A Holostic Approach*. Chichester, UK: John Wiley & Sons Ltd, 1995.

[12] F. P. Brooks, *The Mythical Man-Month*, 2nd ed: Addison Wesley Longman, Inc, 1995.

[13] W. C. Lim, *Managing Software Reuse*. London: Prentice-Hall Inc, 1998.

[14] M. Mannion, B. Keepence, H. Kaindl, and J. Wheadon, "Reusing Single System Requirements from Application Family Requirements," presented at 21st International Conference on Software Engineering, 1999.

[15] J. L. Cybulski and K. Reed, "Requirements Classification and Reuse: Crossing Domain Boundaries," presented at Software Reuse: Advances in Software Reusability, Vienna, Austria, 2000.

[16] A. Lynex and P. J. Layzell, "Organisational considerations for software reuse," *Annals of Software Engineering*, vol. 5, pp. 105 - 124, 1998.

[17] D. Schwabe, G. Rossi, L. Esmeraldo, and F. Lyardet, "Web Design Frameworks: An Approach to Improve Reuse in Web Applications," *Lecture Notes in Computer Science*, pp. 335-352, 2001.

[18] T. Smart, *The Oxford Modern English Dictionary*. Godalming, UK: Oxford University Press, 1995.

[19] H. Gomaa, "Reusable Software Requirements and Architectures for Families of Systems," *Journal of Systems and Software*, vol. 28, pp. 189-202, 1995.

[20] A. M. d. Cima, C. M. L. Werner, and A. A. C. Cerqueira, "The Design of Object-Oriented Software with Domain Architecture Reuse," presented at 3rd International Conference on Software Reuse, 1994.

[21] W. Schafer, R. Prieto-diaz, and M. Matsumoto, *Software Reusability*. Hemel Hamstead, UK: Ellis Horwood Limited, 1994.

[22] D. Kaspersen, "For Reuse, Process and Product both Count," *IEEE Software*, vol. 11(5), 1994.

[23] R. M. Carro, E. Pulido, and P. Rodriguez, "Improving Web-Site Maintenance with TANGOW by Making Page Structure and Contents Independent," *Lecture Notes in Computer Science*, vol. 2016, pp. 325-334, 2001.

[24] J. Segal, "Organisational Learning and Software Process Improvement: A Case Study," *Lecture Notes in Computer Science*, vol. 1881, pp. 70-80, 2001.

[25] G. Tong, "Software Development Process Improvement: The Forgotten Son?," *World Class Design to Manufacture*, vol. 1, pp. 21-25, 1994.

[26] K. E. Emam, J.-N. Drouin, and W. Melo, *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Los Alamitos, California: IEEE Computer Society, 1998.

[27] D. B. Lowe and R. Webby, "Hypermedia Process Modelling," University of Sydney and University of New South Wales, Internal Report 1998 1998.

[28] A. Ginige and S. Murugesan, "The Essence of Web Engineering," *IEEE Multimedia*, vol. Apr-Jun 2001, pp. 22 - 25, 2001.

[29] M. L. Russ and J. D. McGregor, "A Software Development Process for Small Projects," *IEEE Software*, vol. September/October 2000, pp. 96 - 101, 2000.

[30] R. V. Horvat, I. Rozman, and J. Gyorkos, "Managing the Complexity of SPI in Small Companies," *Software Process: Improvement and Practice*, vol. 5, pp. 45-54, 2000.

[31] A. Sweeny and D. W. Bustard, "Software process improvement: making it happen in practice," *Software Quality Journal*, vol. 6, pp. 265-273, 1997.

[32] T. C. Green and K. M. Anderson, "Configuration Management Culture as the Kernel to Success in Software Process Improvement Efforts," *Lecture Notes in Computer Science*, vol. 2077, pp. 236-241, 2001.

[33] J. Batista and A. Dias de Figueirdo, "SPI in a Very Small Team: a Case with CMM," *Software Process Improvement and Practice*, vol. 5, pp. 243-250, 2000.

[34] S. E. Institute, "List of High Maturity Organizations," vol. 2002: Software Engineering Institute, 2002.

[35] M. C. Paulk, J. Perdue, M. B. Chrissis, and C. V. Weber, "The Capability Maturity Model for Software, Version 2B," Software Engineering Institute, Technical Report September 1997 1997.

[36] M. C. Paulk, "A Comparison of ISO 9001 and the Capability Maturity Model for Software," Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report July 1994 1994.

[37] H. K. N. Leung and T. C. F. Yuen, "A Process Framework for Small Projects," *Software Process - Improvement and Practice*, vol. 6, pp. 67-83, 2001.

[38] D. R. Goldenson and J. D. Herbsleb, "After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success," Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report August 1995 1995.

[39] C. Debou and A. Kuntzmann-Combelles, "Linking Software Process Improvement to Business Strategies: Experiences from Industry," *Software Process - Improvement and Practice*, vol. 5, pp. 55-64, 2000.

[40] IEEE, "IEEE Standards on Software Engineering," 1991.

[41] Y. Deshpande and S. Hansen, "Web Engineering: Creating a Discipline among Disciples," *IEEE Multimedia*, vol. Apr-Jun 2001, 2001.

[42] A. Ginige and S. Murugesan, "Web Engineering: An Introduction," *IEEE Multimedia*, vol. Apr-Jun 2001, pp. 14 - 18, 2001.

[43] K. S. Norton, "Applying Cross-Functional Evolutionary Methodologies to Web Development," *Lecture Notes in Computer Science*, vol. 2016, pp. 48-57, 2001.

[44] C. Boldyreff, E. Burd, and J. Lavery, "Towards the Engineering of Commercial Web-Based Applications," presented at SSGRR 2001, L'Aquila, Italy, 2001.

[45] L. Schmeiser, "Web Site Evolution: Designing and Developing for the Future," presented at 1st International Conference on Web Site Evolution, Atlanta, GA, USA, 1999.

[46] D. B. Lowe and B. Henderson-Sellers, "Web Development: Addressing Process Differences," *Cutter IT Journal*, vol. July 2001, 2001.

[47] D. B. Lowe and B. Henderson-Sellers, "Characteristics of Web Development Processes," presented at SSGRR 2001, L'Aquila, Italy, 2001.

[48] D. B. Lowe, "A Framework for Defining Acceptance Criteria for Web Development Projects," presented at 2nd ICSE Workshop on Web Engineering, Limerick, Ireland, 2000.

[49] A. Ginige, "Web Engineering in Action," *Lecture Notes in Computer Science*, vol. 2016, pp. 24-32, 2000.

[50] P. Kyaw and C. Boldyreff, "A Survey of Hypermedia Design Methods in the Context of World Wide Web Design," University of Durham, Durham, Computer Science Technical Report March 1998 1998.

[51] F. Coda, C. Ghezzi, G. Vigna, and F. Garzotto, "Towards a Software Engineering Approach to Web Site Development," presented at 9th International Workshop on Software Specification and Design, 20th International Conference on Software Engineering, Kyoto, Japan, 1998.

[52] F. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model," *Communications of the ACM*, vol. 37, pp. 30 - 39, 1994.

[53] K. Gronbaek, J. A. Hem, O. L. Madsen, and L. Sloth, "Hypermedia Systems: A Dexter-Based Architecture," *Communications of the ACM*, vol. 37, pp. 65 - 74, 1994.

[54] K. Gronbaek and R. H. Trigg, "Hypermedia Design Issues for a Dexter-Based Hypermedia System," *Communications of the ACM*, vol. 37, pp. 40 - 49, 1994.

[55] F. Garzotto, P. Paolini, and D. Schwabe, "HDM - A Model-Based Approach to Hypertext Application Design," *ACM Transactions on Information Systems*, vol. 11, pp. 1-26, 1993.

[56] L. Hardman, D. G. A. Butterman, and G. van Rossum, "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model," *Communications of the ACM*, vol. 37, pp. 50 - 62, 1994.

[57] T. Isakowitz, E. A. Stohr, and P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design," *Communications of the ACM*, vol. 38, pp. 34 - 44, 1995.

[58] L. Schmeiser, *The Complete Website Upgrade and Maintenance Guide*: SYBEX, 1999.

[59] S. Ceri, P. Fraternali, and A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites," presented at WWW9, Amsterdam, 2000.

[60] D. E. Avison and G. Fitzgerald, *Information Systems Development: Methodologies, Techniques and Tools*, 2nd ed. Maidenhead, Berkshire: McGraw-Hill, 1995.

[61] B. W. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software*, vol. 8, pp. 32-41, 1991.

[62] H. Ledgard and J. Tauer, *Software Engineering Concepts*, vol. 1: Addison-Wesley Publishing Company, 1987.

[63] R. S. Pressman, "What a Tangled Web We Weave," *IEEE Software*, vol. Jan/Feb 2000, pp. 18 - 21, 2000.

[64] P. J. Warren, C. Boldyreff, and M. Munro, "The Evolution of Websites," presented at International Workshop on Program Comprehension 1999, Pittsburgh, USA, 1999.

[65] M. C. Paulk, "Using the Software CMM in Small Organizations," presented at The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality, Portland, Oregan, 1998.

[66] P. Brereton, D. Budgen, and G. Hamilton, "Hypertext: The Next Maintenance Mountain," *IEEE Computer*, pp. 49 - 55, 1998.

[67] J. Verner and H. A. Muller, "Management of Web Site Evolution," presented at 1st International Conference on Web Site Evolution, Atlanta, GA, 1999.

[68] S. Murugesan, "Web Engineering," *SigWeb*, pp. 28 - 32, 2001.

[69] M. N. Louka, "A Review of Hypermedia Methodologies and Techniques," University of Surrey, UK, Guildford March 1994 1994.

[70] C. Boldyreff, W. Paul, C. Gaskell, and A. Marshall, "Web-SEM Project: Establishing Effective Web Site Evalutation Metrics," presented at Web Site Evolution, 2000.

[71] W. Chisholm, G. Vanderheim, and I. Jacobs, "Web Accessibility Content Guidelines 1.0," vol. 2002, 1.0 ed: Web Accessibility Initiative, 1999.

[72] J. Donkin, C. Boldyreff, E. Burd, and S. Marshall, "The Case for the Use of Plain English to Increase Web Accessibility," presented at Web Site Evolution, Firenze, Italy, 2001.

[73] J. Donkin, C. Boldyreff, E. Burd, and S. Marshall, "Supporting Sign Language Users of Web-Based Applications: A Feasibility Study," presented at Universal Access in Human-Computer Interaction, New Orleans, USA, 2001.

[74] E. Burd, "Lecture Notes for Advanced Software Engineering," vol. 2002, 2002.

[75] J. Brewer and A. Chuter, "Policies Relating to Web Accessibility," vol. 2002: W3C, 2002.

Appendix A – Results of CMM Questionnaire

These results are in the form of the answers to the questions. Y=yes, N=no, D=Does not apply. X indicates where an improvement has been made, and O indicates an anomaly in the answers.

Requirements Management		
1	N→N	
2	Y→Y	
3	N→Y	X
4	N→N	
5	N→N	
6	N→Y	X
Software Project Planning		
1	N→Y	X
2	N→Y	X
3	Y→Y	
4	N→Y	X
5	Y→N	O
6	N→Y	X
7	N→Y	X
Software Project Tracking and Oversight		
1	N→N	
2	N→N	
3	N→N	
4	N→Y	X
5	N→Y	X
6	N→N	
7	N→Y	X
Software Subcontract Management		
1	D→D	
2	D→D	

3	D→D	
4	D→D	
5	D→D	
6	D→D	
7	D→D	
8	D→D	
Software Quality Assurance		
1	Y→Y	
2	N→N	
3	N→N	
4	N→N	
5	N→Y	X
6	N→Y	X
7	N→N	
8	N→Y	X
Software Configuration Management		
1	N→Y	X
2	N→Y	X
3	N→Y	X
4	N→N	
5	N→Y	X
6	N→N	
7	N→N	
8	N→N	
Organisation Process Focus		
1	N→Y	X
2	N→Y	X
3	N→Y	X
4	N→Y	X
5	N→Y	X
6	N→N	
7	N→N	
Organisation Process Definition		
1	N→Y	X
2	N→N	

3	N→Y	X
4	N→N	X
5	N→Y	X
6	N→Y	X
Training Program		
1	N→Y	X
2	N→Y	X
3	Y→Y	
4	Y→Y	
5	N→Y	X
6	N→N	
7	N→Y	X
Integrated Software Management		
1	N→Y	X
2	N→Y	X
3	N→Y	X
4	N→N	
5	N→N	
6	N→Y	X
Software Product Engineering		
1	D→Y	X
2	D→N	
3	D→Y	X
4	D→Y	X
5	D→N	
6	D→Y	X
Inter-Group Co-ordination		
1	Y→Y	
2	Y→Y	
3	N→Y	X
4	N→N	
5	N→Y	X
6	N→Y	X
7	N→Y	X
Peer Reviews		
1	D→D	

2	D→D	
3	D→D	
4	D→D	
5	D→D	
6	D→D	

Appendix B – Domain Survey Results Report

Introduction

The Council for the Advancement of Communication with Deaf People (CACDP) is the awarding body for qualifications in Deaf Awareness and British Sign Language (BSL) in the UK. They are currently undertaking research into the possibility of migrating their existing Deaf Awareness and BSL curricula onto a web-based system. It is intended that this will increase the availability of the qualifications. Within the UK as a whole there is often desire or necessity to communicate with deaf people, but there is a lack of experienced, trained teachers to be able to train the hearing community to appropriate skilled levels where easy communication is possible. Furthermore, most of this training must be provided within people's leisure time, so the training must be flexible and fun. This project, through the use of web-based materials, will seek to reduce this training need gap.

The Study

This study has been carried out as part of the requirements determination process. The aim is to study existing similar software in order to gain an insight into how such systems are designed and the good and bad points about each. Hopefully this will enable the requirements for the new system to be more comprehensive. The criteria, on which the systems will be evaluated, has been devised so that it will cover the main components which must be considered when designing systems. The evaluation was carried out after about an hour's usage of each system by one person.

Criteria

The systems will be evaluated in terms of specific criteria in order to allow comparison between systems. These criteria have been adapted from HCI textbooks (Preece, Dix and Finlay). The evaluation of the system must cover the three major components of the design, that is, the User Interface, the Teaching Method and the Technology Used. Therefore, the criteria are as follows:

- System Usability Characteristics – this includes the presentation of the information on the computer screen, its layout and the general 'first impressions' felt by users.

- Language Specifics – this evaluation criterion investigates if the software is composed for a specific deaf communication language and what, if any, implications this support will have for other deaf communication.
- Pedagogical Nature of the Application – including the training approaches used and whether this appropriately supports the learning process.
- Navigation Capabilities – this criterion evaluates if learners are able to appropriately navigate around the software, whether they become lost and disorientated and the overall effect this has on the training.
- Use of Technology – this aspect evaluates the types of technologies that are used and how effective each technology is in terms of supporting learning.
- Real Time Capabilities – what speeds are demanded by the software and whether technology is able to deliver the required expectations and overall how this effects the learning process.
- Entertainment – since many of the learners will be using these courses as non-vocational the entertainment factor is also considered an important aspect in the process of maintaining the motivation of the learner.
- Accuracy – the information provided by the system must be accurate and correct.

NB: This study does not cover the Accuracy criteria at this point due to the lack of availability of experts, and the lack of expertise in BSL, ASL and Deaf Awareness by those carrying out the study.

Systems Evaluated

The systems to be evaluated are all described as applications for teaching either Deaf Awareness, British Sign Language or American Sign Language. They are all available commercially. These systems were selected due to availability, and the similarity of their objectives to the CACDP project.

- Deaf Awareness Online CD-ROM
- Sign Language For Everyone CD-ROM
- Simple Signs CD-ROM
- Personal Communicator CD-ROM

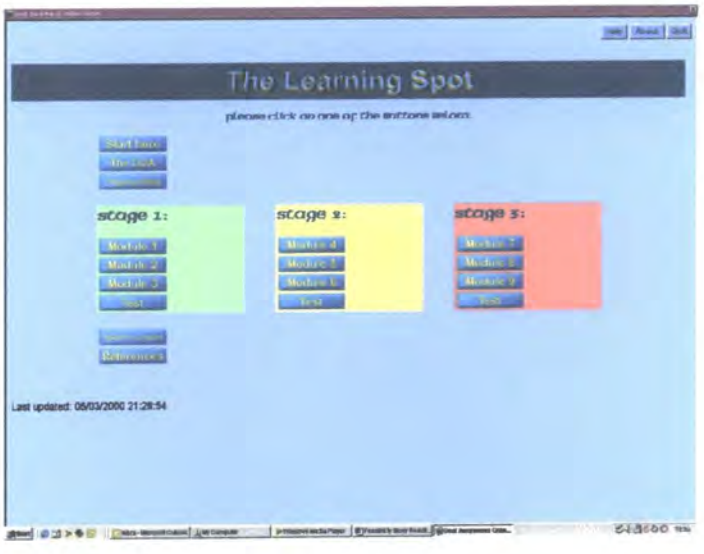
Two of the systems use ASL and two use BSL. Since the systems are being evaluated on their user interface, their teaching methods and the technology used, it doesn't matter which language the systems are teaching. Therefore it is appropriate to include both British and American software.

Feasibility Study Results

System 1: Deaf Awareness Online (DAO)

System Usability Characteristics

This system is quite simple to use as it uses the mouse to navigate round the package. The system doesn't give the impression of being very high-tech as it uses certain techniques, such as a spinning title, which look quite juvenile and low-tech. The background is bright blue which could cause problems for partially sighted users as it may not offer enough contrast with the darker blue buttons. The Irish font used for some of the text is hard to read. The program opens as a full screen.



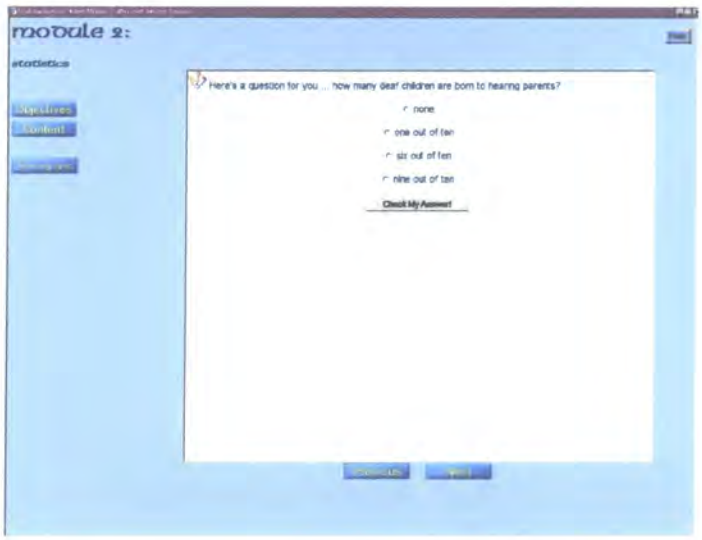
Language Specifics

This system is only designed to teach Deaf Awareness and basic fingerspelling. There are no facilities for learning sign language.

Pedagogical Nature of Application

The system simply gives the user the information in the form of text, and then there is a short multiple choice test at the end of every third module. The test uses radio buttons for the user to select an answer and then tells them if they chose the correct one or not. If not, the user is prompted to chose a different answer. The system does not offer any indication as to which answer is correct, and the user must continue until they select the right one. There is no interactivity within the system. There are a few points where the system offers the user more information about something if they click on a link, but these are limited.

The system makes use of simple questions within the modules to make it more interactive, as shown below.



This makes it a little more interesting than simply text cards.

Navigation Capabilities

The user is presented with a menu listing Module 1, Module 2 etc with no indication of what topics are covered in which modules. The system cannot indicate to the user which modules the user has already completed and which are still to be done. The user must select a module and read each text entry before moving on to the next.

The system has good, clear 'next' and 'previous' buttons. However, when the user reaches the end of a module, there is no indication that they have reached the last page. If the user tries to continue by clicking on the 'next' button, they get an error message which doesn't look professional, and could be prevented. The error message does not inform the user how to return to the menu so they still do not know how to continue.

The system also uses tests at the end of every third module so the user can check how they are getting on. These are made up from about 10 multiple choice questions and the user must answer them and then they get a total score

Use of Technology

The system does not make use of any technology which requires high specification hardware. The minimum system requirements are stated as "Windows 95/98, Internet Explorer 5, CD-ROM Drive, VGA Monitor, Mouse". The System runs straight from the CD with no installation required which reduces confusion and problems. It loads automatically when the CD is loaded which is a good idea for novice users. As the application contains only text and a few still photographs, it runs quickly from the CD without long delays.

The system makes use of the mouse to navigate and work through the modules. The system offers an introduction to using the mouse as a help section. This is a great idea as some users may be unconfident when using the mouse, but it may be considered confusing as it goes through a lot of activities which are not used within Deaf Awareness Online.

Real Time Capabilities

The CD runs quite quickly and the speed is appropriate because the text takes time to read and digest anyway.

Entertainment

One problem with this package is that, as it is mainly text based, it can become tedious after a short time. The tests at the end of some of the modules are quite motivating as they offer the chance to gain a total score rather than just individual questions, but the test does not offer the user any feedback other than "Correct" or "Incorrect" and does not offer the chance to go back and restudy the information which they got incorrect. Another confusing point is that, when the user gets a question incorrect, they are offered the same question again and must continue until they select the correct answer.

System 2: Sign Language for Everyone (SLFE)

System Usability Characteristics

SLFE is a very high tech looking system, which loads in a flurry of classical music and video clips. It makes use of audio and video to grab the users attention. It loads in a small window which users may want to be able to resize (they cannot) and this may make the text very small. It also means the background can be viewed behind the window which can be distracting for the viewer.

Language Specifics

This system is aimed at beginners who wish to learn ASL. It starts with a basic introduction to the history and grammar of sign language and then moves on to simple signs, before introducing more complex phrases and concepts.

Pedagogical Nature of Application

SLFE is based around a lesson approach where the user proceeds from simple to more advanced topics. This allows them to build on their previous experience and information to learn the more difficult concepts. The system makes use of text, audio and video to pass on the information to the learner.

Navigation Capabilities

The navigation is well organised but badly implemented. The lessons are arranged between Introductory Lessons and Main Lessons, and then into topics. This lets the user know what topics are contained within each lesson, and also indicates where they should start. However, the buttons used for navigation are not well designed. They are unclear, especially for those with impaired vision, and they offer no textual alternative. There is no indication of which icons will do which actions as the icons themselves offer no connection to the tasks. The only way to discover what they do is either trial and error, or reading the introductory lesson each time. Neither is acceptable.

Use of Technology

This system makes very good use of high quality video and audio capabilities. The video is large enough to see the hands of the signer, and the system also shows what shape the individual hands should be at the bottom of the video window. The audio simply reads the text printed at the side, but it allows the learner to focus on the signing window and listening without needing to read the text as well. The audio can be turned

off without affecting the learning process, so the package will be suitable for deaf users as well as those without audio capabilities on their computer.

Real Time Capabilities

The video playback can be jerky sometimes, even with a high speed processor and a high RAM specification. It can also take a while to load up the CD sometimes when playing the video which can make the playback jump and contain a double image.

Entertainment

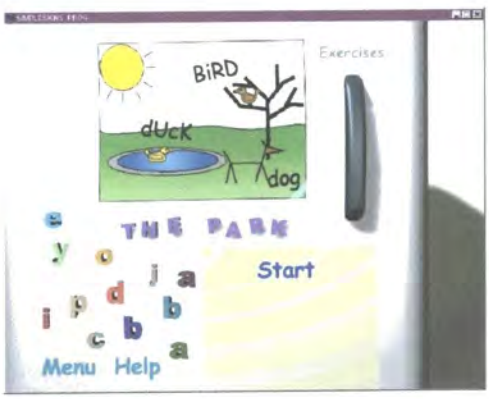
This system is very interactive which helps to keep the users attention. The tests offered by the system ask the user to recognize signs within a time limit. This introduces a competitive element which can help to motivate users. The time limit is quite short, however, which could lead to frustration for less quick users, or those with limited keyboard skills as they may not have time to type in the answer. The system allows two guesses and then gives the learner the answer. There is no overall score for the test, and the user can chose how much or how little he wishes to take as it is done question by question.

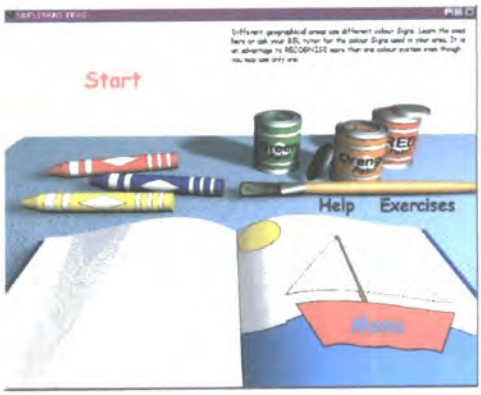
System 3: Simple Signs (SS)

System Usability Characteristics

SS starts with an introduction to the girl who does the signing in the video clips. It appears in a small window, which cannot be enlarged. It then loads the main menu as shown below.

This offers the different topics that the user can select. The graphics and arrangement seem to be very much aimed at children, as is shown from the topic screens show below, which are for Animals and Colours respectively.





Language Specifics

The signs are shown in both BSL and fingerspelling which is good because it aids familiarity with fingerspelling and the concept of spelling words which the user does not know the BSL for, but it may also be confusing as there is no indication of where the signs are different or even that this is what is happening.

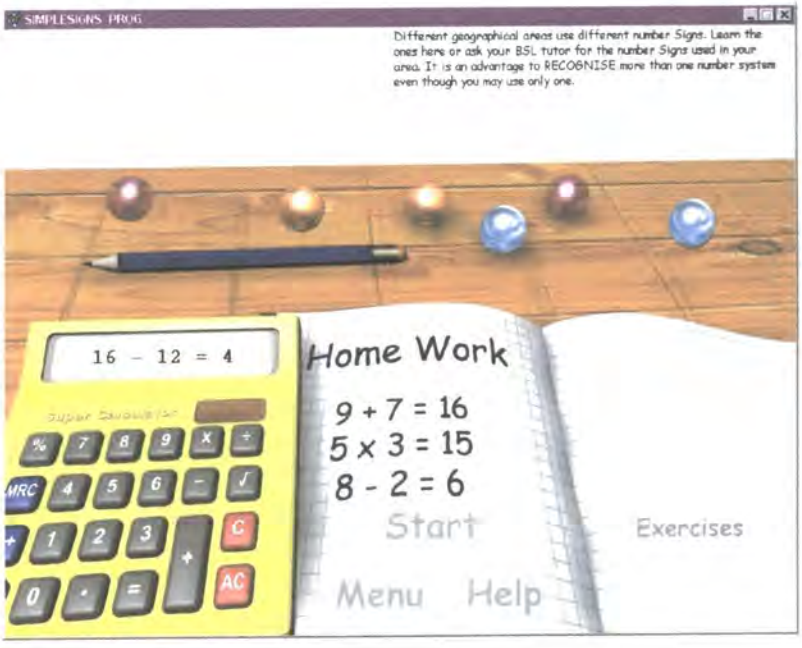
SS only offers vocabulary and no grammar when teaching sign language. This system may be suitable for those already proficient in basic signing, who may need to extend their repertoire. There is provision for a Dictionary function, however, this is very limited, for example, there are only 3 words listed for the letter 'F' – first, for and from.

Pedagogical Nature of Application

SS offers a wide range of simple vocabulary, but no grammar or syntax. The vocabulary is single words only with no sentences. It is taught by the simple method of displaying a short video clip of the sign and then asking the learner to select from three choices as to which one they think is the translation. This can aid with recognition of signs but the user must learn by trial and error which may not be very effective.

Navigation Capabilities

The navigation is excellent. The menu at the start is clear and states what topics are in each module. Within each module there are textual links (as shown below) back to the Menu and to the next sign. The links and clear and consistent. One aspect which is a little confusing is that there is always a link 'Exercises' which simply links to a list of words/phrases which are in that category.



Use of Technology

SS makes use of short video clips together with text and audio to make it more interesting for the learner.

Real Time Capabilities

The video clips can be a little jerky due to the time it takes for the CD to load each time.

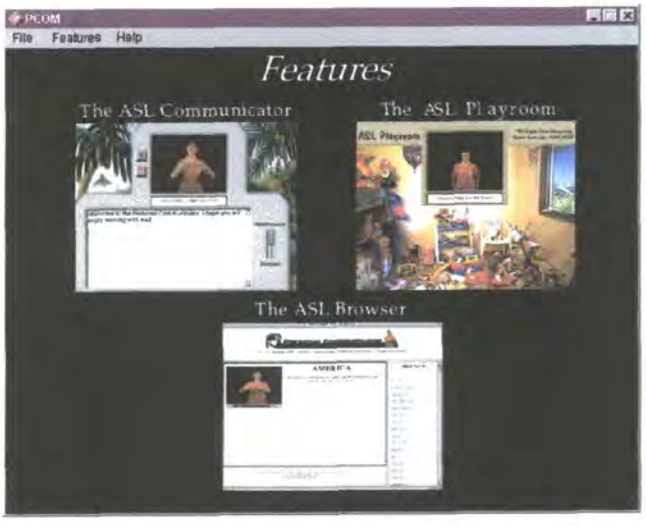
Entertainment

SS is a good concept but may struggle to retain the audiences attention due to the lack of teaching, i.e. it just involves guessing which sign is which, and the lack of challenge.

System 4: Personal Communicator (PC)

System Usability Characteristics

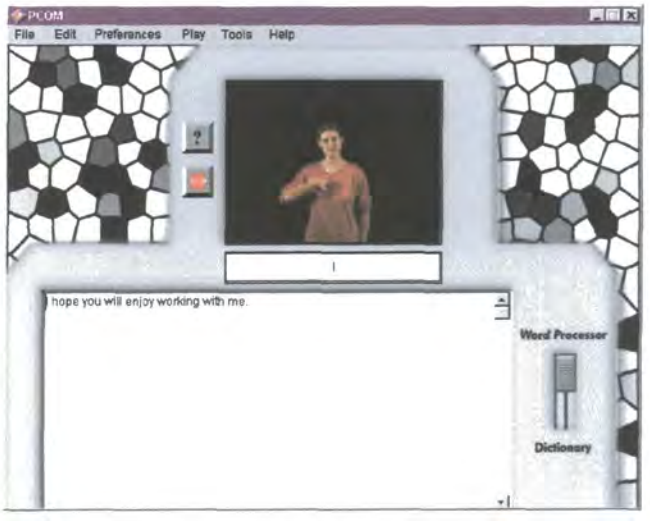
The PC loads from the CD-ROM, but if you don't have the plug-ins for QuickTime (movie player) or the Text-to-Speech program it will insist on installing them which may be confusing for some users and may cause problems for inexperienced computer users.



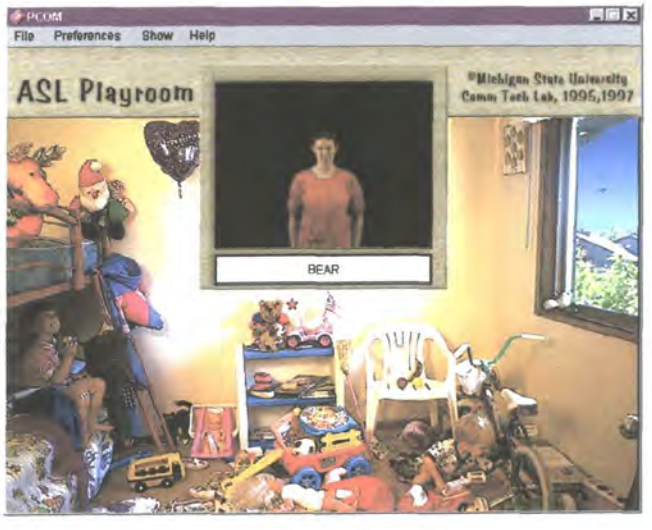
There is no indication from the menu (shown above) which section does which tasks.

The three different sections are the ASL Communicator, the ASL Playroom and the ASL Browser.

The Communicator offers a translation service where the user enters a word or phrase, in this case, *I hope you will enjoy working with me* and the program will provide the translation. This is a simple word for word translation which is not the correct way to translate ASL. The video is too small to clearly see the signers hands and to see what they are signing.



The ASL Playroom is an odd idea as it is a single screen where the user has to click on different objects within the room and then they are slightly animated, i.e. the bus drives across the floor, or the balloon bursts. The signer will then sign the word. Again, the signer is too small to be able to accurately identify the signs. This room gets very boring after the first one or two objects. It is also quite difficult to identify which objects will animate as they are all crammed in together.



The ASL Browser opens with a full screen window and it seems to be simply a dictionary function as it asks the user to select a word from the side list and then plays the video. The video clip is very small compared with the window and is hard to see, as before. The system does offer a text description of the sign, which is very useful. However, this is quite brief. The browser uses the users own Web Browser to display the information. This is confusing as the rest of the program doesn't do this. It is not well designed as the column on the right hand side needs a horizontal scroll bar because the words are too long to fit, yet there is a lot of white space in the rest of the window. Also, the system uses QuickTime to play the video clips, but it offers no

instructions on how to play the clips (they don't start automatically), and the controls available are very small.

Language Specifics

This system uses ASL and uses fingerspelling when the system does not recognize the word entered. This at least means the system will not fail when it does not recognize text. The dictionary section offers a large vocabulary, including words such as *acquiesce* and *reindeer*.

Pedagogical Nature of Application

This system does not teach any sign language or deaf awareness. It is simply a translation and dictionary system.

Navigation Capabilities

Navigation around the package is generally by means of the drop down menus at the top of the screen, as used in the majority of windows programs. This makes it recognizable by any experienced users but may confuse novices who may not know about drop down menus. One problem is that there is no indication about what each section does, not even in the help files.

Use of Technology

The PC makes use of video, text, audio and animation. The video clips can be slow to load but they are not jerky and run quite fast.

Real Time Capabilities

The video clips can be slow to load but they are not jerky and run quite fast.

Entertainment

The PC is not very absorbing as a system. It certainly wouldn't be interesting enough to use for more than a short time.

Conclusions

From the study of the systems, certain conclusions can be made about the design and implementation of these products.

- Only one of the four programs actually attempted to teach material as opposed to informing the user. SLFE made use of recognized teaching methods by basing the package on a lesson by lesson approach. The rest of the systems simply offered a "guess the sign" approach which will not keep the users attention for long and will not teach the language in a very effective way.
- Three of the systems do not make use of the full screen area. This is frustrating to users who may not be able to see the text or video properly as it is too small. The background can be distracting.
- Video clips can be used with great effect, but can be jerky or slow to load if systems are not fast enough, or the files are very large.
- The more interactive systems were more interesting and could keep the users interest for longer.
- All the systems offered the chance to play the video clips as many times as the user required.
- None of the systems indicated how far through the material the user was, so it was a little disconcerting to not know how many more screens there were to go.

Recommendations

From the conclusions identified above, recommendations for the new system can be made:

- Design the program so that it takes up the whole screen, otherwise it is too small to read the text properly, and the background can be distracting. People will automatically attempt to make the screen bigger and will be annoyed when they can't.
- Make the navigation system obvious, simple and consistent. Ensure there are textual alternatives if the icons used are graphics only.
- The lessons should be arranged in a logical manner.

- Use a lesson based approach, which progresses from level to level so that users interest and attention is kept, and they can build on the information they gained from previous levels.
- Use short tests in order to challenge the user, but make them realistic, i.e. in terms of timing and length.
- Video clips are essential when attempting to teach sign language. Textual descriptions of the movements and actions are also desirable as they offer another way for the learner to grasp the concept.
- Systems should be interactive, and should keep prompting the user to take part and take in the information.
- Ensure that text is large enough and clear enough to read, especially if the system may be used by partially sighted people.
- Let the user select when and how many times they wish to view the video clips.
- Topics should be reasonably short in length otherwise users will get bored. If possible, an indicator should tell the user how far thorough the particular module or topic they are so that they can judge how long they have to go.

These are not all the requirements for the new system, but a selection that can be inferred from studying the existing systems.

Appendix C – Market Survey

We Need YOUR Help!!

CACDP are setting up a new website which will offer course material for deaf awareness and BSL. It will also offer the chance to take the examination online. We need some information from you, our potential candidates, so please can you fill in the following questionnaire in order to help us. All information will be confidential. Thank you.

Personal Information

Date of Birth: □□/□□/□□□□

Postcode: □□□□-□□□

Nearest Town: ----------

County: ----------

First Language:

□ English

□ BSL

□ Other (Please Specify)

Are you?

□ Hearing

□ Hard of Hearing

□ Deafened

□ Deaf

Access to Computers and Experience of the Internet

Do you use a computer? (Tick all that apply)

□ At Work

□ At Home

□ At school/college/university

□ Elsewhere (Please Specify)

□ I don't use a computer

If you don't use computers, why not?

Do you use the Internet? (Tick all that apply)

- ☐ At work
- ☐ At home
- ☐ At school/college/university
- ☐ Elsewhere (please specify)
- ☐ I don't use the Internet

If you don't use the Internet, why not?

What do you use the computer for? (Tick all that apply)

- ☐ Email
- ☐ World Wide Web
- ☐ Writing letters, keeping records etc
- ☐ Online Training/Learning
- ☐ Games
- ☐ Other (please specify) ----------

How often do you use a computer?

- ☐ Every Day
- ☐ Several times a week
- ☐ Once a week
- ☐ Less than once a week
- ☐ Never

How often do you access the Internet?

- ☐ Every Day
- ☐ Several times a week
- ☐ Once a week
- ☐ Less than once a week
- ☐ Never

How long do you spend on the Internet each week?

- ☐ No time
- ☐ Less than an hour
- ☐ 1 – 5 Hours
- ☐ 5 – 10 Hours
- ☐ Greater than 10 hours

What type of connection do you use?

- ☐ Dial up using a modem
- ☐ ISDN
- ☐ ADSL
- ☐ Local Area Network, i.e. at work
- ☐ Other (please specify)
- ☐ I don't know

Have you ever tried any form of learning using web sites or CD-ROMs?

- ☐ Yes
- ☐ No

If yes, what subject was this on?

What did you think of the experience?

Name of website or CD-ROM if known:

Would you consider using the proposed website:

- ☐ Instead of traditional methods (i.e. face to face learning)
- ☐ In addition to traditional methods
- ☐ Not at all

Thanks for your help, if you have any more questions or suggestions, please contact CACDP.

Appendix D – Market Survey Results

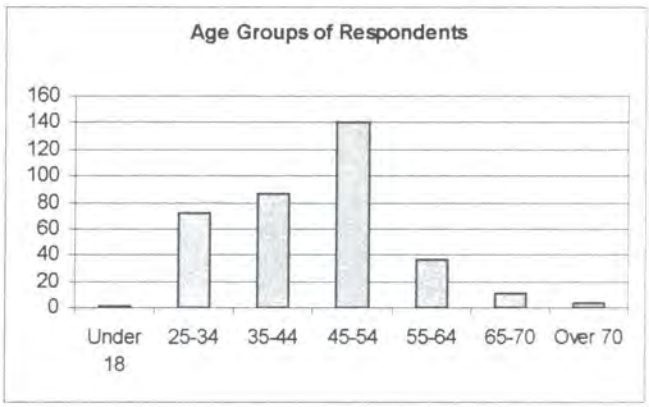
Questionnaire Response Statistics

There were 1,911 questionnaire's sent out to CACDPs customers via the newsletter. We received 364 responses (to 21-02-01). The statistics of the responses are as follows:

Personal Information

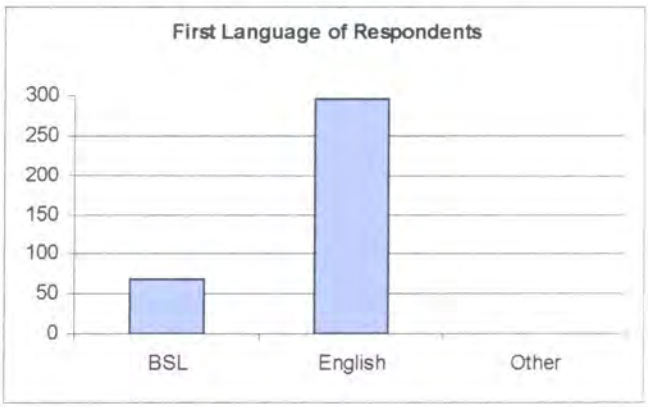
Ages of Respondents:

Age Group	People
Under 18	1
25-34	72
35-44	87
45-54	140
55-64	37
65-70	11
Over 70	4



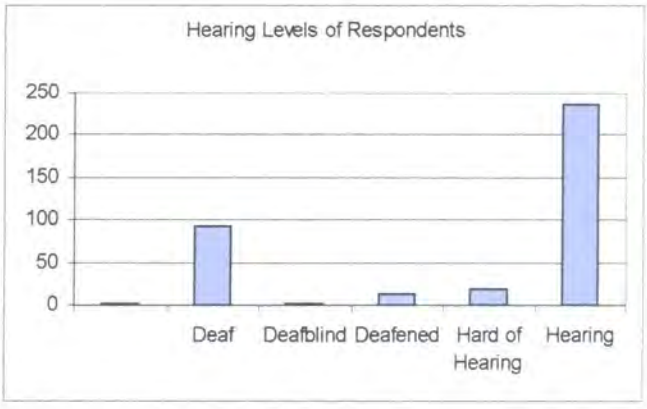
Languages:

Language	People
BSL	68
English	295
Other	1



Hearing Level:

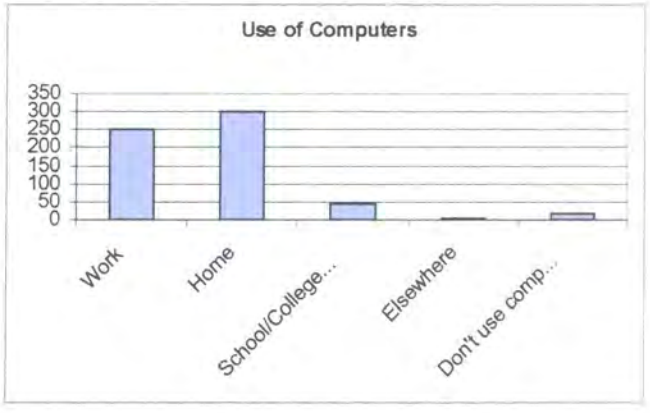
Hearing Level	People
No response	2
Deaf	92
Deafblind	1
Deafened	14
Hard of Hearing	18
Hearing	237



Access to Computers and Experience of the Internet

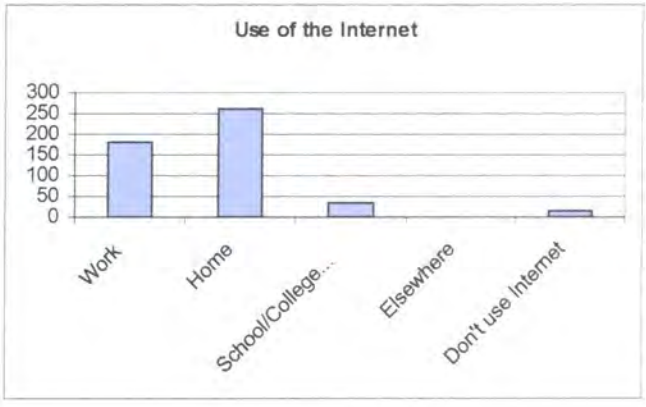
Where do people use computers?

Location	People
Work	253
Home	302
School/College/Uni.	43
Elsewhere	4



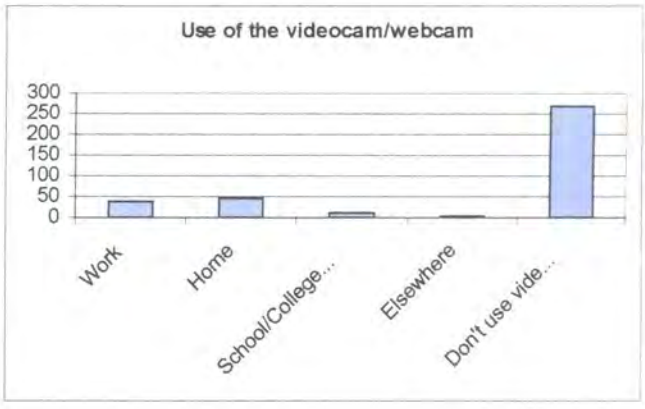
Use of the Internet:

Location	People
Work	182
Home	262
School/College/University	35
Elsewhere	1
Don't use Internet	17



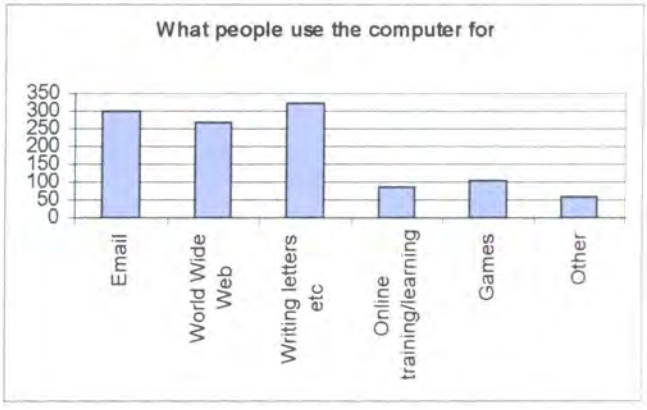
Use of Videocam/Webcam:

Location	People
Work	39
Home	45
School/College/Uni.	10
Elsewhere	3
Don't use	270



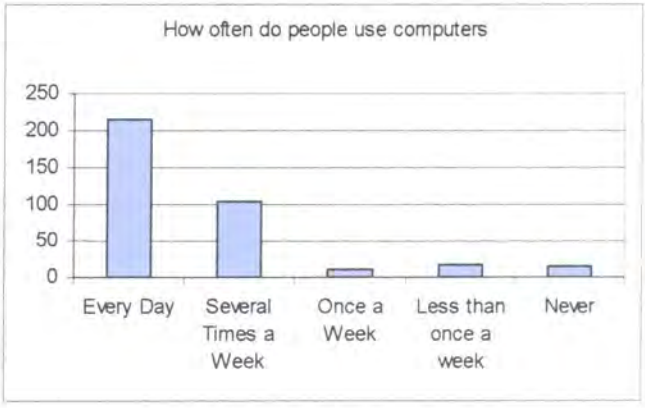
Use the Computer for:

Use computer for	People
Email	299
World Wide Web	270
Writing letters etc	323
Online training/learning	85
Games	103
Other	59



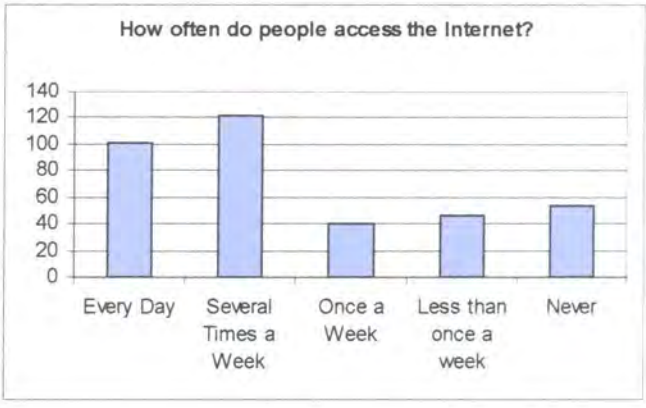
How often do they use the computer?

Frequency	People
Every Day	216
Several Times a Week	104
Once a Week	10
Less than once a week	17
Never	15



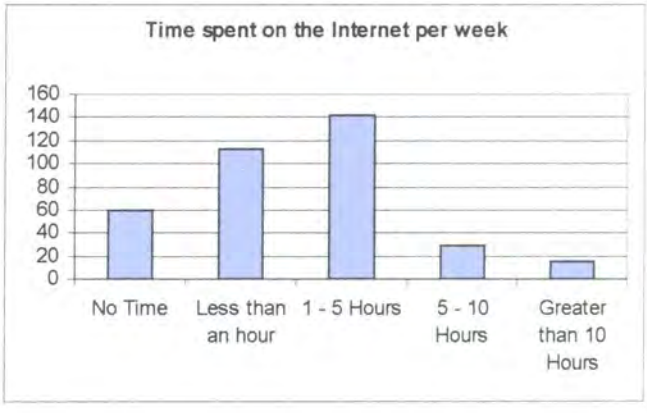
How often do people access the Internet?

Frequency	People
Every Day	101
Several Times a Week	122
Once a Week	40
Less than once a week	46
Never	53



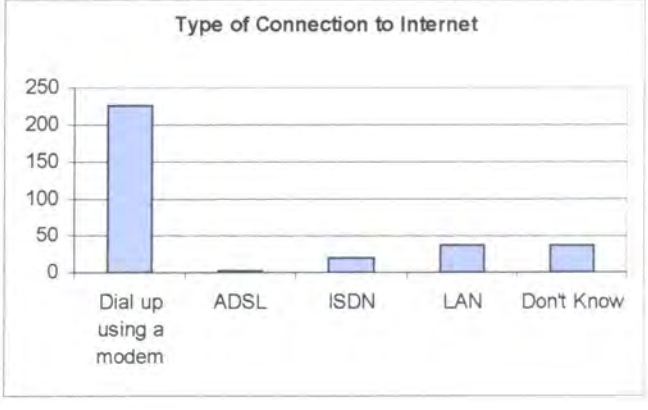
How long do people spend on the Internet:

Length of term	People
No Time	60
Less than an hour	113
1 - 5 Hours	142
5 - 10 Hours	29
Greater than 10 Hours	16



Type of connection:

Connection	People
Dial up using a modem	227
ADSL	2
ISDN	19
LAN	37
Don't Know	37



Tried online learning:

Tried	People
Yes	129
No	235

Use of new website:

New Website	People
Instead	69
As Well	258
Never	52