The Agile Web Engineering (AWE) Process

Andrew Gregory McDonald

Submitted for the Degree of Doctor of Philosophy
University of Glasgow
Department of Computing Science

August 2004

Abstract

During the late 1990s commerce and academia voiced major concerns about the problems with development processes for Web Engineering. These concerns primarily centred upon the perceived chaotic and 'ad-hoc' approach to developing Web-based applications in extremely short time-scales when compared to traditional software development. Based on personal experience, conducting a survey of current practice, and collecting supporting evidence from the literature, I proposed a set of seven criteria that need to be addressed by a successful Web engineering process:

1. Short development life-cycle times;
2. Delivery of bespoke solutions and different business models;
3. Multidisciplinary development teams;
4. Small development teams working in parallel on similar tasks;
5. Business analysis and evaluation with end-users;
6. Requirements capture and rigorous testing;
7. Maintenance (evolution) of Web-based applications.

These seven criteria are discussed in detail and the relevance of each to Web engineering is justified. They are then used to provide a framework to assess the suitability of a representative sample of well-known software engineering processes for Web engineering. The software engineering processes assessed comprise: the Unified Software Development Process; Dynamic Systems Development Method; and eXtreme Programming.

These seven criteria were also used to motivate the definition of the Agile Web Engineering (AWE) process. AWE is based on the principles given in the Agile Manifesto and is specifically designed to address the major issues in Web Engineering, listed above. A number of other processes for Web Engineering have been proposed and a sample of these is systematically compared against the criteria given above. The Web engineering processes assessed are: Collaborative Web Development; Crystal Orange Web; Extensions to the Rational Unified Process; and Web OPEN.

In order to assess the practical application of AWE, two commercial pilot projects were carried out in a Fortune 500 financial service sector company. The first commercial pilot of AWE increased end-user task completion on a retail Internet banking application from 47% to 79%. The second commercial pilot of AWE used by an Intranet development team won the company's global technology prize for 'value add' for 2003. In order to assess the effect of AWE within the company three surveys were carried out: an initial survey to establish current development practice within the company and two further surveys, one after each of the pilot projects.

Despite the success of both pilots, AWE was not officially adopted by the company for Web-based projects. My surveys showed that this was primarily because there are significant cultural hurdles and organisational inertia to adopting different process approaches for different types of software development activity within the company. If other large companies, similar to the one discussed in this dissertation, are to adopt AWE, or other processes specific to Web engineering, then many will have to change their corporate goal of a one size fits all process approach for all software technology projects.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First and foremost I would like to thank my Mother (Kay McDonald), Father (Andrew McDonald) and my supervisor (Prof. Ray Welland), whose support, encouragement, guidance and tolerance made completing this dissertation possible. There are a small number of other individuals I would like to thank for assisting me during my research. Their names appear in no particular order: Jack Gibson, Andrew Massey, Huw Evans, Jim Devine, Noel Winstanley, Ted Cowan, Matthew Chalmers, Malcolm Atkinson, Alan Radcliffe, Alistair Hutton, Derek Gott, Paul Smith, Stuart Low, John Wilson, Richard Wilson, Marcia McDonald, Eva McDonald, Hugh Clarke, Hugh Maguire, Tracey Connor and Gerry Hodgett.

In addition to the individuals who assisted with the research, thanks also goes to the individuals within the companies and organisations that participated in the surveys presented within this dissertation. In particular I would like to pay special thanks to the financial services company that employed me during my Ph.D. Internship.

Declaration

I hereby declare that this thesis has been composed by myself (Andrew McDonald), that the work herein is my own except where otherwise stated, and that the work presented has not been presented for any university degree before.

The author's original work presented in this dissertation has formed the basis of a number of publications (McDonald & Welland 2001a, McDonald & Welland 2001b, McDonald & Welland 2001c, McDonald & Welland 2002, McDonald & Welland 2003, McDonald & Welland 2004a and McDonald & Welland 2004b) that have been co-authored with my supervisor Prof. Ray Welland.

SIGNED:______________________________ DATE:__________________

1 Introduction

The growth of the World Wide Web (WWW) over the past decade has been phenomenal. The dramatic impact the WWW has had on business and society over the past ten years has forced a number of radical paradigm shifts in the way business and society function. In many countries the education, entertainment, financial services, health care and government sectors all now depend upon the functioning of Web-based systems as part of their primary operations. Spending on Internet, Intranet and Extranet applications has become a multi-billion pound global industry, with estimated spending on e-business initiatives in 2001 averaging 17% of companies' IT budgets in the United States of America (Rubin and Butler 2003, p.200). Revenues from business to business (B2B) e-commerce sales in the United States of America alone are estimated to be over one trillion US dollars during 2004 (Rubin and Butler 2003, p.166). The sums of resources used to develop Web-based applications and the monies passing through these applications are growing in significance. Despite the critical role played by Web-based applications in many areas of modern society, a number of indicative reports McDonald & Welland (2001a), McDonald and Welland (2001b), Barry & Lang (2001), Lowe and Eklund (2002), Taylor et al. (2002a), Taylor et al. (2002b) and Zhou and Stålhane (2004) have emerged in the literature illuminating the ad-hoc and chaotic manner in which Web-based applications are being developed in practice.

During the late 1990s a number of software engineers expressed major concerns about the way Web-based applications were being developed (Pressman et al. 1998), which helped to give rise to the birth of the Web engineering community (Murugesan et al. 1998). These concerns are highlighted by the somewhat unusual characteristics that describe Web application development in comparison to traditional software development, and the seemingly poor suitability of traditional software engineering approaches and techniques to the development of Web-based systems (Pressman 2000a). The following quote from one of the earliest Web engineering papers (Murugesan et al. 2001) serves as a "broad objective" of this subject area:

> Web engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications.

Murugesan et al. (2001) list a number of characteristics that differentiate traditional software development from Web-based development. These include the Web's legacy as an information medium rather than an application medium and the evolutionary nature of Web-based applications. Deshpande and Hansen (2001) argue that Web engineering requires influence from a wide variety of disciplines in addition to software engineering, information engineering and computing science. Murugesan et al. (2001) present the argument that Web engineering requires a process, to address the multidisciplinary nature of Web engineering (Deshpande, Murugesan & Hansen 2001). The following four sections of this chapter comprise the thesis statement, the research methods employed, a dissertation overview and the research contribution presented within this dissertation.

1.1 Thesis Statement

My hypothesis is that developing Web-based applications (Web Engineering) has specific characteristics that differ from those normally assumed for software development processes. Therefore, a different type of development process is required for Web engineering.

This thesis attempts to answer the following research questions concerning the above hypothesis:

1. Is it possible to define a set of criteria that a Web engineering process must fulfil?
2. Can a new development process be defined to meet the criteria for Web engineering process?
3. Can it be shown that such a new process is better suited to Web-based application development than existing plan driven, agile and alternative Web engineering processes?
4. Is it possible to demonstrate that this new Web engineering process can be used successfully in practice?

1.2 Research Method

Zelkowitz and Wallace (1998) describe and present a taxonomy for software engineering experimentation that comprises twelve different experimental approaches. Each of the twelve experimental approaches are categorised into one of three broad categories: observational methods collect data as a project develops; historical methods collect data from a project that has already been completed; controlled methods provide for multiple instances of an observation for statistical validity of the results. Table 1, reproduced from Zelkowitz and Wallace (1998), lists and describes the twelve different experimental approaches, categorises them under one of observational, historical or controlled methods, and provides short descriptions of the strengths and weaknesses of each experimental approach.

Experimental Approach	Category	Description	Weaknesses	Strengths
Project Monitoring	Observational	Collect development data	No specific goals	Provides baseline for future; Inexpensive
Case Study	Observational	Monitor project in depth	Poor controls for later replication	Can constrain one factor at low cost
Assertion	Observational	Use ad hoc validation techniques	Insufficient validation	Serves as a basis for future experiments
Field Study	Observational	Monitor multiple projects	Treatments differ across projects	Inexpensive form of replication
Literature Search	Historical	Examine previously published studies	Selection bias; treatments differ	Large available database; Inexpensive
Legacy	Historical	Examine data from completed projects	Cannot constrain factors; data limited	Combines multiple studies; Inexpensive
Lessons Learned	Historical	Examine qualitative data from completed projects	No quantitative data; cannot constrain factors	Determine trends; Inexpensive
Static Analysis	Historical	Examine structure of developed product	Not related to development method	Can be automated; Applies to tools
Replicated	Controlled	Develop multiple versions of product	Very expensive; Hawthorne effect	Can control factors for all treatments
Synthetic	Controlled	Replicate one factor in laboratory setting	Scaling up; interactions among multiple	Can control individual factors; moderate cost

			factors	
Dynamic analysis	Controlled	Execute developed product for performance	Not related to development method	Can be automated; Applies to tools
Simulation	Controlled	Execute product with artificial data	Data may not represent reality; not related to development method	Can be automated; Applies to tools; Evaluation in safe environment

Table 1. Summary of software engineering experimental approaches. Reproduced from Zelkowitz and Wallace (1998).

There were four research methods or experimental approaches (Zelkowitz and Wallace, 1998) employed during this work. The experimental approaches employed comprised:

1. Case Study. The author's personal experience on Web engineering projects, pre commencement of Ph.D. research 1996-1999 and during the last three years of the author's Ph.D. research 2001-2004;
2. Literature Search. Literature surveys to find supporting evidence for the personal experience observations and to identify other Web engineering processes;
3. Static Analysis. A systematic comparison of different processes to verify the relevance of the criteria and establish that existing practices do not meet the needs of Web engineering;
4. Lessons Learned. A series of structured interviews were employed using a qualitative one-to-one interview technique for gathering the opinions and experience of others during Web application development, see Appendices 1, 2, 3, 6, 7 & 8.

1.3 Overview

The objective of this research was to describe the areas a successful Web engineering process needs to address, and to develop and evaluate a process specifically for Web engineering. The stages used to achieve this objective are described below, chapter by chapter with reference to where more detail is presented in the dissertation.

Chapter 2 provides a working definition for process and the elements used to describe processes throughout the rest of the dissertation.

Chapter 3 presents a taxonomy for Web-based applications and presents the author's personal experience developing two large Web sites during the 1990s. It also describes the first published survey of Web engineering in practice and reviews similar surveys that have subsequently appeared in the literature. Chapter 3 concludes by identifying the criteria for a Web engineering process.

Having presented empirical evidence to support the criteria for a Web engineering process in chapter 3, chapter 4 goes on to discuss the criteria in detail, distinguishing between the major differences required by a Web engineering process in comparison to traditional software engineering processes.

Chapter 5 describes three traditional software engineering processes across the plan-driven, rapid application development (RAD) and agile software engineering process spectrum and evaluates each against the criteria for a Web engineering process. Chapter 5 concludes with a critical assessment of how the three traditional software engineering processes support the criteria for a Web engineering process.

Chapter 6 describes the Agile Web Engineering (AWE) process in detail, presenting the rationale behind the development of this process for Web engineering. Chapter 6 concludes by comparing the AWE process against the criteria for a Web engineering process.

Chapter 7 evaluates four other Web engineering process approaches that have emerged in the literature over the past five years. It concludes with a summary of how these other Web engineering process approaches support the criteria for a Web engineering process.

From October 2001 until September 2002 the author undertook a Ph.D. Internship within a Fortune 500 Financial Services Company, the objective of which was to evaluate the AWE process in a commercial setting. Chapter 8 evaluates the company's in-house process for building Web applications against the criteria for a Web engineering process. In addition the details of an initial survey established how the company, with extensive experience of software development, was coping with the changing demands of developing Web-based applications and other software projects where time-to-market pressures are a major driver.

Chapter 9 describes the first commercial pilot of the Agile Web Engineering (AWE) Process on a retail Internet banking application. This first pilot significantly increased end-user task completion rate for basic Internet banking tasks from 47% to 79%. Chapter 9 presents a second survey (post-AWE pilot) of employees within the financial services company, who were involved in the promotion and use of AWE on its first commercial pilot. Chapter 9 concludes by using Boehm and Turner's home grounds analysis to determine the suitability of an agile process approach to Web application development, realised through AWE, within the company.

Chapter 10 discusses the second commercial usage of AWE by a development team, one of thirteen, on a 53 million pound Intranet project. This team won the company's global technology award for 'value add' in early 2004. Chapter 10 discusses in detail the elements of AWE explored during its second commercial usage. Chapter 10 concludes with a third survey of the Intranet team who used AWE during its second commercial usage including a discussion of the team's perceptions of the AWE process.

Chapter 11 presents the conclusions and further work. These are then followed by the Appendices, Glossary, References and Index.

1.4 Research Contribution

The criteria for a Web engineering process, discussed in chapters 3 and 4, presents a mechanism for other practitioners and researchers to evaluate and contrast processes for Web engineering. The criteria were presented in a technical report (McDonald & Welland 2001a). In addition, two other publications based upon the work in chapter 3 have been peer reviewed and published (McDonald & Welland 2001b) and (McDonald & Welland 2002).

Chapter 5 presents a comparison of commercial software engineering processes against the criteria for a Web engineering process. The processes evaluated include plan-driven, rapid application development and agile processes. Chapter 5 describes in detail why traditional software engineering processes are poorly suited to addressing the criteria for a Web engineering process.

The Agile Web Engineering (AWE) Process presented in a technical report (McDonald & Welland 2001c) is discussed in Chapter 6. AWE is the first agile process for Web engineering developed specifically to address the criteria, discussed in chapters 3 and 4, for a Web engineering process. Recognition of the contribution of this work within the software and Web engineering communities is reflected by the reference and discussion of the AWE process technical report in Pressman (2004), one of the major software engineering textbooks. In addition, there has also been a peer reviewed publication (McDonald & Welland 2003) based on features of AWE at the International Conference on Web engineering 2003.

Chapter 7 presents a comparison of existing commercial Web engineering processes against the criteria for a Web engineering process. Chapter 7 describes in detail the suitability of other Web engineering processes to addressing the criteria for a Web engineering process. This chapter is the basis of a peer reviewed publication at the International Conference on Web Engineering 2004 (McDonald & Welland 2004a).

The survey material presented in chapters 8 and 9, the pre- and post- AWE pilot surveys, and the material presented in section 9.3 form the basis of a paper submitted to the Journal of Web Engineering (McDonald & Welland 2004b).

AWE's second commercial influence on a European wide Intranet project with a budget of 53 million pounds sterling is discussed in chapter 10. The work carried out by this Intranet team using AWE won the global technology award for 2003, in early 2004, within the financial services company. Chapter 10 describes the perceptions of AWE within a Fortune 500 financial services company, and concludes by identifying some of the hurdles encountered when trying to get a large enterprise technology operation to adopt a different process specifically for Web engineering. Chapter 11 suggests further possible areas of research within the field.

2 The Role of Process in Software Engineering and Web Engineering

This chapter focuses on the role of process in software engineering and Web engineering. Section 2.1 provides a working definition of process, with section 2.2 discussing the elements used to describe processes throughout the rest of the dissertation.

2.1 A Working Definition of Process

Software engineering as a discipline has been an active area in both research and practice for more than thirty-five years. Software engineering endeavours to provide guidance to those involved in software development. The broad objectives of success in software engineering are to produce systems that are: reliable, robust, capable of addressing the issues within the project's problem space, delivered on time and within budget. These success criteria have proved elusive over the years (Standish Group 1995) particularly with the adoption of new technologies and different applications for software systems.

There exists no uniformly agreed definition of software engineering process. The objective of this research is not to define software process. Rather, the following discussion serves to help illustrate the complex nature of software development and the different perspectives held with respect to the definition of a software engineering process.

Many software process approaches have been presented over the past forty years (Benington 1956, Royce 1970, Boehm 1988, Beck et al. 2001). Software processes try to tackle and provide guidance on the problems associated with software engineering projects. Recently many have begun to classify software processes as either monumental or plan-driven (also referred to as heavyweight) or agile (also referred to as lightweight) including Fowler (2001) and Boehm and Turner (2003a) and (2003b).

The plan-driven process, the Unified Software Development Process (Jacobson, Booch & Rumbaugh 1999), has been influenced heavily by the iterative and incremental risk driven developments (Boehm 1998) derived in the 1980s and developments in the object-oriented design community (Jacobson et al. 1992) in the 1990s. Plan-driven processes often try to cover a wide range of software development activities, relying on organisations and their process stakeholders to create a subset of the process and its supporting techniques and tools to suit their particular organisational and project needs. Most plan-driven processes are process oriented and predictive in nature, determining very early in the development life-cycle the problem, design and plan for developing the proposed solution. The following quote from Jacobson, Booch and Rumbaugh (1999, p. xviii) gives a definition of software process from three plan-driven advocates and practitioners:

> A process defines *who* is doing *what, when,* and *how* to reach a certain goal. In software engineering the goal is to build a software product or to enhance an existing one. An effective process provides guidelines for the efficient development of quality software. It captures and presents the best practices that the current state of the art permits. In consequence, it reduces risk and increases predictability. The overall effect is to promote a common vision and culture.

At the opposite end of the process spectrum lie the recent process developments classified as agile processes which follow the agile manifesto (Beck et al. 2001) and are seen as lightweight processes in comparison to traditional plan-driven processes. Agile processes include Extreme Programming (Beck 2000) and Dynamic Systems Development Method (DSDM) (Stapleton 1997). Agile processes tend to focus on specific kinds of software development activities and have common characteristics such as small development teams and shorter development life-cycle times, in addition to focussing on software deliverables as opposed to documented deliverables. Agile processes encourage small teams of highly skilled developers to start with the initial agile or lightweight process and expand it to suit their organisation and project needs. Agile processes are people oriented and they encourage and embrace change, allowing nearly the full project development time to define the problem, design and implement the proposed solution in its entirety. The following quote from Cockburn (2002, p. 117) gives a definition of software process from an agile process advocate and practitioner:

> How activities fit together over time, often with pre- and post-conditions for the activities (for example, a design review is held two days after the material is sent out to participants and produces a list of recommendations for improvement). Process-intensive methodologies focus on the flow of work among the team members. Process charts rarely convey the presence of loopback paths, where rework gets done. Thus, process charts are usually best viewed ás workflow diagrams, describing *who* receives *what* from *whom.*

It can be argued that plan-driven processes allow organisations and projects to use a smaller skilled set of key software professionals (Boehm & Turner 2003a) who make more important higher-level project decisions. These critical high-level decisions are then used to guide less knowledgeable or skilled developers who work at a lower-level where decisions are more rigid. Based within the framework of the higher-level decisions, lower-level decisions are seen to be less critical to project success. Many in the agile community (Cockburn 2000) and (Fowler 2001) would argue that this is flawed, and that lower-level decisions are just as critical to project success.

The objective of this research is not to debate plan-driven versus agile processes. Both approaches have their relative merits for different types of development project. There is no reason to suppose that the same process would be appropriate for developing an operating system and a cultural heritage Web site for a museum, although similarities may exist. Unlike traditional software engineering, Web engineering must cope not only with the development of software components but also a dramatic increase in the development of data and the inter-dependencies between both types of deliverable, as discussed in chapter 3. Most Web-based development is ultimately therefore focused on the delivery of bespoke solutions.

The following definition of a process is from an economist Howard Baetjer (1997, p. 85) also quoted in Pressman (2000b):

> Because software, like all capital, is embodied knowledge, and because that knowledge is initially dispersed, tacit, latent, and incomplete in large measure, software development is a social learning process. The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interaction between users and designers, between users and evolving tools, and between designers and evolving tools [technology]. It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of dialogue eliciting more useful knowledge from the people involved.

By evolving the above definition, to replace the word *tools* with *deliverables,* the above definition of software process should hopefully satisfy both plan-driven and agile process practitioners and advocates. Based on Baetjer's definition, the following will serve forthwith as a working definition of process for both software engineering and Web engineering:

> Because software, like all capital, is embodied knowledge, and because that knowledge is initially dispersed, tacit, latent, and incomplete in large measure, software development is a social learning process. The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interaction between users and designers, between users and evolving *deliverables*, and between designers and evolving *deliverables*. It is an iterative process in which the evolving *deliverables* themselves serve as the medium for communication, with each new round of dialogue eliciting more useful knowledge from the people involved.

2.2 A Working Description of Process

In chapters 5, 6, 7 and 8, a number of software and Web-based development processes are evaluated against the criteria for a Web engineering process, presented in section 3.4, and discussed in detail in chapter 4. In order to present a uniform overview of each process, processes are described using a template that identifies the principal elements of process in general. The focus of this research is not to present a unified definition of the elements that comprise a process, but to present a definition to assist the reader in understanding the characteristics of different processes. The definition presented below serves to assist the reader with a brief description of each evaluated process. The definition of the elements that comprise a process is based on work by Conradi, Fernström and Fuggetta (1994):

> A process comprises the 'real world' elements involved in helping people during the development and maintenance of a product. These 'real world' elements include: *deliverables*, *tools*, *activities*, *roles*, *process life-cycle*, *life-cycle phases* and *techniques.*

The elements of a process presented in italics above are described in more detail in Table 2. Again these are based on work by Conradi, Fernström and Fuggetta (1994). These elements are used to help provide a high-level description for each process being evaluated.

Each process evaluated against the criteria for a Web engineering process will be given a rank against each of the criteria, points 1-7. The rank will illustrate how strongly a particular process supports each criterion under the following echelon: *no support*, *weak support*, *partial support*, *strong support* or *very strong support*.

Process Elements	Element Description
Process Life-cycle	A high-level description of the workflow of the phases involved in producing deliverable(s). The process life-cycle should describe how the process begins and ends with respect to the process phases and deliverables. It should also describe the dependencies between the respective phases and their deliverable(s).
Life-cycle Phase	A subcomponent or stage of the process life-cycle, which often produces one or more deliverables.
Deliverable	A product or sub-product of the process and its respective phases.
Activity	A step within the process producing changes to the

	deliverable(s).
Tool	A computer program supporting or automating a part of the work related to an activity.
Role	Describes a set of responsibilities, understanding and skills necessary to accomplish a specific activity in the process.
Technique	A mechanism, including formal notation, employed by the process to help one or numerous roles produce or evolve deliverables.

Table 2. Description of the Elements Used to Provide a Brief Overview of a Process

2.3 Summary

This chapter focused on the role of process in software engineering and Web engineering. In order to structure the descriptions of processe evaluated later in the dissertation, a working definition of process is presented in section 2.1, and a working description of the elements that comprise a process is presented in section 2.2. The next chapter defines and presents empirical evidence for the criteria for a Web engineering process.

3 Characteristics of Web Engineering Projects

Web engineering is a constantly evolving area. Given the rapid change within the field, it is difficult to present an exhaustive classification for the types of systems delivered as part of Web engineering projects. The purpose of this research is not to present a definitive classification for the types of systems delivered as part of Web engineering projects. However, it is important that the reader understand clearly the types of systems that are being discussed in this dissertation. To address this, the following working taxonomy of Web-based applications, reproduced from Dart (2000, pp. 50-51), is presented to assist the reader:

- *Informational*: "Informational sites with read-only usage, commonly called *brochureware*; for instance, content that gives details about a company and its products. First-generation Web systems were of this type";
- *Delivery system*: "The site can download content to users or a resource; for example, download upgrades or plug-ins";
- *Customised access*: "Access is via a customised interface or based on user preferences; for example, my customized view of my Internet service provider's homepage or my favourite portal";
- *User-provided content*: "The user provides content by filling in a form on the site; for instance, a subscription to a magazine or registering for a company's seminar";
- *Interactive*: "Two-way interaction between sites, users, and resources as in business-to-business exchanges (B2B)";
- *File sharing*: "Remote users collaborate via common files stored on the site; for example, a team that coordinates on-line schedules or reviews documents";
- *Transaction oriented*: "The user buys something such as books or airline tickets";
- *Application service provider*: "The site represents rentable applications; the user rents an application on a per user, per month, or per transaction basis, such as a virus scan program or a testing suite";
- *Database access*: "The site uses databases that the user can query, directly or indirectly; for example, a supplier looks up a catalogue of parts";
- *Document access*: "The site provides access to libraries of on-line documents, such as a set of current corporate standards";
- *Workflow oriented*: "The site ensures that the process or workflow is followed, as in supply-chain management or order entry automation";
- *Automatic content generator*: "Robots or software agents automatically generate content. For example, 'bots' scour the WWW to bring back specific information, such as best price on products".

In the past decade a number of papers have appeared in the Web engineering literature referring to the characteristics of Web engineering projects (Dart 2000, Murugesan et al. 2001, Deshpande, Murugesan & Hansen 2001), defining the differences between Web engineering and software engineering projects (Deshpande & Hansen 2001) and discussing the suitability of traditional software engineering approaches to Web engineering (Pressman et al. 1998). However, these papers presented little empirical evidence to support their claims. The limited evidence presented in these papers is exemplified by Deshpande, Murugesan and Hansen (2001), who present evidence based on experience of building one Web site in academia. The lack of empirical evidence in the Web engineering literature was addressed by the publication of a number of surveys of Web development in practice:

McDonald and Welland (2001a), McDonald and Welland (2001b), Barry and Lang (2001), Taylor et al. (2002a), Taylor et al. (2002b), Lowe and Eklund (2002) and Zhou and Stålhane (2004). The first of these published surveys of Web engineering in practice (McDonald & Welland 2001a), was driven by the author's desire to see if others out with the academic Web engineering community were facing the same problems and challenges on their Web engineering projects.

During October, November and December 2000, the author conducted interviews with a number of people within organizations in the United Kingdom who are involved in the development of Web-based applications. The goals of the survey were to try to identify more clearly the major issues facing the development of Web-based systems, and to see which, if any, traditional software engineering practices and techniques were being successfully applied.

This chapter begins by describing some of the author's own empirical experience building two large Web sites, see sections 3.1 and 3.2. It goes on to describe the first survey of Web engineering in practice, see section 3.3. Based on the empirical evidence presented in sections 3.1-3.3, section 3.4 summarises the criteria for a Web engineering process. Such criteria describe the issues that a successful Web engineering process will have to address. The penultimate section (3.5) reviews subsequent surveys of Web engineering in practice that have appeared in the literature, providing further support for the criteria for a Web engineering process. The last section summarises the material presented in this chapter.

3.1 The Hunterian Museum and Art Gallery (HMAG) Web Site

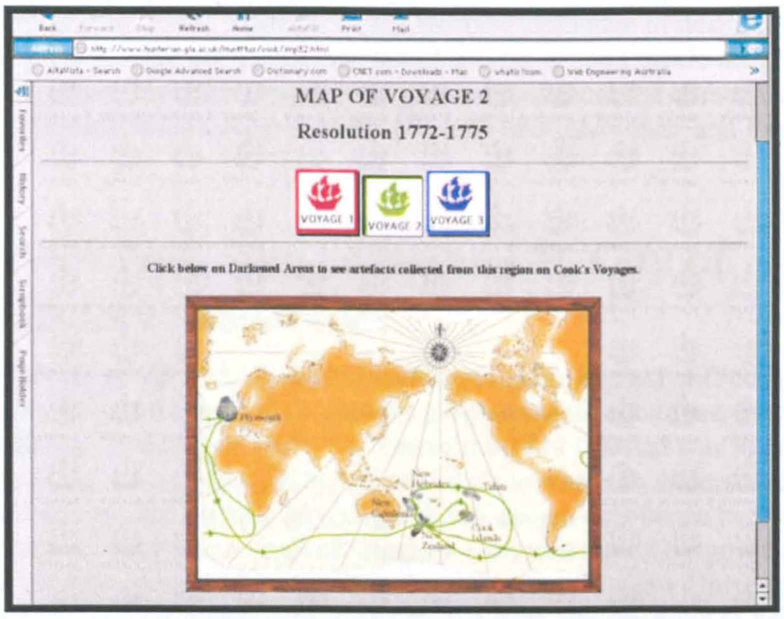
The Hunterian Museum and Art Gallery (HMAG) Web site (Devine et al. 2004) started as a University of Glasgow, Department of Computing Science (DCS) undergraduate team project in 1994. Since then over 100 students from DCS, working in teams and on solo projects, in addition to many members of staff within the HMAG and DCS have contributed to this world leading cultural heritage Web presence (Nielsen 2000a). The author gained valuable initial experience in Web engineering developing aspects of this site as an undergraduate student (McDonald et al. 1996). Additionally monitoring the long-term development of this Web site has provided valuable insights into the evolution of a large Web presence.

Each student project comprises a client within the HMAG, an academic DCS supervisor, the HMAG's Web designer, and the student or students themselves. Initially the majority of the work carried out on the HMAG Web site focused on small independent sections of the HMAG collections. The initial site structure developed in 1994, focused heavily on Human Computer Interaction (HCI) issues within the Web presence and the use of, at the time, new Web-based technologies such as Apple's QuickTime Technology suite (Apple 2004a). Over the past ten years the work carried out on the student projects has been recognised internationally, not just in the Museum and Cultural Heritage Web sector (Devine & Welland 2000), but also by other Web design gurus as an example of good practice (Nielsen 2000a, p. 132).

A characteristic of the Museum Web site is that the data is largely persistent. Once basic data is captured about artefacts it is not removed. Of course, new ways of presenting material often arise, such as replacing still images with QuickTime VR (Apple 2004b) object movies, or new ways of using objects for educational purposes. Often the data is enhanced as more information becomes available or new links are discovered to other related Web sites.

However, a major characteristic of the HMAG site is that it accumulates data and any evolution of the Web site must maintain the large investment in data capture.

A problem with relying on student projects is that one cannot expect the students to deliver the large volumes of data that the Museum requires in most cases. The students' challenge is to explore technology rather than capture data. Therefore, it is essential to ensure that each student project delivers a framework that can be evolved to include more data, and thus there is strong encouragement to design for evolution. For example, the author worked on a student project to present the artefacts associated with Captain Cook's voyages (McDonald et al. 1996). Here the challenge was to provide a navigational structure that would provide a coherent framework within which these artefacts could be displayed. The project explored the problems of image mapping and produced a visual framework based on active maps of Cook's three voyages, see Figure 1. The lower level structure developed below the maps is capable of expansion to include many artefacts but was only populated by the students with sufficient exemplars to illustrate the possibilities. However, the HMAG was confident that this structure could be evolved to add many more artefacts.


MAP OF VOYAGE 2

Resolution 1772-1775

Click below on Darkened Areas to see artefacts collected from this region on Cook's Voyages.

Figure 1. The HMAG Captain Cook Web Page

When the HMAG project started neither the HMAG nor DCS had any idea of how quickly the site was going to grow and how many diverse technologies were going to be included in it. The initial design, which achieved the desired aim in 1994, became dated and a major overhaul of the aesthetic design was required by 2002. This is one aspect of Web site maintenance that is quite challenging, changing the 'look and feel' of the site while maintaining the structure and content, which is an integral part of many Web applications. This is especially problematic in the HMAG context because of the large volume of persistent data, throwing away the existing site and replacing it with a brand new one was simply not an option.

As an increasing amount of material was developed for the HMAG, it became obvious that there were going to be major problems in carrying out routine maintenance on the site. There are many straightforward jobs, such as updating the technology when new versions of plug-ins become available and checking that links are still live, which need to be carried out. These correspond to the traditional software development activities of adaptive and corrective maintenance. As more technologies and more data collections were added these problems increased. It was considered unethical to use students to maintain the site through their undergraduate projects, and both HMAG and DCS insisted that only projects that explored new technologies or challenged the students through the new application of existing technologies would be proposed.

Although there are some specific problems that arise because the HMAG site has been built largely by student projects there is one specific characteristic that is very interesting. The HMAG site has a very high turnover of Web developers. Every year the HMAG starts afresh with two or three teams of students, perhaps fifteen new Web developers who are unfamiliar with the site and have no formal training in Web development. Due to the high turnover of developers there are problems with maintaining the corporate identity of the site; each team tends to add a few new variations on the agreed style. More importantly perhaps is that there is no Web development process that can be employed to teach the students, thus each team invents its own process and working practices. The HMAG site also lacks a notation for describing the current design and its rationale that could be passed on from year to year.

Even after a couple of years of developing the HMAG Web site it was clear that evolution problems were emerging. Thus, when presented with the challenge of designing a Web site in 1996 for a festival in 1999 the author was determined to try and tackle the problems of an evolving Web presence. The following section goes on to discuss the author's experience during the development of the Glasgow 1999: UK City of Architecture and Design Festival Web site.

3.2 The 1999 Web Site: Glasgow UK City of Architecture and Design

Glasgow won the title of UK City of Architecture and Design against stiff competition from other UK cities in 1996. The Festival launched a programme of initiatives, events and collaborations that ran throughout 1999. The aim of the 1999 Festival was to position Glasgow internationally as a major European city of ideas where an understanding of the architectural and design process and the recognition of design excellence became inherent in its people, business and culture. As part of Glasgow University and Glasgow School of Art's contribution to 1999, the author and a number of others from Glasgow University and the Glasgow School of Art, built and hosted the Festival's Web site (referred to hereafter as the 1999 Web site).

The 1999 Web site was deemed a great success during 1999. It was commended at the Scottish Enterprise Network Winners at the Web Competition, under the 'Innovation and Design Category', for the best example of good design practices in creating an Internet/Intranet site and the novel deployment of Internet technologies. In addition, the Festival's Education Director, Dr. Stuart Macdonald was quoted as saying:

> The 1999 Web site was a great success, making the major contribution towards the Festival's international profile.

The evolutionary aspects of the 1999 Web site were considered by the 1999 Festival to be successful. For example, the design of the site was created in 1996, and thus the Festival organisers were very pleased to win awards in 1999. The rest of this section details some of the techniques applied to the design of the 1999 site and some of the interesting issues that arose during the life-cycle of the Web site.

When given the original brief by the Festival Director, the development team were told that there would only be three sections in the 1999 Web site: City, Architecture and Design. The author, using previous experience of the evolution of the HMAG Web presence, knew that this would probably change, and designed the 1999 Web site initially with the three sections identified in the brief but with scope for the easy addition of four other major sections. There arose a great deal of conflict between the technical and creative design developers over the issue of designing the site to evolve. A creative designer is a developer role responsible for aesthetic issues in a development team. Primarily the creative designers were apprehensive about building a navigation structure that was flexible enough to be able to double the number of navigation icons. These initial concerns were focused around the apparent feeling of 'emptiness' observed from some of the initial prototypes. However, after much discussion and compromise, a design using frames finally satisfied both camps. The design, see Figure 2, used three HTML pages contained within frames. The top frame contained the major sections in the Festival, initially just City, Architecture and Design, in addition to a link to the 1999 Web site homepage. The initial left-hand frame, like the top frame was a navigation aid, and was used to record the end-user's navigation path within each individual section of the site. The end-user role represents the users of a proposed solution. The frame on the right-hand side contained all the Festival information to be presented.

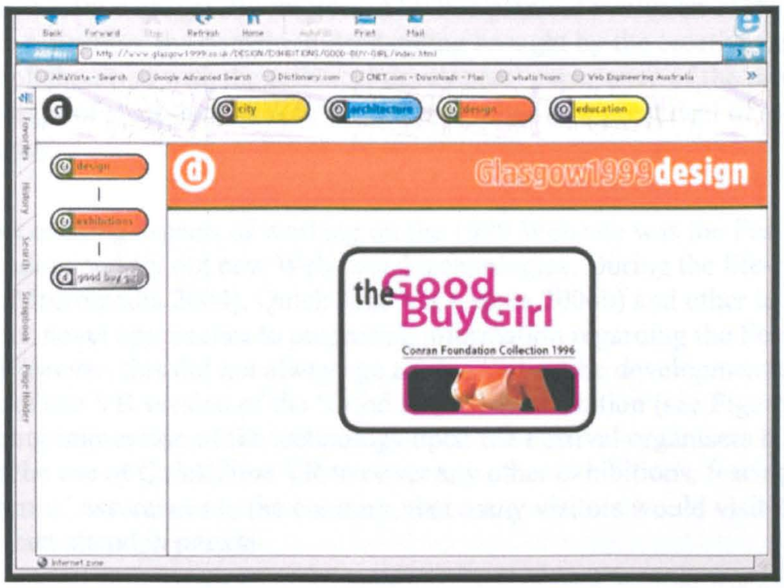


Figure 2. Glasgow 1999: UK City of Architecture and Design Festival Web site - The 'Good Buy Girl' exhibition

The separation of the information within the 1999 Web site into major sections followed closely the business model adopted by the 1999 Festival. Each of the Festival's major strands (City, Architecture and Design) were quite separate and this mapped very well to the 1999 Web site design.

The site was launched in September 1996, and the following spring the Education Director was appointed. He immediately wanted an education section detailing all his plans for the next two and a half years. Given his limited budget of 400 pounds (sterling), the development team were able to employ two undergraduate students to build the education section over a period of one week, during the Easter break in 1997. At the end of the week, the education section contained more information than any of the other three sections.

In addition to the inclusion of the education section, a few months later in the summer of 1997, the new typeface and corporate identity had to be incorporated. This resulted in a complete overhaul of the form of the 1999 Web site presence. However, as the development team had foreseen this, and had designed the 1999 Web site to evolve to include more sections, the functionality of the existing elements of the 1999 Web site remained the same. Primarily this was achieved by long consultation between the technical developers and the creative design developers. The creative design developers were extremely confident about the direction the typeface and corporate identity would follow, basing this judgment on previous work from the design firm commissioned to create the items. The author, as part of the technical development team, was keen to ensure the greatest consistency between the initial form of the Web presence and final form with the new typeface and corporate identity. To the credit of the creative developers, they chose a font and colour scheme that matched closely those of 1999. The new identity was incorporated into the site over a three-day period by one of the creative designers. The entire development team was impressed when it became apparent that there was no need to restructure a lot of HTML. All the graphics using the new corporate identity remained the same size. In fact, the only HTML that changed was the link, active link and visited link tags to match the new colour scheme. This was trivially achieved using a Perl script. When think aloud (Ericsson & Simon 1984, Lewis & Rieman 1994) evaluations were carried out with a disparate sample of end-users, the development team observed no obvious disruption to experienced end-users' use of the evolved site, and found novice and intermittent end-user experience to be comparable to the previous design. The development team were greatly impressed by this painless evolution of the Web presence. Upon reflection, the domain understanding brought by the creative designers, when used in collaboration with the technical developers' experience of the problems of evolving the design of a substantial Web presence, resulted in a great deal of saved time, energy and money.

One of the most exciting aspects of working on the 1999 Web site was the Festival's willingness to adopt and try out new Web-based technologies. During the life-cycle of the project, Flash (Macromedia 2004), QuickTime VR (Apple 2004b) and other technologies were adopted for novel approaches to presenting information regarding the Festival and its many events. However, this did not always go to plan. When the development team created an online QuickTime VR version of the 'Good Buy Girl' exhibition (see Figure 2), such was the overwhelming impression of the technology upon the Festival organisers that they decided to ban the use of QuickTime VR to cover any other exhibitions, fearing, despite the development teams' assurances to the contrary, that many visitors would visit the online version rather than attend in person.

It was at this point that the author and the development team became more aware of the need for closer cooperation with the business and domain experts within the 1999 Festival. The domain expert is a role that provides data or content for Web applications. In addition to providing content, the business expert role is expected to provide guidance on achieving the business objectives of the Web application and is more involved in contributing to the overall structure of the Web presence. This opinion was reinforced by a comment from the Festival Director who stated that the 1999 Web site was not high on the list of mechanisms for communicating information regarding the festival. To his surprise, during 1999, the Web presence was receiving requests from ten times as many unique IP addresses as phone calls through the events hotline. The telephone hotline received approximately 200 telephone enquiries per week during 1999. However, on an average week during 1999, the 1999 Web server sent HTTP responses to approximately 2000 unique IP addresses.

After the 1999 Festival had finished it became apparent to the author that there was a clear need for a different type of development process that explicitly supported the multidisciplinary nature of Web application development. Unlike traditional software development, where clients and end-users are often at a distance during many aspects of a system's development, it was very hard to get anything done on a day-to-day basis without close collaboration amongst developers who can be classified as business experts, domain experts and creative designers. Often the technical developers had to educate the other non-technical developers regarding areas considered in their domain of expertise. In addition, many occasions arose when non-technical developers were educating the technical developers regarding their areas of expertise.

There have been a number of developments to try and tackle problems associated with documenting and modelling aspects of Web development projects (Conallen 1999) and (Ward & Kroll 1999). The author's experience gained during the development of the HMAG and 1999 Web sites indicated that there is a need for a fully specified Web development process, an equivalent of the traditional software engineering process, but taking account of the special characteristics of Web development. The author's experience is supported by other researchers (Murugesan et al. 1999) who highlight the need for "process and product models" specific to Web engineering. Murugesan et al. (1999) also identify a number of other areas requiring research in Web engineering. These areas include, but are not limited to: "requirements analysis and system design", "information modelling" and the "testing verification and validation" of Web-based solutions. While there are many areas identified that require research in Web engineering, each new development will benefit from a process that will encompass and guide developers who embrace new approaches to the development of Web-based systems.

Experience gained on the HMAG and 1999 Web site indicates that a Web development process must explicitly address the issues raised about the evolution of Web sites. However, in order to strengthen the author's observations, based on personal experience, further investigation was required to find out whether the problems that the author observed were also occurring on other Web-based application development projects. The next section (3.3) describes the first published survey of Web engineering in practice.

3.3 The First Survey of Web Engineering in Practice

To gain a better understanding of the challenges facing Web Engineering, the author approached nine independent organisations involved in Web-based development to take part in a survey. Each organisation had Web-based development operations in Scotland or the North of England. Each organisation was placed into one of three categories: *contractors*, *outsourcers*, or *in-house*. Contractors are vendors who service Web engineering projects that are put out to tender by organisations who are classified as outsourcers. The in-house category describes organisations that primarily develop their Web applications within their organisation. It was decided that the three categories of contractors, outsourcers, or in-house would suffice as a basic classification for organisations who are involved in Web development.

The survey was conducted in a qualitative manner using an in-depth one-to-one interview technique. All the answers were recorded on paper by the interviewer conducting the survey. As access to potentially commercially sensitive information may be revealed it was decided to present the results of the interviews anonymously, by individual and organisation. Five of the organisations that participated can be classified as contractors. One organisation that participated was classified as an outsourcer, which puts out to tender millions of pounds worth of contracts for building Web applications annually. In addition, three organisations that, for one reason or another, primarily build their Web sites in-house, were also approached. Of the three in-house organisations, only one was willing to participate. Despite personally knowing people involved in Web development within the two organisations that declined to participate, and providing them with advanced copies of the questionnaire, no official reasons were given for their refusal.

Seven participant organisations and fifteen interviewees were involved in the survey. Where possible within each organisation more than one type of developer was interviewed. Each different type of developer was classified under one of the following roles: software engineer, creative designer, team manager, business expert or domain expert. See section 6.5 for a detailed definition and discussion of these roles. It was felt that this basic classification of developer roles would enable classification of all interviewees. Each of these five basic Web developer classifications can be subdivided into a number of other roles. For example, database administrators, Web masters and programmers would all be classified under software engineer. Eleven of the interviewees worked for organisations categorised as contractors, two of the interviewees worked for one outsourcing organisation and two interviewees worked for one in-house organisation.

While there were questions that were appropriate to all organisations involved in Web development, it was felt that a deeper insight would be gained by tailoring small parts of each questionnaire to suit each different category of Web development organisation. Therefore, each category of organisation, contractor, outsourcer or in-house had additional questions added that were specific to the category of organisation involved. The questionnaires had twenty one questions that all interviewees were asked. The contractor questionnaire had one additional question, the outsourcing questionnaire had five additional questions and the in-house questionnaire had three additional questions. The questions and corresponding recorded answers for contractors, outsourcer and in-house organisations are listed in Appendices 1, 2 and 3 respectively. The questions that were common across all three categories are presented in italics with the questions specific to each category presented in bold, see appendices 1, 2 and 3.

The results from the survey are broken into three sections. Section 3.3.1 describes the type of people, and the structure of the teams involved in Web-based development. Section 3.3.2 focuses on describing the characteristics of Web engineering projects. Section 3.3.3 addresses the features common to the Web engineering processes being used in industry, their shortcomings and their perceived advantages.

3.3.1 Web Development Team Demographics

The survey highlighted an inconsistency between organisations with respect to the titles being given to Web-based developers. However, the survey also illuminated a number of similarities in the tasks developers from different organisations were responsible for during the life-cycle of a Web development project. For example, two of the interviewees belonging to different organisations in the contracting category had the job titles Producer and Senior New Media Designer. These titles conjure up radically different perceptions of what responsibilities would accompany such roles, however, when each interviewee was asked to identify their responsibilities within their respective Web development teams their answers were almost identical. A similar scenario repeated itself with two developers, again from the contracting section, who had the titles Head of Production and Programming Team Leader, and again had almost identical responsibilities within their respective Web development teams.

According to the results, the average age of a Web-based developer was observed to be twenty six years and 70% of the teams involved in this survey were male. One of the major differences between traditional software development and Web-based development is seen to be the small size of Web development teams (Reifer 2000). This survey indicates that the average size of a Web-based team is approximately six[1] developers.

It has also been noted that one of the other differences between traditional software development and Web-based development is the nature of the project deliverables (Pressman et al. 1998) and (Pressman 2000a). In traditional software development often what is delivered comprises a series of related software components with supporting artefacts, whereas in Web application development, the majority of delivered solutions have a large bespoke component, with the project deliverables comprising software components and data that are inter-dependent. The creation, integration and delivery of data is just as important as the design, construction and delivery of software components in Web engineering. With this increased emphasis on the data, the developers responsible for the creation of data are classified into one of two Web developer categories, business and domain experts. It should be noted that the majority of business and domain experts reside out with contracting organisations, and are most frequently found in different departments to the core Web development team, within in-house organisations.

The multidisciplinary nature or wider diversity of the types of developers within Web development teams is another factor that distinguishes Web development from traditional software development. The wider diversity of disciplines involved in Web development has also been noted in the literature by (Pressman et al. 1998), (Deshpande, Murugesan & Hansen 2001) and (Deshpande & Hansen 2001). Hansen, Deshpande and Murugesan (2001) present a definition of the different types of developer roles involved in Web engineering. Riefer (2000) presents a figure of 3-5 developers for the size of a typical Web development team based on empirical evidence working with over forty Web-based projects. However, there is little empirical evidence in the published literature that details the different types of developers that are present within commercial Web development teams, and in what quantity they are represented. One of the goals of this survey was to attempt to identify what different types of developer comprise Web development teams, and in what quantity the developers are represented. This information was gathered by asking each interviewee to classify every member of a standard Web development team in their organisation under one of the following six categories: software engineer, creative designer, team manager, business expert, domain expert or other. The answers show that given a Web development team with eight members[2] , two will be contributing technical skills, two will be contributing creative design skills, two will manage the team, one will provide domain specific business or market advice and one will be providing domain information.

With respect to the interviewees' experience in Web application development, eight of the interviewees questioned had been involved in Web application development for less than three years, the other seven interviewees had been involved in Web development for less than six years. It is also worth mentioning that five of the Web development teams involved

[1] Note the interviewees in the outsourcing organisation were unable to put an accurate figure on the average size of a Web development team. They instead gave figures for the number of people under their respective control who are involved in Web development. The outsourcers' answers were therefore ignored in this calculation.

[2] The average size of a Web development team from the answers given to the breakdown was two developers higher than the average size of a Web development team. This was due to two factors. Firstly, we rounded the answers to the nearest whole developer. Secondly, one of the interviewees broke down the developers in an average team using one of the organisation's current projects, that totalled seventeen developers, rather than using the figure of six developers previously given for the average size of a Web development team.

in the survey had been in operation for less than three years, the other two being involved in Web application development for less than six years.

3.3.2 Characteristics of Web Development Projects

Many consider the biggest hurdle to the successful adoption of traditional software engineering approaches to Web development to be the short development life-cycle time, or time-to-market pressures that are associated with the development of most Web applications (Pressman 2000a) and (Reifer 2000). Indeed, in the experience of the interviewees no Web applications had a development life-cycle time longer than six months, with the average development life-cycle time being just under three months.

One of the objectives of this survey was to gather information regarding Web-based projects that run over budget and/or over predicted time scales. A number of practitioners in the field of cost and effort estimation have highlighted the difficulties in applying traditional software cost and effort estimation techniques to Web-based development projects (Mendes 2000), (Mendes & Counsell 2000) and (Reifer 2000). In this survey the problems associated with cost and effort estimation were addressed by asking every interviewee the following questions:

1. How often do your Web development projects run over time?

2. What are the reasons for projects running over time?

3. How often do your Web development projects run over budget?

4. What are the reasons for Web engineering projects running over budget?

The formal answers given to questions 1 and 3 did not correspond with the informal discussions held with members of the respective organisations in this survey. The author perceived that the interviewees were inclined to give political answers to questions 1 and 3. In addition, the answers often differed between interviewees within the same organisation and were generally too widely spread to draw any statistical conclusions. One of the prime reasons for going over budget on a software or Web project is due to a failure to deliver on time. Scepticism regarding the accuracy of the answers to questions 1 and 3 were supported by a number of interesting anomalies, discovered when comparing answers by the same interviewees to questions 1 and 3. For example, one interviewee stated that approximately 62% of the projects that they have worked on run over predicted time efforts, and then stated that only 5% of projects run over budget. Another interviewee stated that 25% of projects run over budget but that 0% of projects run over time. Further, another interviewee stated the opposite to this, that 25% of projects ran over time, but that 0% of projects ran over budget.

Questions 2 and 4 showed a lot of consistency in the interviewee answers[3]. The interviewees in the contracting organisations, mentioned reasons for Web engineering projects running over time to be because of:

- poor communication between themselves and their clients,

[3] Eleven out of the fifteen interviewees answered question 2. Nine out of the fifteen interviewees answered question 4.

- late project changes by their clients,
- or poor understanding of the process of building a Web application on behalf of the client.

These answers indicate a problem with the requirements and analysis phases of most contractor's Web engineering processes. In addition when discussing reasons for projects running over time, the outsourcing organisation also acknowledged problems in capturing requirements and both in-house interviewees mentioned problems in controlling requirements creep. Other reasons given for projects not delivering on time include poor project management on behalf of the contractors, poor project effort estimation techniques, and inadequate testing procedures. The primary reason given for projects running over budget was also down to problems with the requirements phase of the Web engineering process. Other reasons given for projects running over budget include lack of resources, poor delivery of data and content, poor management, lack of professionalism and unforeseen costs. Strangely no one mentioned failure to deliver on time as a reason for projects running over budget.

The interviewees in both the in-house and contracting organisations were asked if they ever found themselves short of development expertise during the development of a Web application. All interviewees answered the question[4]. Two interviewees mentioned a lack of business experts, two mentioned a lack of domain experts, seven mentioned a lack of technical expertise, one mentioned a lack of creative design skills and one interviewee mentioned a lack of management skills. The interviewees were asked the following two questions, regarding the success rate of Web application development and the reasons for project failure.

5. How many proposed Web projects result in a delivered system?

6. What are the prime reasons for projects not resulting in a delivered Web system?

Question 5 suffered similar problems to questions 1 and 3 previously. The formal answers to both questions did not match the informal discussions with many of the members of the organisations interviewed. Also, like questions 1 and 3, the answers varied too much in order to draw any statistical conclusions. However, like questions 2 and 4, question 6 was very illuminating, with six interviewees claiming lack of budget and four interviewees claiming problems with the analysis and requirements phases in their Web engineering process as a reason for project failure. The answers to question 6 further strengthen the claim that Web engineering processes being used in commerce need to focus more on analysis and requirements phases if they are going to increase the rate of successful projects.

3.3.3 Web Engineering Processes in Practice

Practitioners and researchers in the field of Web engineering and software engineering have commented on the lack of suitable software engineering processes that can be used to build Web applications (Pressman et al. 1998), (Pressman 2000a) and (Reifer 2000). In order to investigate the way commercial Web engineering is being carried out each interviewee was

[4] The two outsourcers were not asked this question as it was felt that they would not be outsourcing the development of their Web applications if they had the skills in-house.

asked the following question relating to their organisation's Web application development process.

> 7. Do you have a well-defined and documented development process for building Web-based projects?

Seven of the fifteen interviewees claimed to have a development process in place for building Web applications. However, of the seven interviewees who answered yes to this question, only two, who both belonged to the in-house organisation, were actually using an industry standard software development process for building Web applications. The other six organisations were using a development process that had been created in house. The industry standard software development process in question, used by the in-house organisation, is Dynamic Systems Development Method (DSDM 2004a). DSDM is a Rapid Application Development (RAD) approach that focuses on delivering solutions in short time scales and professes close integration with business experts and domain experts, referred to in DSDM as 'Users'. DSDM is independent of any software engineering tools and techniques and is a high-level approach to building software systems. A more detailed discussion of DSDM and its suitability to Web engineering is presented in Chapter 5.

When asked to describe the process of building a Web application all the interviewees' processes started with the development of a project scoping document, which covered the requirements and design phases of a Web engineering project. The Scoping document was one of only two deliverables that were created in the vast majority of Web engineering processes used by the organisations in this survey, the Web application itself being the second deliverable.

If one considers the major stages in a software development process, regardless of what order or how often one wishes to address these phases, to be Analysis, Requirements Capture, Design, Implementation, Testing (verification), Evaluation (validation), Training and Maintenance, then one would hope that every Web engineering process would at least try to address each of these stages at some point in the development life-cycle. Sadly though, the majority of the development processes seem to focus on Implementation with Requirements Capture and Design being carried out as one combined phase at the beginning of the project life-cycle, and Testing being carried out in parallel with Implementation, if at all. Only two interviewees mentioned an Analysis stage, and no interviewees mentioned Evaluation, Training or Maintenance stages in their Web development process. The number of problems highlighted by the interviewees with respect to the requirements definition and capture appeared to be a result of combining the design phase with the requirements phase. Often the development team members are making low-level decisions regarding how they will realise the design of a system before they understand what goals or problems the system should be addressing. This suggests that Web engineering processes need to focus more on analysis of the specific problem(s) being addressed by the Web-engineering project in question.

When asked what features of their Web development process interviewees considered successful, three mentioned the Scoping document, four mentioned good relations with clients and four mentioned good management. When asked what features of the process interviewees considered problematic, nine mentioned problems with managing requirements or poor communication mechanisms with their clients, three mentioned problems with managing the development process, two mentioned poor testing procedures and one mentioned poor documentation.

At the end of the questionnaire all interviewees were asked the following two questions.

8. What mechanisms do you employ to measure the success of a project?

9. At what stage in the development process do you employ these mechanisms?

Ten of the interviewees considered a project to be successful primarily if the client was happy with the deliverable, while seven mentioned achieving budget and time estimates as a mechanism for measuring a project's success. Only three interviewees mentioned passing a testing phase as a mechanism for measuring project success and no interviewee mentioned involving end-users as a mechanism for validating the success of a project. Since the majority of interviewees were using a prototyping approach during the development of their Web applications, it was alarming to hear developers mention clients rather than end-users as a mechanism for validating the project deliverables. Ramsay and Nielsen (2000a, p. 4) point out one of the main problems currently facing Web engineering projects is developers striving to satisfy their client's desires rather than their client's needs. All the interviewees measured success at the end of a project's life-cycle, or after the project had ended. The interviewees discussed no mechanisms to properly *evaluate* or *validate* the project deliverables during development.

One of the most disturbing issues facing Web engineering is the poor attention paid to analysis, evaluation, training and maintenance in commercial Web engineering processes. Without proper analysis of the problems or challenges to be addressed, development teams will find it more difficult to know if they are building the right product. The lack of an evaluation stage leaves developers no mechanisms to properly validate the deliverables, and therefore no mechanisms for checking to see if they have built the right product. If one considers the simple hypertext paradigm of the Web then it is easy to ignore training; however, if one does not require training to use a software system this does not imply that there is no need to train the people responsible for maintaining the system. Indeed the complete lack of attention paid to maintenance issues in general during Web engineering processes must reduce the quality and longevity of the systems being developed, and enforce the growing opinion of many that the Web will be where most of the legacy problems of the future will be found (Tilley 2000).

Each interviewee was asked to identify which tools and technologies they used to build their Web applications and at what stage in the development process these tools were implemented. The results showed little consistency between or within organisations. Often there were many competing tools and technologies being used within an organisation with no clear rationale, other than developer preference, for the selection of one solution over another. All that could be concluded with confidence from the answers to the questions relating to tools and technologies was that Microsoft's Outlook Express (Microsoft 2004b) was by far the most popular email client across all major operating system platforms and that Adobe Photoshop (Adobe 2004) was the most popular raster image manipulation tool.

As was shown previously, two of the major differences between software development and Web-based development are the multidisciplinary nature and smaller size of teams. Consider a large-scale software development project (100 developers, or more) and a large Web engineering project (100 developers, or more), it would not be uncommon for the overall management of developers to be similar in structure across both projects. In both cases developers would probably be managed in small teams, however this is where the similarity ends. In software engineering, the teams are broken down into smaller units who will address different problems and tasks. Consider a large software engineering project, with the goal of building a new operating system. In such a project, teams could be broken down as follows, one team responsible for the development of the kernel, one team responsible for the windowing system, another responsible for standard device drivers, and so forth. On a large Web engineering project teams will often be broken down into multidisciplinary groups, who will build different sections of the Web application, but in general will produce very similar deliverables and work on very similar problems. In the course of developing a large software system each small team will have to interact with other teams, this is normally done through predefined interfaces, each team mostly viewing other team's deliverables as black boxes. In Web engineering, not only do teams have to communicate information regarding their deliverables through predefined interfaces, but in order to reduce duplication of effort and ensure consistency, good communication amongst similar types of developer in different teams is just as essential as good communication within teams and between different teams.

The author's personal experience of developing Web applications in both commerce and academia has highlighted the conflicts that often arise among different types of disciplines within a Web development team. To try to gain a better understanding of these conflicts every interviewee was asked the following two questions.

> 10. What types of conflict arise between different developers in a Web development project?
>
> 11. How are conflicting views resolved between different developers in a project?

According to the answers given to question 10, each different type of developer seems to poorly appreciate the contributions made by every other type of developer. Sadly, according to the answers to question 11, there seems to be little policy in place for resolving conflicts. Only one organisation mentioned using the Scoping document to resolve conflicts with the best interests of the project at hand.

A number of members of the Web engineering community have commented on the lack of a suitable Web engineering process for the development of Web applications (Murugesan et al. 2001) and (Pressman 2000a). At present most of the commercial focus in Web engineering surrounds tools that aid and assist the implementation of Web-based systems. Yet as this survey has shown, major process problems exist in the Analysis, Requirements, Testing (verification), Evaluation (validation) and Maintenance stages of Web engineering processes.

3.4 The Criteria for a Web Engineering Process

Sections 3.1, 3.2 and 3.3 in this chapter have presented empirical evidence that highlights the challenges that a successful Web engineering process needs to address. This evidence takes the form of the author's experience of building two large Web applications and presents the findings of the first survey of Web engineering in practice. In summarising the major themes identified in sections 3.1-3.3, a Web engineering process, if it is to be successful, needs to explicitly address the following:

1. **Short development life-cycle times.** According to the results gathered in section 3.3, the average Web development life-cycle time is less than three months. This figure is dramatically lower than that of traditional software development (Reifer 2000, p. 58). If a Web engineering process is going to be successful, then it has to cope with exceptional time pressures.

2. **Delivery of bespoke solutions and different business models** (hereafter referred to as **Different Business Models**). Unlike traditional software engineering, Web engineering must cope not only with the development of software components but also with the development of data, and the inter-dependencies between both. In addition, the new communication mechanisms introduced as part of Web engineering initiatives often require a significant degree of business process change, evolution or re-engineering.

3. **Multidisciplinary development teams.** A Web engineering process must take into account the different types of developer required to build a successful solution. Ensuring that all those involved understand their roles and responsibilities and where overlap occurs how to resolve conflict in the best interests of the project in question.

4. **Small development teams working in parallel on similar tasks**. Like traditional software development, large numbers of Web developers are split into smaller teams. Ideally, each team should provide an interface to the deliverables that they produce, enabling other teams to view the deliverable as a black box. However, in large Web engineering projects the teams are working in a shared solution space, which is difficult to separate into independent deliverables. Therefore a Web engineering process should also allow different types of developer to communicate amongst their peers, beyond their immediate project team. This will help ensure consistency and will help to reduce duplication of effort amongst teams.

5. **Business Analysis and Evaluation with End-Users** (hereafter referred to as **Analysis and Evaluation**). There is a need for greater focus on Analysis and Evaluation stages in Web engineering processes. The survey in section 3.3 shows that there is little attention paid to either of these stages in most commercial Web engineering processes. A Web engineering process should encourage developers to address the following questions:

    a. Should we develop a Web application?

    b. Why are we going to develop a Web application?

    c. What problems or goals will the Web application address?

    d. How will we know if the solution addresses these problems or goals?

    e. How will we measure and validate the deliverables?

6. **Requirements Capture and Rigorous Testing** (hereafter referred to as **Requirements and Testing**). In addition to understanding what issues the Web-based solution should address, and how one can measure the success of the deliverables in tackling these issues, a Web engineering process needs to focus more on Requirements and Testing. A Web engineering process should strongly encourage its users to ask the following questions:

    a. Are we building the product right?

    b. How will we ensure that we have built the product right?

7. **Maintenance (Evolution) of Web-based Applications** (hereafter referred to as **Maintenance**). If the longevity and quality of Web-based solutions are to be improved, then more attention needs to be paid to the issue of Maintenance. While a well-documented system may not be essential during the development of a Web application, it is certainly necessary for ensuring the proper maintenance and update of the deliverables.

Not only must the process address points 1-7 above, but the process will also benefit from independence from specific tools, techniques and technologies. This is not to say that supporting tools and techniques are not important in Web engineering. Rather, such is the diversity and rapid change of technologies used to build Web applications, that a Web engineering process should remain as distant as possible from the mechanisms used to implement Web-based applications.

3.5 Other Surveys of Web Engineering in Practice

Since 2001, more literature involving surveys of Web and multimedia development practice has been published. These subsequent surveys present further evidence to support the criteria for a Web Engineering process in section 3.4, points 1-7. For example, Barry and Lang (2001) carried out a survey throughout Ireland and concluded that no uniform approach existed for conducting multimedia systems development and that there is a requirement for new techniques for capturing systems and integrating them within the development framework. Barry and Lang also note the need for a design process with greater focus on end-users, supporting point 5. Barry and Lang observed that 63% of those using an in-house process rated the speed-to-market strengths of their process as being the second most important benefit derived from the in-house process being used, strengthening point 1. Taylor et al. (2002b) carried out a survey in the North West of England, and noted the problems with the lack of formal testing procedures in development processes, stating that only 25% of those involved in their survey had formal testing procedures, supporting point 6. Taylor et al. (2002b) also note the involvement of non IT staff in creating aspects of the Web deliverable, strengthening point 3. Taylor et al. (2002a) focus on the maintenance issues in Web site development processes stating that only 30% of companies produced documentation, strengthening point 7. Lowe and Eklund (2002) carried out a survey in Australia, observing: shorter time scales, multidisciplinary teams, volatile requirements and inadequate requirements documentation, supporting points 1, 3, 6, and 7 respectively.

The most recent survey of Web engineering in practice by Zhou and Stålhane (2004) is based on a survey of project management and developer employees in eleven Norwegian IT organisations. Zhou and Stålhane presents ten main findings based on the data collected during their survey. These findings provide strong support for the criteria for a Web engineering process, and indicate that previous findings which are geographically limited to Australia (Lowe & Eklund 2002), the United Kingdom (McDonald & Welland 2001a), (McDonald & Welland 2001b), (Taylor et al. 2002a) and (Taylor et al. 2002b) and Ireland (Barry & Lang 2001) exist elsewhere as well. For example, finding 3 "Web-based system projects tend to be short and small" Zhou and Stålhane (2004, p. 366), strengthens point 1. Finding 4 "There is agreement that the development phase (implementation and testing) is the most time-consuming phase and the requirements phase is the least time consuming phase" Zhou and Stålhane (2004, p. 369) supports point 6. Finding 8 "The overall level of use of engineering methods and techniques for reliability and robustness is not high in the Web-based systems development process." Zhou and Stålhane (2004, p. 369) indicates little attention to the long term quality issues that are major drivers in successfully evolving and maintaining a solution, directly supporting point 7.

Zhou and Stålhane (2004) also report a number of other observations about the practice of Web engineering that support the data presented in appendices 1-3. For example:

> 54.5% of respondents stated that their organizations do not have a process model at all; 18.2% stated that they have a process model but seldom apply it in practice; while only 27.3% of respondents stated that they develop Web-based systems with a defined process model. In other words, despite the emphasis in the literature on the importance of the process model, it is little applied in practice.

The above quote clearly indicates the poor suitability of traditional software engineering processes to Web application development and indicates the chaotic and ad-hoc process approaches adopted in commerce for building Web-based systems. At the time of writing, I have not been able to find any survey or other empirical evidence, which suggests that the above criteria (section 3.4) do not characterise the distinctive nature of Web engineering. However, further work will hopefully extend and refine these criteria as Web engineering matures as a discipline.

3.6 Summary

This chapter has presented the criteria for a Web engineering process (section 3.4 points 1-7). The criteria are derived from empirical evidence based on the author's experience of building two large Web applications (sections 3.1 and 3.2) and the first survey of Web engineering in practice (section 3.3). Section 3.5 has reviewed subsequent published surveys of Web engineering in practice identifying further evidence in support of the need for the criteria for a Web engineering process. The next chapter discusses the criteria for a Web engineering process in greater detail.

4 The Criteria for a Web Engineering Process

The previous chapter presented empirical data identifying the major characteristics that describe Web-based application development, and the criteria that a successful Web engineering process will have to address:

1. Short development life-cycle times;
2. Different business models;
3. Multidisciplinary development teams;
4. Small development teams working in parallel on similar tasks;
5. Analysis and Evaluation;
6. Requirements and Testing;
7. Maintenance.

As mentioned in chapter 3, the need for a suitable development process for Web engineering is widely recognised (Murugesan et al. 2001) and (Pressman 2000a). At present most of the commercial and academic focus is on tools and techniques that aid and assist the implementation of Web-based systems. It is important that suitable tools and techniques underpin any development process. However, it is more important that the process is defined and understood first, followed by the activities and techniques to support it, before tools to support the process itself are successfully developed. This chapter describes in detail the above criteria required to be addressed by a Web engineering process and the rationale behind these criteria. To aid the discussion the following broad role categorisation of stakeholders involved in any software development process is used:

- The *client*. The individual, party or body who commissions and pays for the project or software system that will be developed;
- The *developer*. The individuals who work on the project developing and maintaining the software system commissioned by the client;
- The *customer community*. Those individuals who interact directly with the live software system (the end-users) or are affected by the live operation of the software system.

Many software processes evolve a more detailed classification of stakeholders. For example, the stakeholders classified as developer can be broken into a number of different roles including architect, designer, programmer, administrator, tester and manager. However, the above three classifications serve to demonstrate that there are different viewpoints across all development processes.

In the following discussions, three examples of projects within the banking industry are used to illustrate various points. At one extreme is the development of a branch teller system for a bank. Here the majority of the customer's transactions take place through the medium of the teller operating the system in a branch. It is assumed that this was developed using a plan-driven traditional software process approach. This type of project was typical of many carried out in the 1980s and early 1990s in the financial services sector. These systems are often referred to as legacy systems. Hence it will be referred to as the *legacy teller system.*

At the other extreme is the development of a retail Internet banking system, which will allow the bank's customers to directly interact with their accounts via a Web-based system. This type of development is typical of projects carried out in the late 1990s and the early part of this millennium in the financial services sector. This project will be referred to as an *Internet banking system*. The third example is the ***Web-based teller system***. The Web-based teller system describes a project, as part of a larger Intranet development, the scope of which includes replacing the legacy teller system with a Web-based teller system. This type of initiative is common today within the financial services sector. Again, like the legacy teller system, the bank's customer transactions will take place through the medium of the teller operating the Web-based teller system.

4.1 Short Development Life-Cycle Times

Some Web applications have fixed lifetimes, such as conference Web sites, while other Web presences have lifetimes that have no foreseeable date for decommission, such as online banking Web applications. However, regardless of the known lifetime of a Web application, the most important criteria that a Web engineering process must address (Baskerville et al. 2003) is the short development life-cycle time, or time-to-market pressures that are associated with the development of most Web applications. In the experience of all interviewees who took part in the first survey of Web engineering in practice, see section 3.3 for more details, no Web applications had a development life-cycle time longer than six months. The average development life-cycle time of a Web application was observed by the interviewees who took part in the survey to be just under three months. Baskerville et al. (2003) argue that development speed is of primary focus, before cost and quality which are of secondary importance in Internet speed software development. The extreme time-to-market pressures experienced by organisations involved in Web engineering projects are regularly driven by the fierce competition in the global e-marketplace. Thus, if a commercial Web engineering process is to be successful then it must place primary focus on addressing the extreme time-to-market pressures associated with Web engineering projects.

4.2 Different Business Models

Traditional software engineering projects are primarily concerned with the creation of software components with supporting artefacts. These software components and supporting artefacts try to address problems that are either:

- *Generic* across a number of organisations and their various stakeholders. For example, the development of an operating system such as Microsoft's Windows XP (Microsoft 2004a) or tools such as Adobe's Photoshop (Adobe 2004);
- *Bespoke* to one organisation and its stakeholders. For example, the development of a customer relationship management system for a bank, or the building of an Internet based Web presence for a museum.

Generic software components and supporting artefacts are often developed independently of the data upon which they will operate. Web engineering on the other hand results in deliverables of software components and supporting artefacts that are developed in parallel with the creation of the data that they will operate upon and in conjunction with. In essence, Web engineering projects result in bespoke solutions comprising data and software as discussed in chapter 3. Therefore, discussion of software engineering processes forthwith will focus on bespoke systems development.

As evidence presented in chapter 3 indicates, the building of Web applications requires multidisciplinary development teams. The multidisciplinary nature of Web application development is in stark contrast to traditional software development (Deshpande, Murugesan & Hansen 2001, Deshpande & Hansen 2001). A Web engineering process must therefore support the creation of bespoke data and software deliverables and assist multidisciplinary development teams to balance the interdependencies that exist between the data and the software during the Web development life-cycle.

In general there are only three models of any great significance commonly represented within traditional software engineering processes:

- The *software model*. The software model reflects a view of the issues associated with developing a software solution that achieves the business objectives reflected by the views of the business model, and the constraints in applying the business model to the domain model;
- The *business model*. The business model reflects the business objectives and any associated constraints with respect to the project using the process;
- The *domain model*. The domain model reflects views of the domain to which the business objectives and the proposed software solution are to be applied.

For example, consider the development of the legacy teller system. The software model reflects the issues associated with developing the proposed teller system to automate the branch teller activities. The business model would reflect the business objectives of the branch and teller activities, detailing the perceived inefficiencies of the current procedures and the proposed benefits of the new automated system. The domain model would reflect the issues associated with branch and teller activities, detailing working procedures and environmental conditions under which these procedures are carried out.

The primary impact of the traditional software engineering process used to develop the legacy system was in the software model, driven by views from the business and domain models. Rarely do traditional software engineering processes explicitly give feedback into the business and domain models. Figure 3 shows the impact business, domain and software models have upon each other in traditional software engineering processes. The software model is impacted only by a partial section of business and domain models. This impact is exerted usually only once or twice during the development life-cycle in the case of plan-driven processes, with the information that the software model holds regarding the business and domain models primarily being seen to be static. More modern agile processes on the other hand encourage impact from the business and domain model to the software model throughout the development life-cycle. Rarely does the software model have a radical impact on the business and domain models. As a result, many of the software solutions resulting from traditional software engineering processes are largely implementations of existing business practice. Web Engineering on the other hand is complicated by the addition of a creative design model:

- The *creative design model*. The creative design model reflects the issues associated with the aesthetic aspects of the user interface that need to be reflected in a Web engineering process.

There is significant interaction or impact between the software model and the creative design model during Web development. Balancing the conflict between *function and form* is essential. With respect to *function*, it is essential that the Web presence is user friendly. However, it is also very important to ensure that with respect to *form* that the aesthetic presence of the site attracts customers and enforces the corporate identity of the Web-based deliverable.

Web engineering requires the business and domain models to design and develop data, influence Web site structure, and requires them to not only impact and affect change in the software and creative design models, but requires the software model to impact and affect change in the business and domain models. The business and domain models also have to address impact and affect change between them in Web engineering. This is particularly true in e-business initiatives where new business models are being applied to different domains. See Figure 4 for the impact business, domain, creative design and software models have upon one another in Web engineering. Figure 3 and Figure 4 also illustrate dependencies between the business and domain models outside of the engineering process, which represents business initiatives that take place independently of the development of a software or Web-based system.

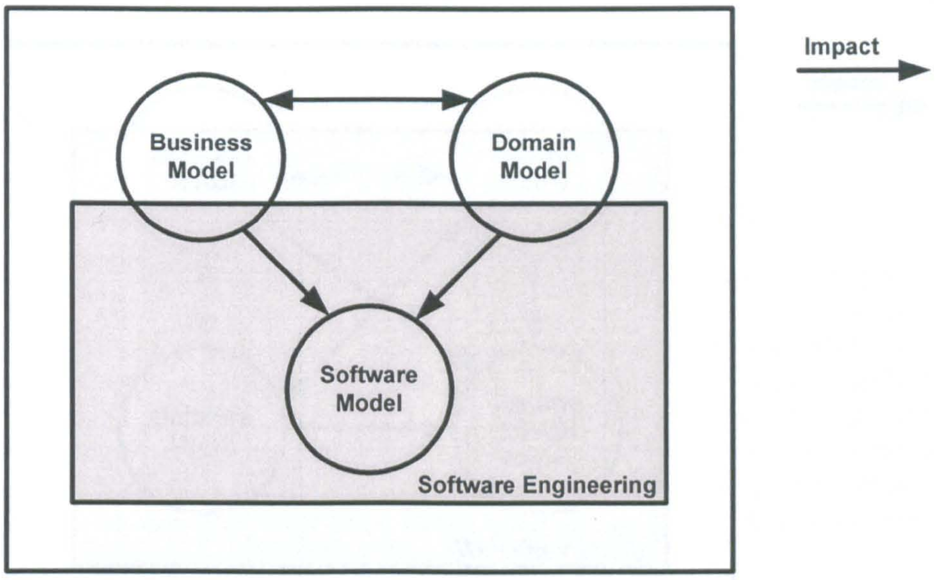


Figure 3. Impact Exerted by the Software, Business and Domain Models Upon Each Other in Traditional Software Engineering Processes

The creative design model has limited impact back upon the business and domain models, and therefore there is no impact reflected back from the creative design model to the business and domain models in Figure 4. It would be extremely irregular for the form or aesthetic aspects of a Web application to drive a change on the business and domain models. It is not that this impact will never occur, but rather that it would be the rare exception rather than rule. For example, the need to evolve the branding of a Web presence may impact changes to the marketing literature associated with the Web application. However, it would be extremely rare for such a creative design driven impact to have any changes on how the business and domain models function, and so changes to the business process would in the vast majority of cases be non-existent.

The development of Web-based applications often requires a degree of re-engineering within the business and domain models. A recent survey by Datamonitor (2002, p. 33) claims that "as a result of eBusiness investment, 52% of companies require changes to job functions and 25% require restructuring of departments and lines of business". Research has been conducted into and on the importance of business process re-engineering during software development activities involving legacy systems environments (Henderson 2002). However, quantifying exactly how critical business process re-engineering activities are to the success of Web engineering projects is an area requiring further research. The author's experience and that of others (Datamonitor 2002) and (Kirkpatrick 2004) suggests that it is often very necessary. Such is the significance of the impact exerted between the four models in Web-application development, that if an organisation wishes to harness this impact to their benefit, then a re-engineering initiative is often required in order to ensure the success of the proposed Web-based system. It is crucial that those organisations and individuals involved in Web-based endeavours understand the impact exerted amongst the models in Web engineering. Indeed many of the 'e-words', such as 'e-Revolution' and 'e-Transformation', associated with Web-based developments indicate the importance of such re-engineering activities. Ultimately, organisations and individuals that do not understand the significance of business process re-engineering and who fail to focus upon it during Web application development, risk the success of their projects and the long term survival of their organisation (Kirkpatrick 2004).

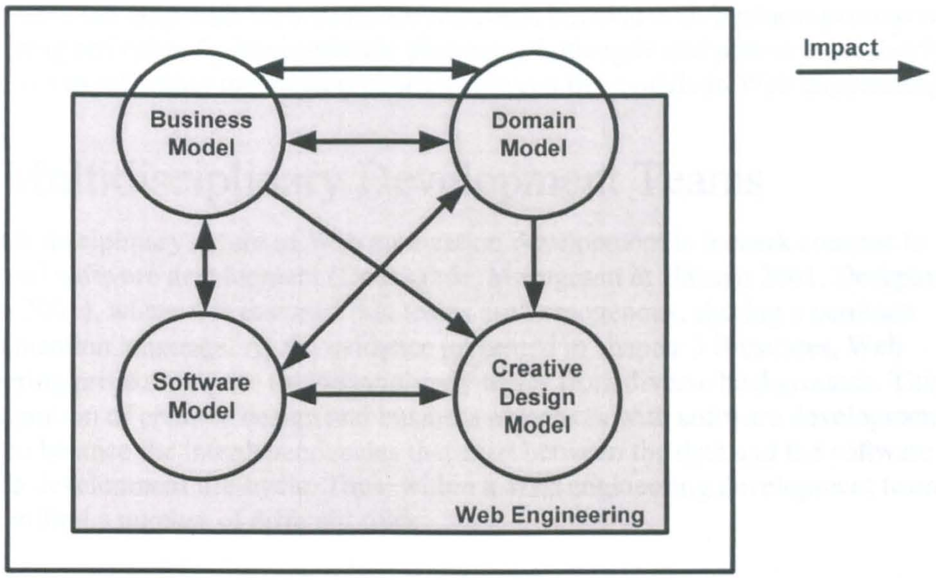


Figure 4. Impact Exerted by the Software, Business, Domain and Creative Design Models Upon Each Other in Web Engineering

This is not to say that the business and domain models can evolve independently. Rather, the contribution that business and domain models have in Web engineering is so substantial and critical to the success of the project, that both should have development models with different views from the software and creative design models reflected in the development process and development team. It is not realistically possible for any one individual stakeholder to understand every view reflected by each of the models in Web engineering. However, it is crucial that every different type of developer, whether they are representing the business, domain, creative design or software model, educate, and collaborate with, those representing other roles within the team. This interaction is critical to achieve the best solution possible, given the project or organisational problems or goals being tackled. Developers are required to understand the impact exerted on their model and to feed this understanding back into re-engineering processes.

Consider the development of the Web-based teller system to replace the legacy teller system. The problem is complicated by a software model that has to have an impact on the business and domain models. For example, the impact of Web-based technologies may render current operational methods redundant and may present new business opportunities, such as centralising business functions currently federated amongst the bank's branches. Therefore, the impact exerted by the Web-based technologies will have a direct impact on the business and domain models. The creative design model, newly introduced, is partially responsible for the issues associated with the user interface design, focusing on the aesthetic aspects of the

Web application. The business and domain models will also have to impact the creative design model. For the first time it may be possible to introduce the corporate identity or branding across all teller applications, allowing for easy end-user distinction between the Web-based Teller System and other external Web applications. Conflicts will often arise between aesthetic design and usability. The relationship between the software model and the creative design model has a two way impact as discussed previously.

The impact exerted between the software, creative design, business and domain models in Web engineering, see Figure 4, needs to be reflected in a Web engineering process. It is essential that the development team is aware of the various interactions between the models in Web engineering during the development life-cycle, so that they may harness these interactions, not only with Web-based deliverables, but also with business process re-engineering activities. Current software processes, both agile and plan-driven, poorly lend themselves to reflecting the impacts exerted between the models in Web engineering.

4.3 Multidisciplinary Development Teams

The multidisciplinary nature of Web application development is in stark contrast to traditional software development (Deshpande, Murugesan & Hansen 2001, Deshpande & Hansen 2001), where it is assumed that teams are homogenous, sharing a common communication language. As the evidence presented in chapter 3 illustrates, Web engineering projects require multidisciplinary teams from diverse backgrounds. This requires the integration of creative design and business objectives with software development, and an ability to balance the interdependencies that exist between the data and the software during the Web development life-cycle. Thus, within a Web engineering development team one can expect to find a number of different roles:

- *Domain experts* – responsible for issues associated with the application domain of the Web application, including low-level integration and creation of content;
- *Business experts* – provide guidance on achieving the business objectives of the project;
- *Creative designers* – responsible for aesthetic issues;
- *Software engineers* – leading on technical issues and implementation;
- *Team leaders* – providing team guidance and project management.

The diverse nature of a team obviously leads to many different channels of communication among team members, and unlike a conventional software engineering team, there is no common notation for communication. For example, the software engineer and the creative designer have to work closely together to produce a satisfactory solution, balancing function and appearance. One possible communication mechanism is through the shared *browser experience* of the developing system. The browser experience is a similar concept to pair programming in XP (Beck 2000). In pair programming homogeneous developers communicate primarily through the medium of the code itself. Using the shared browser experience as a communication mechanism, heterogeneous developers, i.e. the software engineering and creative designer, can use the browser experience presented to the end-user as a communication channel to help resolve problems and conflicts between the creative design and software models. It is therefore very important that a Web engineering process supports multidisciplinary teams, given the heterogeneous nature of Web development teams.

4.4 Small Development Teams Working in Parallel on Similar Tasks

Large software engineering projects, such as the development of the legacy teller system, regularly split the developers into smaller teams, with each team being challenged with a different type of problem space. For example, one team may be responsible for the development of reports for senior management, one team for the user interface, one team for integrating into desktop devices such as printers, or card readers. Each team may be using similar technologies and techniques to build their respective deliverables, viewing every other team's deliverables primarily as a black box, through predefined interfaces. In essence, each team will be working on a different type of problem space.

In large Web engineering projects, developers are often organised into a number of smaller teams, where deliverables are viewed as black boxes through pre-defined interfaces, just like large software engineering projects. However, each team will be working on more similar types of problem spaces. For example, a large number of separate teams on the same Intranet presence, of which the Web-based teller system is a sub-component, will be concerned with Web-based sign-on, and will want to leverage the same development effort across multiple teams to achieve Web-based single sign-on. The objective is to reduce development effort and cost, and to enhance the end-users' experience across multiple Intranet teams' deliverables within the Intranet presence. It is essential that a Web engineering process supports many small teams working on the same Web presence, thus, helping to ensure that there is no 'reinvention of the wheel'. This will assist with preventing duplication of development effort, decrease development costs and will help to ensure consistency across the deliverables. This should help produce a better end-user experience and ease the maintenance and evolution of the project deliverables.

4.5 Analysis and Evaluation

The first survey of Web engineering in practice, see section 3.3, showed that analysis of the problems Web-based projects are trying to address and the validation of the deliverables being produced to address these problems seemed to be getting ignored in the vast majority of Web projects. It is naive to think that a Web development team can solve problems if it does not clearly understand them. The development team will have little chance of measuring the success of the deliverables (validation) if they have little or no understanding of problems to be addressed. With respect to validation or evaluation of the deliverables, the importance of understanding end-user usage is critical to the success of Web engineering projects. The ease with which end-users can find and change to alternative Web-based solutions should they lack satisfaction with their current Web experience is a stark warning to those who ignore usability during Web development. Nielsen (2001b) and Spool et al. (2002) have both reported on the critical nature of usability in the success of Web application development for e-business. In addition Nielsen (2002) has also reported on the criticality of usability with respect to Intranet and Extranet development. If one is to build successful Web applications then the Web development process must place great focus on understanding end-user usage of the proposed solution.

Figure 5 depicts the customer community view that is reflected in processes used for traditional software development. The end-users (primary impact), those who will interact with the software solution directly, are a sub-set of the company or organisation's employees. In the case of a legacy teller system for a bank, the end-users would be the bank's branch staff, a sub-set of the bank's entire employee population. Although addressing usability issues during development is important in any category of software development (Constantine & Lockwood 1999), there are other mechanisms available to companies and organisations to enable them to address usability issues indirectly, primarily through mandatory operational procedures and training for end-users. Those who may be impacted indirectly (secondary impact) by the introduction of the proposed solution are made up of employees who do not directly interact with the system and the organisation's business customers. Using the example of the legacy teller system these would be retail and small business customers who enter the branches to carry out their banking functions. There may also be a small proportion of head office and IT operation staff involved, this number may be small as most of the operational processing, data input, approval procedures and credit checking, can be carried out in the bank's branches.

Figure 6 shows the customer community view that needs to be reflected in processes used for Internet Web-based development. The target end-users (primary impact), those who will interact with the Internet Web-based solution directly, are generally a large proportion of the company or organisation's business customers. In a retail banking context these would be the bank's small business and retail customer population, primarily a business-to-customer (B2C) Web banking channel. In addition to the bank's business customers there will be a number of other internally facing Web-based interfaces to this channel, for example help desk staff, IT operational staff and centralised head office staff. These employees will use the Web-based interfaces as this will introduce new ways of working to provide banking features not available through branch channels, for example, online direct processing of direct debits, standing orders and currency exchange. These new banking features are generally processed centrally as opposed to within the branch to create savings in operational business costs.

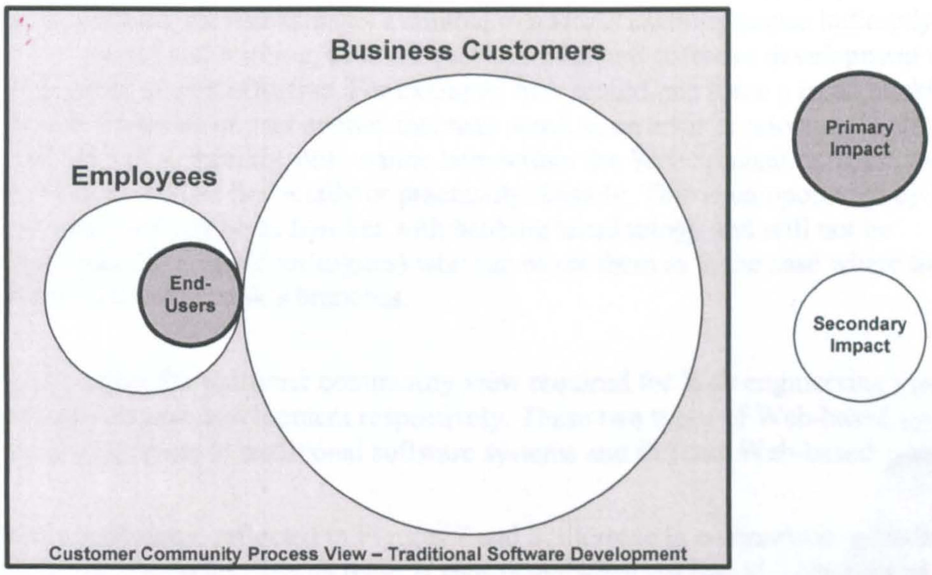

Business Customers
Employees
End-Users
Primary Impact
Secondary Impact
Customer Community Process View – Traditional Software Development

Figure 5. The Conventional Customer Community View in Processes for Traditional Software Development

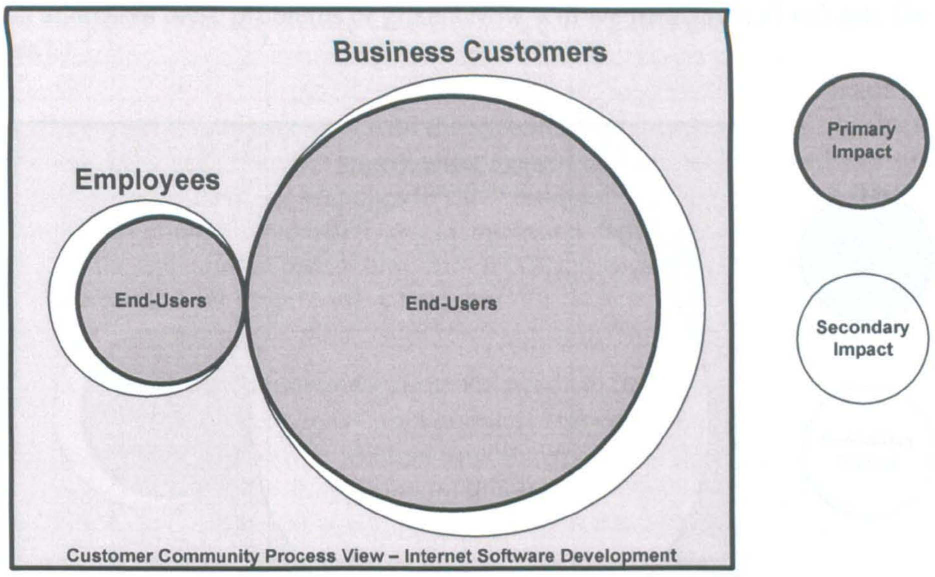


Figure 6. The Required Customer Community View in Processes for Internet Web-based Development

Figure 6 when compared to Figure 5 highlights the requirement for increased focus on usability in Web development processes. Particularly as the end-user population is normally many tens of times larger than would be found in a traditional software development population. In addition, the mechanisms available to address usability issues indirectly, i.e. mandatory procedures and training, as is the case in traditional software development are not normally applicable or cost effective. For example, how would one force a retail banking customer to avoid a series of user actions that may result in an error or anomaly in the system? With respect to training, only online help within the Web application itself, and a telephone help desk will be financially or practically feasible. This is compounded by the fact that end-users will not be as familiar with banking terminology and will not be surrounded by peers (e.g. bank colleagues) who can assist them as is the case where the end-users are bank staff in the bank's branches.

Figures 7 and 8 show the customer community view required for Web engineering processes for Intranet and Extranet development respectively. These two types of Web-based systems fall between the extremes of traditional software systems and Internet Web-based systems.

The end-user populations, reflected in Figures 7 and 8, increase in comparison to traditional software development. While this increase is significant when compared to traditional software development, it is acknowledged that the end-users will use the system more frequently and it will be more feasible to employ indirect methods, e.g. training, to address usability issues. However, there is a need for increased focus on the end-users and their needs in such systems.

If a Web engineering process is to be successful then it must explicitly acknowledge analysis and evaluation during the development of the proposed system and encourage those adopting it to focus more on these areas. A Web engineering process needs primary focus on evaluation, helping the development team optimise their deliverables for usability and the end-user usage of the proposed system. There is a need for this focus whether the project is Internet, Intranet or Extranet. In essence a Web engineering process should encourage those using the process to ask the following questions. Why are we going to develop a Web application? What problems or goals will the Web application address? How will we know if the solution addresses these problems or goals? How will we measure and validate the deliverables?

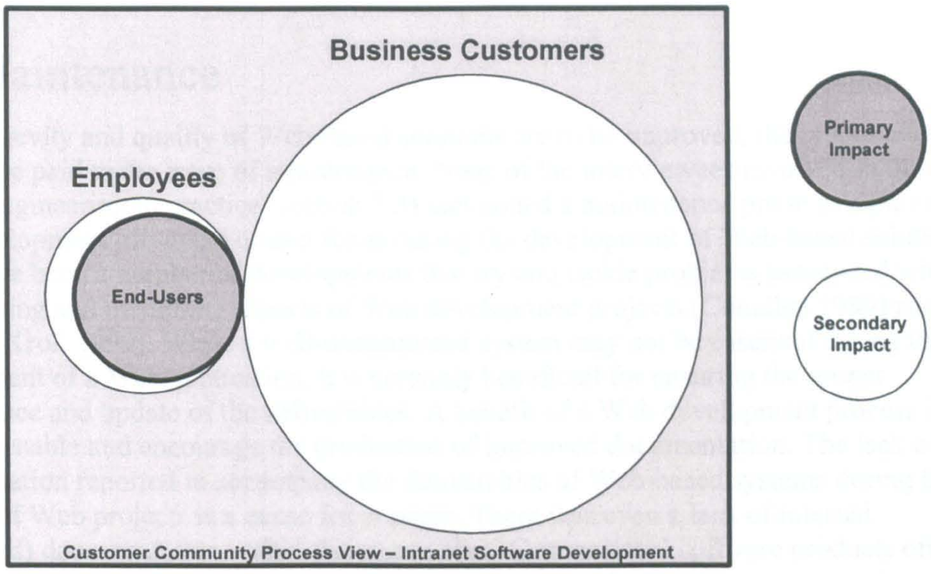


Figure 7. The Required Customer Community View in Processes for Intranet Web-based Development

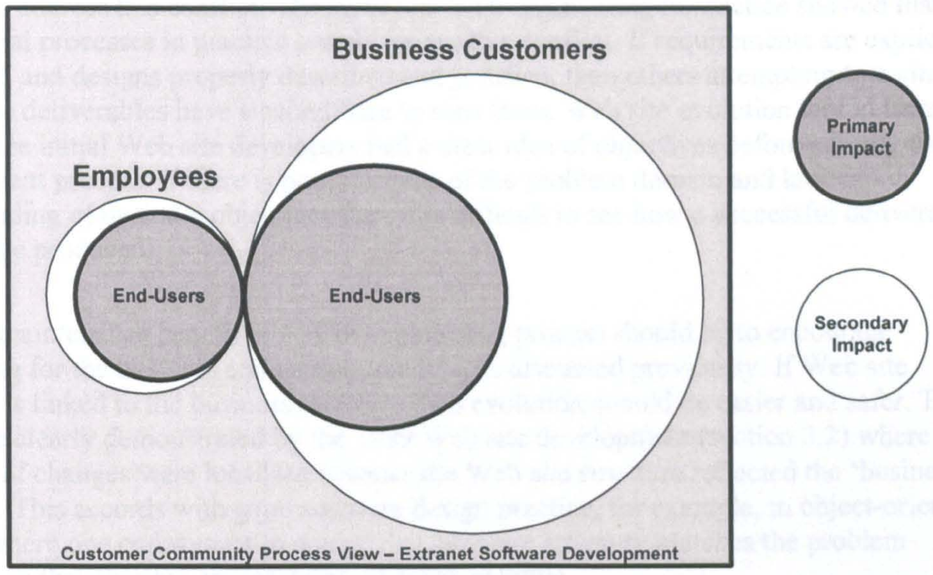


Figure 8. The Required Customer Community View in Processes for Extranet Web-based Development

4.6 Requirements and Testing

The empirical data presented in chapter 3 showed the poor focus on and the lack of activity in the requirements engineering and testing phases in commercial Web engineering process approaches. This lack of focus was observed by many of the interviewees involved in the survey of Web engineering in practice (section 3.3) to be one of the main problem areas with their current Web development process approach. If a Web engineering process is to be as successful as possible, then it must encourage those adopting it to focus more on requirements engineering and testing during their Web development process. In essence a

Web engineering process should encourage those using the process to ask the following questions. Are we building the product right? How will we ensure that we have built the product right?

4.7 Maintenance

If the longevity and quality of Web-based solutions are to be improved, then more attention needs to be paid to the issue of maintenance. None of the interviewees involved in the survey of Web engineering in practice (section 3.3) mentioned a maintenance phase as a part of their development life-cycle or as a focus during the development of Web-based solutions. There have been a number of developments that try and tackle problems associated with documenting and modelling aspects of Web development projects (Conallen 1999) and (Ward & Kroll 1999). While a well-documented system may not be essential during the development of a Web application, it is certainly beneficial for ensuring the proper maintenance and update of the deliverables. A benefit of a Web development process is that it should enable and encourage the production of improved documentation. The lack of documentation reported to accompany the deliverables of Web-based systems during the delivery of Web projects is a cause for concern. There was even a lack of internal (embedded) documentation within the source files. Conventional software products often contain useful embedded commentary (e.g. Javadoc) even if other documentation is incomplete, inaccurate or simply non-existent. Clearly there is a conflict between time-to-market pressures and production of documented deliverables to support maintenance and evolution of Web applications. However, this is a balance that must be addressed by each respective project and organisation. A Web engineering process needs to encourage those using it to address this conflict. The survey of Web engineering in practice showed that commercial processes in practice simply ignore this conflict. If requirements are explicitly identified, and designs properly described and justified, then others attempting to maintain or evolve the deliverables have a sound base to start from. Web site evolution would be much easier if the initial Web site developers had a clear idea of objectives before starting the development process. If there is poor analysis of the problem domain and lack of understanding of business objectives then it is difficult to see how a successful deliverable is likely to be produced.

Another maintenance benefit of a Web engineering process should be to encourage structuring for the business and domain models, as discussed previously. If Web site structure is linked to the business structure then evolution should be easier and safer. This was quite clearly demonstrated by the 1999 Web site development (section 3.2) where the majority of changes were localised because the Web site structure reflected the 'business' structure. This accords with good software design practice, for example, in object-oriented design, where one endeavours to ensure that software structure matches the problem structure so that changes are localised, as far as possible.

A Web engineering process should encourage development teams to have a clear view of the long-term objectives of a Web development project. It should be possible to develop the Web site to allow for the most likely points of change or expansion during maintenance and evolution. This is strongly linked to the point made in the previous paragraph. If you have a good understanding of the business and domain then you can anticipate change. Again, the development of the 1999 Web site (section 3.2) provides a good example of this when the corporate identity was added to the site.

Effective communication within the development team is another Web development process characteristic that will support maintenance and evolution. Web development teams have to be multidisciplinary bringing together technical experts, creative designers, business experts, domain experts and team leaders. The evolution of a Web site will involve the same mix of skills and at the moment there is no common communication notation, shared by these different people, to aid understanding of existing products or discuss the impact of changes. An essential underpinning for a development method must be the integration of suitable notations for capturing these different viewpoints that can then be used as a basis for communication and evolution.

4.8 Summary

Based on the empirical evidence presented in chapter 3, this chapter discussed in detail the seven criteria for a Web engineering process, see section 4.1-4.7. These criteria are used later in the dissertation to assess the applicability of development processes to Web engineering. The next chapter uses the criteria to evaluate the suitability of three traditional software processes for Web engineering.

5 Traditional Software Engineering Processes Support for Web Engineering

The objective of this chapter is to evaluate how traditional software engineering processes support Web engineering. Ideally, such an evaluation would be based on empirical evidence and detailed experience reports of using such processes for Web engineering. However, I have not attempted an empirical evaluation of these processes in practice and, while this is desirable, it would be extremely difficult to do under consistent experimental conditions. Therefore, this chapter is based on a critical review of the literature describing three representative traditional processes, assessing how their characteristics compare with the criteria for Web engineering processes, described in chapter 4.

There exist a number of different plan-driven and agile processes in the literature and in use within commerce today. Definitions of plan-driven processes and agile processes are presented by Lindvall et al. (2002) in the report of a "eWorkshop" held to help gather empirical evidence on the effectiveness and classification of agile projects. Many prominent plan-driven and agile process advocates participated in the eWorkshop, including Kent Beck author of 'Extreme Programming Explained: Embrace Change' (Beck 2000), Barry Boehm who is famous for the 'Spiral Model' (Boehm 1998) and Alistair Cockburn who developed the 'Crystal' family of agile processes (Cockburn 2002). The purpose of this research is not to propose new definitions for plan-driven or agile processes, nor does it attempt to do so. To assist the reader with the different process discussions that follow throughout the rest of the dissertation, working definitions presented by Lindvall et al. (2002) for plan-driven and agile processes are reproduced below:

> Plan-driven methods are those in which work begins with the elicitation and documentation of a 'complete' set of requirements, followed by architectural and high-level-design development and inspection.
>
> Agile Methods are:
>
> - Iterative (Delivers a full system at the very beginning and then changes the functionality of each subsystem with each new release);
> - Incremental (The system as specified in the requirements is partitioned into small subsystems by functionality. New functionality is added with each new release);
> - Self-organizing (The team has the autonomy to organize itself to best complete the work items);
> - Emergent (Technology and requirements are "allowed" to emerge through the product development cycle).
>
> All agile methods follow the four values and twelve principles of the Agile Manifesto (Beck et al., 2001).

Before the emergence of the agile community (Beck et al. 2001) a number of organisations began to recognise the need "to deliver software support in ever-decreasing time scales" (Stapleton 1997, p. xi). To help address these reduced time-to-market pressures a number of developments known as Rapid Application Development (RAD) approaches emerged. Primarily these developments emerged in the form of tools, techniques and processes to support RAD project scenarios. The following working definition of RAD reproduced from (Berry & Naumann 2000) serves to assist the reader with the process discussions presented throughout the rest of this dissertation:

> RAD (rapid application development) is a concept that products can be developed faster and of higher quality through:
>
> - Gathering requirements using workshops or focus groups;
> - Prototyping and early, reiterative user testing of designs;
> - The re-use of software components;
> - A rigidly paced schedule that defers design improvements to the next product version;
> - Less formality in reviews and other team communication.

Boehm and Turner (2003a) provide a comparison of plan-driven and agile processes in the first appendix of their book *Balancing Agility and Discipline: A Guide for the Perplexed*. They provide a "thumbnail sketch of each method" and then geographically characterise a number of plan-driven and agile processes "along three factor dimensions". The first dimension *Levels of Concern* and second dimension *Life-cycle Activities* identify the organisational scope and life-cycle activities respectively, for which a process provides specific guidance. The third dimension, *Sources of Constraint*, identifies constraints placed by the process on the implementor. Boehm and Turner's comparison was inspired by Abrahamsson et al. (2002), who provide a definition and classification for agile processes and a comparison of ten agile processes (Beck et al. 2001). Abrahamsson et al. (2002) highlight the differences between and similarities of the processes compared, and based on this analysis identify future research needs in the area of agile processes. Abrahamsson et al. (2002) conclude that when "considering the adoption of the [agile] methods, the size of the development team is currently one of the main decisive issues". In addition they identify the need for empirical studies "for evaluating the effectiveness and the possibilities of using new agile software development methods" and "adoption or selection models to be used by practitioners".

No.	Criteria for a Web Engineering Process
1.	Short development life-cycle times
2.	Different business models
3.	Multidisciplinary development teams
4.	Small development teams working in parallel on similar tasks
5.	Analysis and Evaluation
6.	Requirements and Testing
7.	Maintenance

Table 3. The Criteria for a Web Engineering Process

This chapter evaluates one traditional plan-driven software engineering process, one Rapid Application Development (RAD) process, and one agile process against the criteria for a Web engineering process, see table 3 or chapters 3 and 4 for more information about the criteria. The plan-driven software engineering process evaluated is the Unified Software Development Process (USD) (Jacobson, Booch & Rumbaugh 1999), the RAD process evaluated is Dynamic Systems Development Method (DSDM) (Stapleton 1997) and the agile process evaluated is eXtreme Programming (XP) (Beck 2000).

The three process approaches selected, USD, DSDM and XP provide illustrative examples of widely used plan-driven, RAD and agile processes respectively. The purpose of this discussion is not to provide an exhaustive evaluation of plan-driven, RAD and agile processes against the criteria for a Web engineering process. Instead, USD, DSDM and XP are used as illustrative baseline examples to evaluate the suitability of traditional software engineering processes for Web engineering across the plan-driven, RAD and agile software engineering process spectrum, see Figure 9. The three processes evaluated are all advocated for use on developing bespoke software projects and this is one of the reasons why they are considered for evaluation against the criteria for a Web engineering process. In addition to being advocated for bespoke software development, each of the three processes are chosen for their independence from any one organisation or company and their wide influence and usage within commerce. The three processes also serve as illustrative examples of some of the major process developments presented within the literature over the past thirty years. These major developments include the Waterfall Model (Royce 1970) and Spiral Model (Boehm 1988) as realised through USD, the move towards RAD approaches as illustrated by DSDM, and the emergence of agile processes (Beck et al. 2001) of which XP is an example.

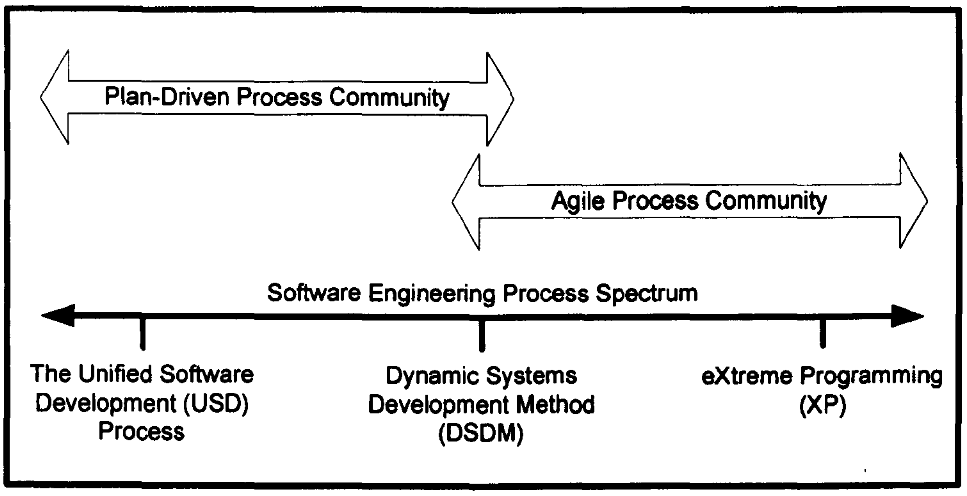

Plan-Driven Process Community
Agile Process Community
Software Engineering Process Spectrum
The Unified Software Development (USD) Process
Dynamic Systems Development Method (DSDM)
eXtreme Programming (XP)

Figure 9. USD, DSDM and XP's Position on the Software Engineering Process Spectrum

This chapter briefly describes the three commercial software engineering processes, USD, DSDM and XP, using the process elements discussed in section 2.2. As references to each criterion surface, through the description of process elements, they are given a ranking to show how strongly each process supports the criteria for a Web engineering process. The evidence and a rationale behind each ranking is also presented.

5.1 The Unified Software Development Process (USD)

The Unified Software Development (USD) Process has been chosen as an exemplar of a plan driven software engineering process for three reasons (Jacobson, Booch & Rumbaugh 1999): adopted use in many organisations; application on projects in a number of different commercial sectors; and adoption by organisations within and across many geographical and political boundaries. The USD has been influenced by a number of process enhancements. These developments include architecture centric (Kruchten 1995) object-oriented technology practices and accompanying techniques (Rumbaugh, Jacobson & Booch 1998) that have emerged over the past twenty years, and the iterative and incremental risk based development approach as illustrated by the Spiral Model (Boehm 1988). The techniques realised in the Unified Modeling Language (UML) (Rumbaugh, Jacobson & Booch 1998) underpin USD to provide "a standard way to visualise, specify, construct, document, and communicate the artefacts of a software intensive system" (Jacobson, Booch & Rumbaugh

1999, p. xix). USD is used in many organisations and represents the "unification of many methodologists – not just at Rational but also at the hundreds of customer sites" that have been using the process over many years (Jacobson, Booch & Rumbaugh 1999, p. xxvi).

The Rational Unified Process (RUP) is a Web-based product that is "a specific and detailed instance" of USD (Kruchten 2003, p. xiv). RUP and its extensions to Web application development (Ward & Kroll 1999), presented in the 2003 version of the RUP product (Rational 2003), is given separate consideration for its support for the criteria for a Web engineering process in Chapter 7. The rest of section 5.1 discusses the elements of USD that are being used to describe the USD process.

5.1.1 USD: Process Life-cycle and Phases

The USD process is primarily geared to developing software solutions as the following quote from its creators Jacobson, Booch and Rumbaugh (1999, p. 4) illustrates:

> First and foremost the Unified Process is a software development process. A software development process is the set of activities needed to transform a user's requirements into a software system.

The USD process contains five main workflows or activities and four life-cycle phases with different types of iteration as illustrated by the following quote from Jacobson, Booch and Rumbaugh (1999, p. 2):

> A chapter is devoted to each workflow: requirements, analysis, design, implementation, and test. The workflows will be used later [in Part III] as substantive activities in the different kinds of iterations in the four phases into which we divide the process ...we describe concretely how work is performed in each phase: in inception to make a business case, in elaboration to create the architecture and make a plan, in construction to grow the architecture into a deliverable system, and in transition to assure that the system operates correctly in the user's environment.

USD places comprehensive focus on requirements capture, including functional requirements (use case model) and non-functional requirements (supplementary specification). There is a strong software-focused testing element contained within USD, however there is no explicit mention of testing issues specific to Web engineering. USD is therefore ranked as showing partial support for criterion 6 'Requirements and Testing'.

USD also discusses business modelling workflow, normally carried out before deriving software requirements as illustrated by the following quote from Jacobson, Booch and Rumbaugh (1999, p. xxvi):

> The process was also expanded with a new workflow for business modelling, based on [*The Object Advantage: Business Process Re-engineering with Object Technology* book by Jacobson et al.], that is used to drive requirements from the business processes the software was to serve.

Business Modelling within USD is separate to the software process and is considered optional, as the following quote (Jacobson, Booch & Rumbaugh 1999, p. 10) illustrates:

> The system *may* also have a domain model or a business model that describes the business context of the system.

USD does not mention the importance of evaluation with end-users in Web engineering or how to support this activity within the USD process. USD is therefore ranked as showing partial support for criterion 5 'Analysis and Evaluation'.

The impact reflected within the USD process is from the business and domain model to the software model only. There is no mention of impact back into the business and domain models from the software model as illustrated by the absence of the business and domain models in Figure 10. Figure 10 describes the "models of the Unified Process" and "the dependencies between the use-case model and the other models" (Jacobson, Booch & Rumbaugh 1999, p. 10).

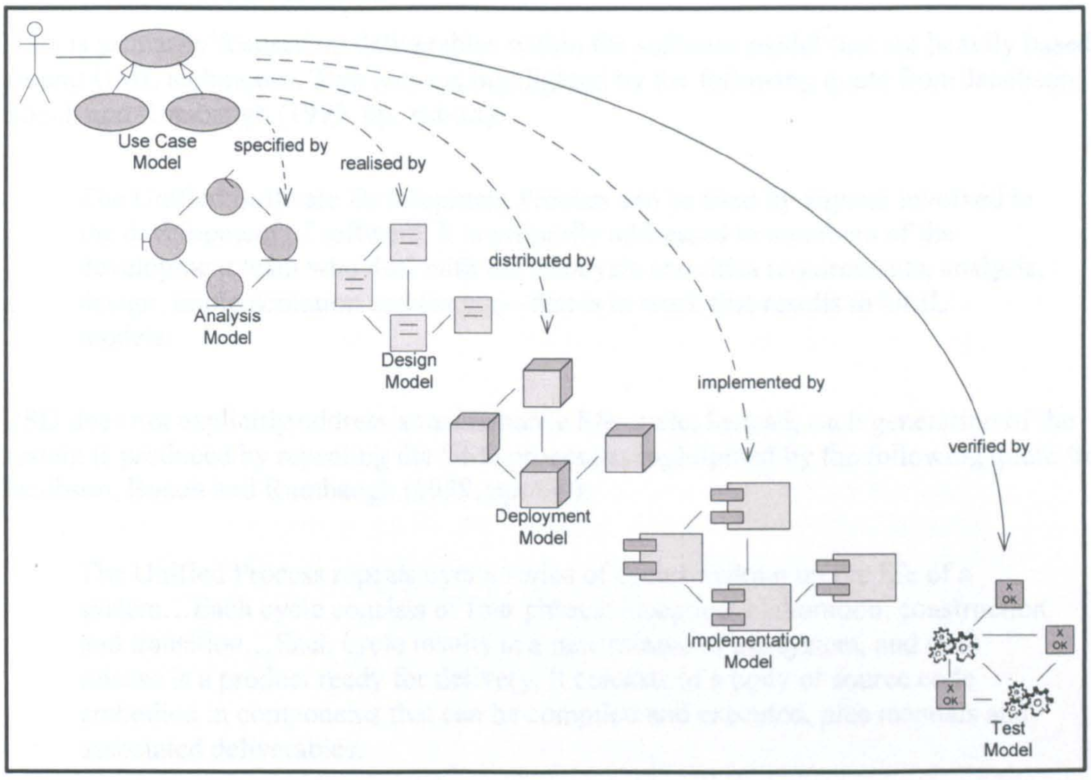


Figure 10. Models of the Unified Process. There are dependencies between many of the models. As an example, the dependencies between the use-case model and the other models are indicated. Reproduced from Jacobson, Booch and Rumbaugh (1999, p. 10)

USD's poor reflection of support for the impact from the software model to business and domain model is reinforced by the following two quotes (Jacobson, Booch & Rumbaugh 1999, pp. 121-122):

> Domain modelling is usually done in workshops by domain analysts, who use UML and other modelling languages to document the results. … The purpose of domain modelling is to understand and describe the most important classes within the context of the domain.

This view of the domain models explains the impact of the domain on the software model but gives no indication that the software model will impact the domain.

> Business modelling is a technique for understanding the business processes of an organisation. ... The goal is to identify the use cases of the software and the relevant business entities to be supported by the software, so we should just model enough to understand the context.

Again, there is a clear recognition that the business model impacts the software model, but no suggestion that the business model should be affected by the software model. The statement "so we should model just enough to understand context" (Jacobson, Booch & Rumbaugh 1999, p. 122) strengthens the point made in Chapter 4 that in most traditional software engineering processes the software model is impacted only by a partial section of the business and domain models. USD shows no support for criterion 2 'Different business models'.

5.1.2 USD: Deliverables

USD is primarily focused on deliverables within the software model that are heavily based around UML techniques. This focus is highlighted by the following quote from Jacobson, Booch and Rumbaugh (1999, pp. xix-xx):

> The Unified Software Development Process can be used by anyone involved in the development of software. It is primarily addressed to members of the development team who deal with the life-cycle activities requirements, analysis, design, implementation and testing – that is in work that results in UML models.

USD does not explicitly address a maintenance life-cycle. Instead, each generation of the system is produced by repeating the USD process as highlighted by the following quote from Jacobson, Booch and Rumbaugh (1999, pp. 8-9):

> The Unified Process repeats over a series of cycles making up the life of a system...Each cycle consists of four phases: inception, elaboration, construction and transition...Each cycle results in a new release of the system, and each release is a product ready for delivery. It consists of a body of source code embodied in components that can be compiled and executed, plus manuals and associated deliverables.

Each generation of a product produced by the USD process is focussed on the deliverables within the software model. This is illustrated by the number of documented deliverables that are produced within the software models as highlighted by this quote from Jacobson, Booch and Rumbaugh (1999, p. 9):

> The finished product includes the requirements, use cases, non-functional requirements, and test cases. It includes the architecture and the visual models – artefacts modelled by the Unified Modeling Language. In fact, it includes all the elements we have been talking about ... because it is these things that enable the stakeholders – customers, users, analysts, designers, implementers, testers, and management – to specify, design, implement, test, and use a system. Moreover, it is these things that enable the stakeholders to use and modify the system from generation to generation.

USD's strong focus on the deliverables in the software model but absence of deliverables in the business, domain and creative design models ranks USD as showing weak support for criterion 7 'Maintenance'. In addition, the USD process is too predictive and heavy weight requiring the exhaustive development of a number of deliverables within the software model alone, and is therefore ranked as showing weak support for criterion 1 'Short development life-cycle times'.

5.1.3 USD: Activities

There is little focus within USD on how to coordinate parallel activities or many small teams, as is commonplace within large Web engineering projects. Thus, USD is ranked as showing weak support for criterion 4 'Small development teams working in parallel on similar tasks'.

5.1.4 USD: Roles

As discussed previously USD is primarily a software engineering process, and places strong focus on the roles within the software model. However, no attention is placed within USD on the increased visibility of the creative design, business or domain roles when building Web solutions. USD does not discuss the roles outside of the software model required in Web engineering, and therefore shows no support for criterion 3 'Multidisciplinary development teams'.

5.1.5 USD: Tools and Techniques

As discussed previously USD places great focus on techniques, as realised with UML, to support activities within the software model. The proceeding quote illustrates the close relationship between USD and UML (Jacobson, Booch & Rumbaugh 1999, p. xix):

> This book presents the software process that was constantly on our minds when we developed the Unified Modeling Language. While UML gives us a standard way to visualise, specify, construct, document, and communicate the artefacts of a software intensive system, we of course recognise that such a language must be used within the context of an end-to-end software process. UML is a means, not an end. The ultimate end is a robust, resilient, scalable software application. It takes both a process and a language to get there, and illustrating the process portion is the goal of this book.

USD itself is independent of tools although many types of Computer Aided Software Engineering (CASE) tools are recommended to automate and assist with the USD process. USD's younger sibling RUP is closely integrated with a number of CASE tools within IBM's Rational (2003) product suite.

5.1.6 USD: Support for the Criteria for a Web Engineering Process

Table 4 below summarises the Unified Software Development Process support for the criteria for a Web engineering process.

No.	Support	Comments
1. Short development life-cycle times	*Weak*	Process is plan-driven and is too predictive and heavy weight for many typical Web engineering projects
2. Different	*None*	The impact reflected within the process is

business models		from the business and domain model to the software model. There is no mention of impact back into the business and domain models from the software model.
3. Multidisciplinary development teams	*None*	No focus on the increased visibility of the creative design, business or domain roles when building Web solutions.
4. Small development teams working in parallel on similar tasks	*Weak*	No mention of parallel activities or coordinating many small teams.
5. Analysis and Evaluation	*Partial*	USD includes business modelling workflow before deriving software requirements. No explicit mention of evaluation with end-users or how to support this activity within the process.
6. Requirements and Testing	*Partial*	Explicit focus on capture of all types of requirements including functional (use case model) and non-functional (supplementary specification). Strong testing element but no explicit mention of Web site testing.
7. Maintenance	*Weak*	No explicit mention of maintenance issues outside the software model. However, a number of documented deliverables are produced within the software models.

Table 4. USD Support for the Criteria for a Web Engineering Proces

5.2 Dynamic Systems Development Method (DSDM)

Dynamic Systems Development Method (DSDM) is a Rapid Application Development (RAD) framework for building business system solutions. DSDM defines "a process and a set of products" at a high-level so that they can be tailored for any technical and business environment (Stapleton 1997, p. xiv). The following quote from Stapleton (1997, p. xi) describes DSDM in her own words:

> Dynamic Systems Development Method (DSDM) provides a framework of controls and best practices for the rapid application development (RAD) of high quality business system solutions.

The first version of DSDM was created in 1994, many years before the emergence of the Agile Manifesto (Beck et al. 2001). Recently DSDM has become classified by many as an agile process (Abrahamsson et al. 2003), although as Boehm and Turner (2003, p. 178) observe it is a "Heavy" agile process that has many similarities with plan-driven processes, such as the Rational Unified Process (RUP) (Rational 2003) and Team Software Process (TSP) (Humphrey 1999). The close alignment between DSDM, RUP and TSP is highlighted by the number of document centric work products defined for each phase in the DSDM life-cycle.

The DSDM Consortium is a not-for-profit organisation established in 1994 with many tens of commercial members (DSDM 2004b), from small companies to extremely large commercial organisations, including some of the world's largest banking organisations, such as The Royal Bank of Scotland Group (RBS 2004), and large technology companies, such as

IBM (IBM 2004). The rest of section 5.2 discusses the elements of DSDM that are used to describe the DSDM process.

5.2.1 DSDM: Process Life-cycle and Phases

Figure 11 shows the DSDM development process, also colloquially known as 'the three pizzas and a cheese' (Stapleton 1997, p. 3). The dark (black coloured) arrows represent "forward paths" with the lighter (grey coloured) arrows representing "recognised routes back to evolve the system". The DSDM development life-cycle has five phases, the first two phases are performed sequentially and used to feed into the remaining three phases that are iterative and incremental. DSDM's five phases are described below with direct quotes from Stapleton (1997, pp. 3-10):

1. *Feasibility Study*. "This phase is more an assessment of whether or not the DSDM approach is the right one for the project than a traditional feasibility study";
2. *Business Study*. "... this is a short exercise to achieve enough understanding of the business and technical constraints to move forward with safety. As its name suggests, the prime activity here is to get a good understanding of the business processes to be automated and their information needs";
3. *Functional Model Iteration*. "The focus of the functional model iteration is on refining the business aspects of the system that is building on the high-level functional and information requirements identified during the business study. To this end, both standard analysis models and software are produced";
4. *System Design and Build Iteration*. "The design and build iteration is where the system is engineered to a sufficiently high standard to be safely placed in the hands of the users. The major product here is obviously the Tested System";
5. *Implementation*. "The implementation phase covers the cutover from the development environment to the operational environment. This includes training the users and handing over the system to them".

The primary activity of the business study phase, as described by Stapleton (1997, p. 6), "is to get a good understanding of the business processes to be automated and their information needs". This statement clearly shows DSDM support for reflecting impact from the business and domain models to the software model. However, like other traditional software engineering processes DSDM does not support any impact from the software model to the business and domain models. This lack of support for impact from the software model to the business and domain models is reinforced by the names used to describe the DSDM process life-cycle which underlie the primary focus of DSDM on delivering software, see Figure 11. DSDM is therefore ranked as showing no support for criterion 2 'Different business models'.

DSDM places strong emphasis on requirements capture. "MoSCoW" rules is an acronym used to describe the relative prioritisation of requirements in a DSDM project. The acronym is the brainchild of Dai Clegg of Oracle who was one of the early participants in the DSDM Consortium (DSDM 2004b). The 'o's in MoSCoW are "there for fun" (Stapleton 1997, p. 28). The rest of the word stands for (Stapleton 1997, p. 29):

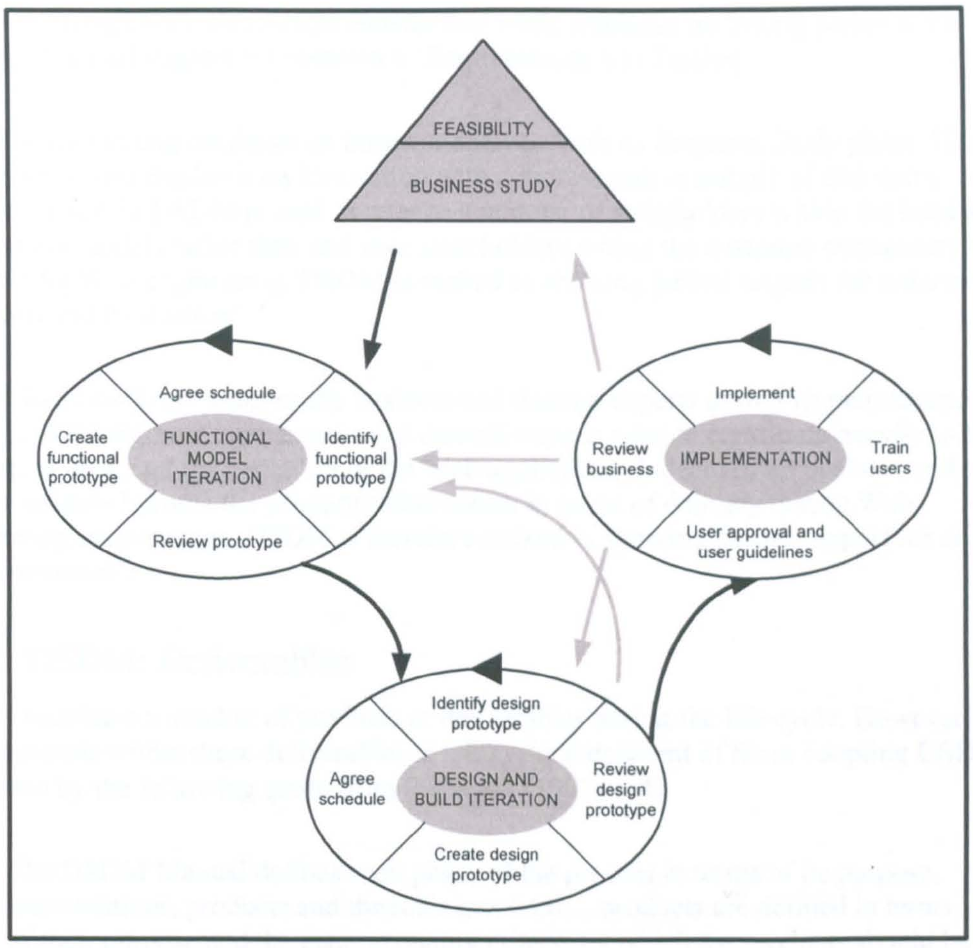

FEASIBILITY
BUSINESS STUDY
Agree schedule
Create functional prototype
FUNCTIONAL MODEL ITERATION
Identify functional prototype
Review prototype
Implement
Review business
IMPLEMENTATION
Train users
User approval and user guidelines
Identify design prototype
Agree schedule
DESIGN AND BUILD ITERATION
Review design prototype
Create design prototype

Figure 11. The DSDM Development Process. Reproduced from Stapleton (1997, p.3)

- *Must have* "requirements that are primary to the success of the system. The next release of this system must successfully address these requirements if it is to be workable and useful";
- *Should have* "requirements are those that are secondary to the success of the system. Successfully addressing these requirements will enhance the system, but are not primary and can be dropped from the current release to address time-to-market pressures";
- *Could have* "requirements are tertiary to the success of the system and can easily be dropped from the current release";
- *Want to have but will not have this time round* "requirements that are valuable but can wait until later".

However there is weak emphasis on testing, with the exception of brief testing discussions indicating that both unit and other types of testing activities need to occur during DSDM. The following is an example of one of Stapleton's (1997, p. 7) discussions on testing:

It is essential that the software components of the functional model are tested as they are produced. This obviously includes unit testing, but many other classes of testing as are possible should be undertaken as well.

DSDM's strong emphasis on requirements and weak emphasis on testing places it into a ranking of partial support for criterion 6 'Requirements and Testing'.

DSDM shows strong emphasis on business analysis with its Business Study phase. However, DSDM shows no emphasis on Evaluation with a representative sample of end-users. Indeed the term 'User' in DSDM is used to refer to a number of stakeholders within the business and domain models rather than end-user stakeholders within the customer community view required for Web engineering. DSDM is ranked as showing partial support for criterion 5 'Analysis and Evaluation'.

DSDM loses its 'User' roles or the business and domain experts during its maintenance stage. In Web engineering, business and domain experts need to contribute heavily to the creation of data, and the evolution of the Web application. Indeed, often business and domain experts become the primary stakeholders in terms of delivery during Web engineering maintenance. DSDM is therefore ranked as showing partial support for criterion 7 'Maintenance'.

5.2.2 DSDM: Deliverables

DSDM mandates a number of products or deliverables during the life-cycle. However, the exact contents within these deliverables is left to the judgement of those adopting DSDM as illustrated by the following quote from Stapleton (1997, p. 4):

> The DSDM Manual defines each phase of the process in terms of its purpose, preconditions, products and the roles involved... products are defined in terms of their purpose and the generic quality criteria by which the products should be assessed. The actual contents will be determined by local practices, such as the preferred analysis or project management techniques.

DSDM as a RAD approach is designed to help address faster time-to-market than can be achieved with plan-driven processes, i.e. DSDM aims to achieve project time scales that are "less than a few years" (Stapleton 1997, p. xv). The major mechanism used to achieve this is the timeboxing as described below (Stapleton 1997, p. xvii):

> DSDM aims to deliver systems to time-scales that would be impossible using the waterfall approach...The major instrument for controlling work is the timebox. The timebox in DSDM is a short period of time (a matter of days or a few weeks) within a project when something is produced to defined quality objectives.

DSDM timeboxes are used to determine the allowed time, of between two and six weeks, to produce deliverables and thus assist DSDM projects with addressing time-to-market pressures. These timescales associated with a DSDM release are difficult to estimate as this is dependent on the developers, the organisation and the project being developed. However, the time scales associated are generally shorter than those associated with plan-driven processes but longer (Boehm & Turner 2003a) than those associated with other agile processes. DSDM is therefore ranked as showing partial support for criterion 1 'Short development life-cycle times'.

5.2.3 DSDM: Activities

DSDM "does not supply a particular set of activities within each phase as this will depend on the application being built, the organisation building it and the organisation for whom it is being built" (Stapleton 1997, p. 4). However, DSDM is based on nine principles, reproduced below from Stapleton (1997, p. xvi), "that have been found to be necessary if a quality system is to be supplied in the time-scale required by the business". The first four principles define the foundations on which DSDM is built and the latter five principles provide guidance on the structure of the method:

1. Active user involvement is imperative;
2. DSDM teams must be empowered to make decisions;
3. The focus is on frequent delivery of products;
4. Fitness for business purpose is the essential criterion for acceptance of deliverables;
5. Iterative and incremental development is necessary to converge on an accurate business solution;
6. All changes during development are reversible;
7. Requirements are baselined at a high-level;
8. Testing is integrated throughout the life-cycle;
9. A collaborative and cooperative approach between all stakeholders is essential.

These principles ultimately have a strong influence on the types of activities used to apply DSDM in a real project scenario and many of them (1, 2, 3, 5, 9) are closely aligned with the Agile Manifesto (Beck et al. 2001). More details about DSDM's alignment with the Agile Manifesto can be found on the DSDM Consortium's Web site (DSDM 2004c).

5.2.4 DSDM: Roles

A DSDM team "consists of developers and users working together" (Stapleton 1997, p. 42). A DSDM team is recommended to be kept small and should consist of two to six people. A typical DSDM project will have "one or two teams, but a large project can grow to as many as six teams, all working in parallel" (Stapleton 1997, p. 43). Stapleton (1997, pp. 43-44) provides a short description of project structures to assist with scaling DSDM to large projects and assistance with scaling DSDM to many parallel teams. DSDM's comprehensive focus on project management (Stapleton 1997, p. 177) provides strong support for criterion 4 'Small development teams working in parallel on similar tasks'.

DSDM defines a number of roles. Stapleton (1997, pp. 42-43) gives a brief discussion of DSDM roles. In conjunction with the roles definitions provided by Stapleton the following description of DSDM roles is based upon the DSDM tour (DSDM 2004d):

1. *Executive Sponsor*: The "Project Champion". The purse-holder and ultimate decision maker in the business area;
2. *Visionary*: Usually responsible for getting a project started. The individual who is responsible for initiating the project through their vision for IT support in their business area;
3. *Ambassador User*: Generally comes from the business area being addressed (business expert). Responsible for bringing the knowledge of the user community into the team and disseminating information from the team to the rest of the users;
4. *Advisor User*: Brings day-to-day knowledge of the job being automated (domain expert). Primarily user is used to cover the disparate views that may exist. Involved on an ad-hoc basis as required by the needs of the project;

5. *Project Manager*: Can come from the user community or from IT;
6. *Technical Co-ordinator*: Sits above the individual development teams. Responsible for defining the system architecture, ensuring the project is technically consistent and that all work produced is of sufficient technical quality;
7. *Team Leader*: Ensures that a development team functions as a whole;
8. *Developer*: Models and interprets user requirements, and converts them into prototypes and deliverable code;
9. *Tester*: Performs the non-user testing;
10. *Scribe*: Sits in on all meetings and workshops to record requirements, etc;
11. *Facilitator*: Independent of the project team, manages the workshops;
12. *Specialist Roles*: Business Architect, Quality Manager, System Integrator...

While there are many roles defined within DSDM there is no mention of the creative design role. In addition, while DSDM defines a number of 'User roles' these are primarily stakeholders used to represent the business and domain models. DSDM is therefore ranked as showing partial support for criterion 3 'Multidisciplinary development teams'.

5.2.5 DSDM: Tools and Techniques

As the following two quotes from Stapleton (1997, pp. xiv-xv) illustrate that DSDM is both technique and tool independent:

> There are no prescribed techniques, but suggested paths are supplied for implementers of both structured and object-oriented approaches.

> The Consortium was inaugurated in January 1994 with the aim of producing a public-domain, commonly agreed method that would be tool independent.

5.2.6 DSDM: Support for the Criteria for a Web Engineering Process

Table 5 below summarises Dynamic Systems Development Method support for the criteria for a Web engineering process.

No.		Support	Comments
1.	Short development life-cycle times	*Partial*	Designed for faster time-to-market compared with plan-driven processes, i.e. "less than a few years"
2.	Different business models	*None*	No impact from the software model to the business and domain models. Impact only reflected from the business and domain models to the software model. As illustrated by the primary focus of the business study phase which "is to get a good understanding of the business processes to be automated and their information needs" and the names used to describe the DSDM process life-cycle.
3.	Multidisciplinary development teams	*Partial*	No mention of the creative design role. Mixture of business and domain roles on standard teams.
4.	Small development teams working in	*Strong*	Discussion of how to scale to six teams each of six developers, although there is no specific focus on Web engineering projects.

parallel on similar tasks		
5. Analysis and Evaluation	*Partial*	Strong analysis. No evaluation with a representative sample of end-users. Relies on end-user substitutes i.e. 'Ambassador User' and 'Advisor User'.
6. Requirements and Testing	*Partial*	Strong requirements capture approach using MoSCoW rules. Weak emphasis on Testing.
7. Maintenance	*Partial*	DSDM loses User roles (business and domain experts) for the maintenance stage. Business and domain experts need to contribute heavily to this phase in Web engineering projects for the creation of data, often they become the primary deliverer during Web engineering maintenance.

Table 5. DSDM Support for the Criteria for a Web Engineering Process

5.3 eXtreme Programming (XP)

eXtreme Programming (XP) (Beck 2000) is arguably the most famous agile process (Beck et al. 2001). XP's notoriety arose during the late 1990s, after its use on the C3 project at Chrysler and has gained the most attention of any agile process in the associated literature. XP's wider exposure in comparison to other agile processes is illustrated by its prominence in the names for the two major conferences associated with the agile community, namely 'The International Conference on Extreme Programming and Agile Processes in Software Engineering' (XP 2004) and 'XP Agile Universe' (XP Agile Universe 2004). XP is described by its founder Beck (2000, pp. xvii-xviii) in the following quote:

> XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. It is distinguished from other methodologies by:
>
> - Its early, concrete, and continuing feedback from short cycles;
> - Its incremental planning approach, which comes up with an overall plan that is expected to evolve through the life of the project;
> - Its ability to flexibly schedule the implementation of functionality, responding to changing business needs;
> - Its reliance on automated tests written by programmers and customers to monitor the progress of development, to allow the system to evolve, and to catch defects early;
> - Its reliance on oral communication, tests, and source code to communicate system structure and intent;
> - Its reliance on an evolutionary design process that lasts as long as the system lasts;
> - Its reliance on the close collaboration of programmers with ordinary skills;
> - Its reliance on practices that work with both the short-term instincts of programmers and the long term interests of the project.

As Beck (2000, p. xviii) acknowledges in the following quote, none of the practices of XP are new. Instead, what differentiates XP from other processes is the unification and extreme focus placed on the practices advocated within XP and the synergy resulting from the fusion of these practices in one method.

> … none of the ideas in XP are new. Most are as old as programming. There is a sense in which XP is conservative – all its techniques have been proven over decades (for the implementation strategy) or centuries (for the management strategy). The innovation of XP is:
>
> - Putting all these practices under one umbrella;
> - Making sure they are practiced as thoroughly as possible;
> - Making sure the practices support each other to the greatest possible degree.

The rest of section 5.3 discusses the elements of XP that are used to describe the XP process.

5.3.1 XP: Process Life-cycle and Phases

> The ideal XP project goes through a short initial development phase, followed by years of simultaneous production support and refinement, and finally graceful retirement when the project no longer makes sense.

The above quote from Beck (2000, p. 131) describes the ideal XP project life-cycle. The phases in an XP project are described as follows, with direct quotes from Beck (2000):

- *Exploration*. The Exploration Phase is about gaining belief that you can go into production. "You are done with exploration when the customer is confident that there is more than enough material on the story cards to make a good first release and the programmers are confident that they can't estimate any better without actually implementing the system";
- *Planning*. "The purpose of the planning phase is for the customers and programmers to confidently agree on a date by which the smallest, most valuable set of stories will be done";
- *Iterations to First Release*. "The commitment schedule is broken into one- to four-week iterations. Each iteration will produce a set of functional test cases for each of the stories scheduled for that iteration";
- *Productionising*. "The end game of a release ('productionising') sees a tightening up of the feedback cycle. Instead of three-week iterations, you may go to one-week iterations. You may have a daily stand-up meeting so every-body knows what everybody else is working on";
- *Maintenance*. "Maintenance is really the normal state of an XP Project. You have to simultaneously produce new functionality, keep the existing system running, incorporate new people into the team, and bid farewell to members who move on";
- *Death*. There are two main reasons for the death of an XP Project. "The good reason to die – the customer is happy with the system and can't think of anything they would like to add for the foreseeable future. … There is also a not-so-good reason to die – the system just isn't delivering".

Analysis and Evaluation phases are missing from the XP life-cycle. XP places no focus on Evaluation with a representative sample of end-users. As a result, XP is ranked as showing no support for criterion 5 'Analysis and Evaluation'.

The following quote from Beck (2000, p. 83) describes the role of the 'business' customer in XP:

> Business people should choose:
>
> - The scope or timing of releases;
> - The relative priorities of the proposed features;
> - The exact scope of the proposed features.
>
> ... Since business decisions take place all through the life of a project, giving business people responsibility for business decisions implies that a customer is as much a part of an XP team as a programmer. In particular, for best results they sit with the rest of the team and are available full-time to answer questions.

Even this quote from a leading exponent of the agile approach emphasises that the role of the business and domain models is to provide input to the software model, not to interact with it. The impact reflected within the XP process is from the business and domain model to the software model. There is no mention of impact back into the business and domain models from the software model. This impact is emphasised by the customer role who "knows what to program" with the XP programmer role knowing "how to program". XP is therefore ranked as showing no support for criterion 2 'Different business models'.

Beck (2000, p. 135) describes maintenance as "really the normal state of an XP Project". Indeed, the practices at the core of XP lend themselves exceptionally well to projects where the non-functional requirements are primarily static, as is common during most maintenance phases. XP is therefore ranked as showing strong support for criterion 7 'Maintenance'.

XP places great focus on the writing of functional requirements through the Planning Game and Testing of functional requirements through the Testing practices. Non-functional requirements capture and testing are critical during Web engineering projects. However, non-functional requirements capture and verification is primarily ignored in XP. As a result, XP is categorised as showing partial support for criterion 6 'Requirements and Testing'.

5.3.2 XP: Deliverables

The primary deliverable in an XP project is a working 'code release'. There are a number of secondary deliverables employed to help ensure the success of the primary deliverable, i.e. story cards, work estimates, daily code builds and functional tests. However, in comparison to plan-driven processes XP is generally light in the number of deliverables mandated.

5.3.3 XP: Activities

There are a number of activities or practices used by an XP team. The practices proposed by XP are described as follows, with direct quotes from Beck (2000):

- *The Planning Game*. The purpose of the Planning Game is to "quickly determine the scope of the next release by combining business priorities and technical estimates";
- *Small Releases*. "Every release should be as short as possible, containing the most valuable business requirements. The release has to make sense as a whole – that is, you can't implement half a feature and ship it, just to make the release cycle shorter";

- *Metaphor*. "The metaphor in XP replaces much of what other people call 'architecture'". The purpose of the metaphor is to help "everyone on the project understand the basic elements and their relationships";
- *Simple Design*. "The system should be designed as simply as possible at any given moment. Extra complexity is removed as soon as it is discovered";
- *Testing*. "Programmers continually write unit tests, which must run flawlessly for development to continue. Customers write [functional] tests demonstrating that features are finished";
- *Refactoring*. "Programmers restructure the system without changing its behaviour to remove duplication, improve communication, simplify, or add flexibility";
- *Pair Programming*. In XP "All production code is written with two people looking at one machine, with one keyboard and one mouse";
- *Collective Ownership*. "Anybody who sees an opportunity to add value to any portion of the code is required to do so at any time". Thus "anyone can change any code anywhere in the system at any time";
- *Continuous Integration*. "Integrate and build the system many times a day, every time a task is completed";
- *40-Hour Week*. "Work no more than 40 hours a week as a rule. Never work overtime a second week in a row";
- *On-Site Customer*. "Include a real, live user on the team, available full-time to answer questions". The on-site customer must "sit with the team, available to answer questions, resolve disputes, and set small scale priorities";
- *Coding Standards*. "Programmers write all code in accordance with rules emphasising communication through the code".

Figure 12, reproduced from (Beck 2000, p. 70), diagrammatically represents how each of the XP practices supports one another. It is the fusion of these practices and the support these practices provide for each other that are the core of the XP process. However, it is possible to benefit from the adoption of many of these practices within another process.

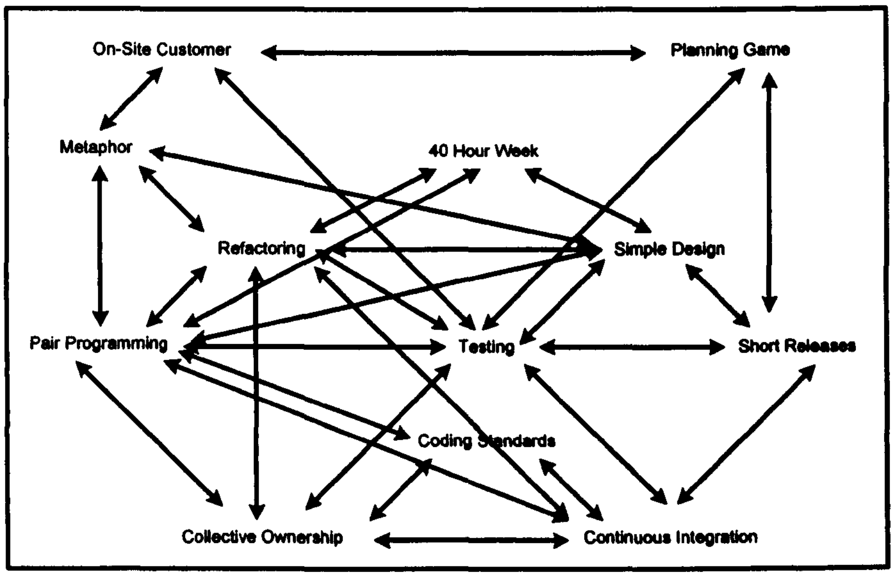


Figure 12. XP Practices Support for Each Other. Reproduced from Beck (2000, p. 70).

XP places great emphasis on short release cycles of "a few months at most" as emphasised by the short release practice. Advocating that "every release should be as small as possible, containing the most valuable business requirements". Within a release, XP recommends "one- to four-week iterations of customer-requested features for fine grained feedback about progress". As a result XP is ranked as providing strong support for criterion 1 'Short development life-cycle times'.

5.3.4 XP: Roles

XP defines seven roles within an XP project. These roles are described as follows, with direct quotes from Beck (2000):

1. *Programmer*. "The programmer is at the heart of XP". The programmer in addition to writing code needs to focus on communication with other team members, writing unit tests and refactoring the code base whenever necessary;
2. *Customer*. "The customer is the other half of the essential duality of extreme programming. The programmer knows how to program. The customer knows what to program";
3. *Tester*. The tester role is focused on helping the "customer choose and write functional tests", running these tests, if they are not part of the integration suite, and communicating the results to the rest of the team;
4. *Tracker*. The tracker role is responsible for "the loop on feedback" to assist the team with their estimates;
5. *Coach*. "...you are responsible for the process as a whole. You notice when people are deviating from the team's process and bring this to the team's attention";
6. *Consultant*. Consultants provide occasional "deep technical knowledge";
7. *Big Boss*. The Big Boss role helps the XP team by providing "courage, confidence, and occasional insistence that they [the XP team members] do what they say they do".

The roles listed above show no explicit focus on the increased visibility of the creative design or domain roles as part of the development team when building Web-based solutions. XP is therefore ranked as showing weak support for criterion 3 'Multidisciplinary development teams'.

XP is designed specifically for a development team of two to ten developers. XP does not consider how to scale to many small teams working on large projects, as is common in Web engineering. XP's position with respect to large teams is illustrated by the following quote from Beck (2000, p. xviii):

> XP is designed to work with projects that can be built by teams of two to ten programmers, that aren't sharply constrained by the existing computing environment, and where a reasonable job of executing tests can be done in a fraction of a day.

As a result, XP is ranked as showing no support for criterion 4 'Small development teams working in parallel on similar tasks'.

5.3.5 XP: Tools and Techniques

XP is tool independent. However, the tool JUnit (2004) is a very popular regression testing framework, written by Erich Gamma and Kent Beck, and is used widely to assist XP's Testing practice by Java developers.

5.3.6 XP: Support for the Criteria for a Web Engineering Process

Table 6 below summarises XP's support for the criteria for a Web engineering process.

No.	Support	Comments
1. Short development life-cycle times	*Strong*	XP places great emphasis on short release cycles of "a few months at most". Advocating that "every release should be as small as possible, containing the most valuable business requirements". Within a release, XP recommends "one- to four-week iterations of customer-requested features for fine grained feedback about progress".
2. Different business models	*None*	The impact reflected within the XP process is from the business and domain model to the software model. There is no mention of impact back into the business and domain models from the software model.
3. Multidisciplinary development teams	*Weak*	No focus on the increased visibility of the creative design or domain roles.
4. Small development teams working in parallel on similar tasks	*None*	XP is designed specifically for a development team of two to ten developers. XP does not consider how to scale too many small teams working on large projects as is common in many Web engineering endeavours.
5. Analysis and Evaluation	*None*	Analysis phase is missing from the XP. The customer is expected to answer all questions, or in the words of Kent Beck "the customer knows what to program". XP also uses the customer role as end-user substitutes. XP does not place focus on Evaluation with a representative sample of end-users.
6. Requirements and Testing	*Partial*	XP focuses on the testing of functional requirements. Both non functional requirements verification and validation is primarily ignored in XP.
7. Maintenance	*Strong*	XP designed for "maintenance" which "is really the normal state of an XP project." XP's almost solitary focus on functional requirements and poor focus on non-functional requirements prevent it from showing gaining a very strong ranking.

Table 6. XP Support for the Criteria for a Web Engineering Process

5.4 Summary

This chapter evaluated the support provided within three traditional software engineering processes for the criteria for a Web engineering process. The Unified Software Development (USD) Process, Dynamic Systems Development Method (DSDM) and eXtreme Programming (XP) are used as illustrative baseline examples to evaluate the suitability of traditional software engineering processes for Web engineering across the plan-driven, RAD and agile software engineering process spectrum. This chapter does not attempt an empirical evaluation of these processes in practice and, while this is desirable, it would be extremely difficult to do. Rather, the available literature on each process has been used to carry out an evaluation. Table 7 summarises the support shown by each of the three processes evaluated for the criteria for a Web engineering process.

No.	USD	DSDM	XP
1. Short development life-cycle times	*Weak*	*Partial*	*Strong*
2. Different business models	*None*	*None*	*None*
3. Multidisciplinary development teams	*None*	*Partial*	*Weak*
4. Small development teams working in parallel on similar tasks	*Weak*	*Strong*	*None*
5. Analysis and Evaluation	*Partial*	*Partial*	*None*
6. Requirements and Testing	*Partial*	*Partial*	*Partial*
7. Maintenance	*Weak*	*Partial*	*Strong*

Table 7. Summary of USD, DSDM and XP Support for the Criteria for a Web Engineering Process

Analysis of Table 7 indicates the following with respect to support within traditional software engineering processes for the criteria for a Web engineering process:

1. **Short development life-cycle times**. Our evaluation indicates that support for time-to-market pressures is better addressed with an agile process approach. Not surprisingly, as one moves through the plan-driven processes towards the RAD and agile processes, time-to-market pressures are better addressed;
2. **Different business models**. Neither plan-driven, rapid application development or agile software engineering processes suggest support for business process re-engineering, evolution or creation;
3. **Multidisciplinary development teams**. The support for the roles required to develop Web engineering projects is limited, with only DSDM showing partial support;
4. **Small development teams working in parallel on similar tasks**. Only DSDM provides a mechanism to assist with scaling to many small teams;
5. **Analysis and Evaluation**. There is a need for stronger focus on Analysis and Evaluation if traditional software engineering processes are to become more suitable for Web engineering;
6. **Requirements and Testing**. Requirements and Testing phases in traditional software engineering need to improve their focus on the issues facing Web application development;
7. **Maintenance**. Support for maintenance seems to increase the closer processes get to the agile community.

The following chapter describes in detail the Agile Web Engineering (AWE) Process that was developed specifically to address the criteria for a Web engineering process.

6 Agile Web Engineering (AWE) Process

This chapter provides an overview of the Agile Web Engineering (AWE) Process. AWE is a lightweight process that is differentiated from other agile and plan-driven processes by specific application to Web engineering and its particular focus on criteria 2, 4 and 5 for a Web engineering process, namely, 'Different business models', 'Small development teams working in parallel' and 'Analysis and Evaluation'. Section 6.1 discusses the rationale behind the creation of AWE and justification for the adoption of an agile approach for Web engineering. Section 6.2 presents a brief derivation of the AWE process. Sections 6.3-6.7 describe AWE and evaluate AWE's support for the criteria for a Web engineering process which is summarised in section 6.8. Section 6.9 presents a summary and the research contribution made by the material in this chapter.

6.1 Requirements for an Agile Process

The term agile has recently been used to categorise a number of lightweight process approaches to software engineering. These include: Extreme Programming (XP) (Beck 2000), the Crystal family of agile processes (Cockburn 2002) and Dynamic Systems Development Methodology (DSDM) (Stapleton 1997), although DSDM predated the 'agile' classification (Beck et al. 2001). Seventeen advocates and methodologists of agile processes convened in February 2001. The result of this meeting was the formation of the Agile Alliance and the production of The Manifesto for Agile Software Development (Beck et al. 2001).

The following quote from The Manifesto for Agile Software Development (Beck et al 2001), reproduced in Appendix 4, gives a summary of its purpose[5]:

> We are uncovering better ways of developing software by doing it and helping others do it.
>
> Through this work we have come to value:
>
> (i) Individuals and interactions over processes and tools;
> (ii) Working software over comprehensive documentation;
> (iii) Customer collaboration over contract negotiation;
> (iv) Responding to change over following a plan.
>
> That is, while we value the items on the right, we value the items on the left more.

The empirical evidence presented in chapter 3 suggests that the developers involved in Web engineering projects are one of the primary factors in the success or failure of Web application development. Thus, the agile route with its focus on people, point (i), lends itself strongly to Web-based application development. However, given the diversity of disciplines required to develop Web-based applications, the AWE Process, or any other Web-based process or methodology, can only hope to have a second order effect on project success.

[5] The author has added a numbering scheme (i)-(iv) to The Manifesto for Agile Software Development to aid the identification of specific statements.

Plan-driven processes are not well suited to addressing the problems associated with Web application development, as described in chapter 5. Many plan-driven processes attempt to codify good practice and experience in too much detail and for developers who do not understand the importance of what they are doing. This often results in development projects using plan-driven processes as cookbook recipes, where developers are lulled into a false sense of security by following the recipe in detail rather than using the ingredients selectively to help them build software deliverables that help address their problems (Hardy, Thompson & Edwards, 1995). Ultimately many development teams and organisations rarely use plan-driven processes as their designers intended (Barry & Lang 2001). Projects using plan-driven processes in such a manner often spend most of their resources producing volumes of documentation as opposed to focusing on delivering working software systems that address the problems the project is meant to be tackling. The AWE Process places the onus on developers and organisations involved in Web engineering to deliver Web-based solutions that satisfy their project goals. It is not that documentation or techniques are inconsequential. On the contrary, documentation, techniques and tools enhance and play a critical role in the success of many Web-based projects. It is just that the most important project deliverable is the Web-application itself, as stated in point (ii) of the manifesto.

Consider the impact exerted between the business, domain, software and creative design models in Web engineering, see section 4.2. The rate of change within these models and the impact change within one model has upon other models necessitates that the AWE Process supports close collaboration, point (iii), and the ability to adapt and respond to change effectively, point (iv). Every project needs some element of contractual agreements and guidelines to help ensure project success, whether written or verbal. However, it is collaboration and the ability to adapt to change that are more critical to project success. Thus, contractual arrangements need to be flexible enough to support the changes brought about through collaboration and the impact change exerts between the models involved in Web application development.

In addition to the previously stated purpose of the Manifesto for Agile Software Development, the document also includes twelve principles. The following quote lists the twelve principles[6] as stated in the Manifesto for Agile Software Development (Beck et al 2001):

> We follow the following principles:
>
> (A) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software;
>
> (B) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
>
> (C) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
>
> (D) Business people and developers work together daily throughout the project;
>
> (E) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done;

[6] The author has added identifying letters (A)-(L) to The Manifesto for Agile Software Development to aid the identification of specific statements.

(F) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;

(G) Working software is the primary measure of progress;

(H) Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely;

(I) Continuous attention to technical excellence and good design enhances agility;

(J) Simplicity - the art of maximizing the amount of work not done - is essential;

(K) The best architectures, requirements and designs emerge from self-organizing teams;

(L) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

This paragraph discusses AWE's support for points (A)-(L). AWE is an iterative and incremental process that helps to allow for: early and continuous delivery of valuable software (A); the ability to harness changing requirements, even late in development (B); and the delivery of working software frequently (C). The AWE Process supports multidisciplinary development treating business experts, domain experts, and creative designers as developers along-side software engineers and team leaders. AWE encourages developers to work in close proximity to each other on a daily basis, in an environment that supports productive development, supporting point (D). Close working proximity and informal (face-to-face) communication (F) between developers, primarily through the browser experience, should help support faster and more productive development. It is the responsibility of each developer, team and organisation to continually deliver Web applications that satisfy end-users (E) and ultimately AWE encourages this as the primary measure of project success (G). One of the primary aims of the AWE Process is to deliver working Web applications that satisfy end-user usage, and in doing so realise project objectives. Delivery of working software should be sustainable through AWE at a constant rate (H). Ultimately, it is the responsibility of every developer to make AWE work through integration of techniques and attention to simple (J), high quality designs (I) and implementations of Web-based solutions. AWE places the responsibility on the development team to influence the architecture, requirements and design of the proposed Web application (K). After each life-cycle iteration the development team is required to reflect upon and evaluate the development process (L).

AWE's adoption of the agile process approach as opposed to a plan-driven process approach helps to ensure greater support for time-to-market pressures. This suggests that AWE shows strong support for criterion 1 'Short development life-cycle times'.

6.2 Derivation of the AWE Process

Software engineering processes, both plan driven and agile, are created in a number of ways. The development of XP, derived by primarily by Beck (2000), grew from practical experience, observations of software engineers, including his father, during the development of many software systems. Others such as DSDM (Stapleton 1997) and the USD (Jacobson, Booch and Rumbaugh 1999) derived with influence both personal and from a consortium of individuals and organisations, across many different projects. All three of these process approaches benefited from practical experience, and peer review, and evolution to each respective process.

The Agile Web Engineering (AWE) process was derived during 2001 to specifically address the criteria for a Web engineering process, see chapters 3 and 4 for more detail. Having had over six years experience in the development of Web-based applications in the government, academic, entertainment and travel sectors, the author had encountered a number of software engineering approaches, both successful and unsuccessful, during the development of Web-based applications. Using this experience and with reference to the Web engineering, software engineering and other computing science literature, cited throughout this dissertation, the author developed the Agile Web Engineering (AWE) Process for the development of Web-based applications. The rest of this chapter describes AWE using the process description approach described in chapter 2.

6.3 AWE: Process Life-cycle and Phases

The AWE Process identifies all the phases that should be explicitly addressed during Web application development, see figure 13. The names for each phase: Requirements, Design, Implementation, Testing and Deployment, should look familiar from other processes and methodologies, in particular those from the plan-driven literature, however, similarities with plan-driven process approaches should end there. The only deliverable that is required to be produced from the AWE Process is the Web application itself. That is not to say that those adopting AWE will not benefit from the production of intermediate deliverables (e.g. documents, diagrams, prototypes etc); it is just that AWE does not impose them. The onus is on the organisation and the developers to find, integrate, evaluate and create techniques, if necessary, to support the phases outlined in Figure 13. See Section 6.7 for a detailed discussion of selecting, integrating, evaluating and developing techniques to support the AWE Process.

Recently the agile process Extreme Programming (XP) (Beck 2000) has gained a lot of attention. In XP, developers are encouraged to communicate through the project source code. This is achieved through pair programming and other XP practices, see section 5.3.3. Communication through source code on a small software engineering project is viable due to the nature of the deliverables (software components), the homogeneous nature of the developers and the small size of the team. Unlike software engineering, Web engineering results in deliverables that comprise data integrated with software. Web engineering does comprise small development teams, however these teams are multidisciplinary in nature and in the vast majority of cases will not be able to communicate through the source code of the Web application due to the heterogeneous nature of the developers. AWE places importance on understanding end-user usage of the proposed system, by encouraging all team members to communicate through the Web application *browser experience*. Even if a developer is working on the database in an n-tier Web application there will be issues relating to the browser experience that all developers must be involved in. For example, the database design may require the Web application to collect data from the user in a particular order. Should this order reveal poor end-user satisfaction with the browser experience then the database design should be changed to allow the Web application to focus more on satisfying end-user usage. See Section 6.5 for more details on collaboration and communication within AWE development teams.

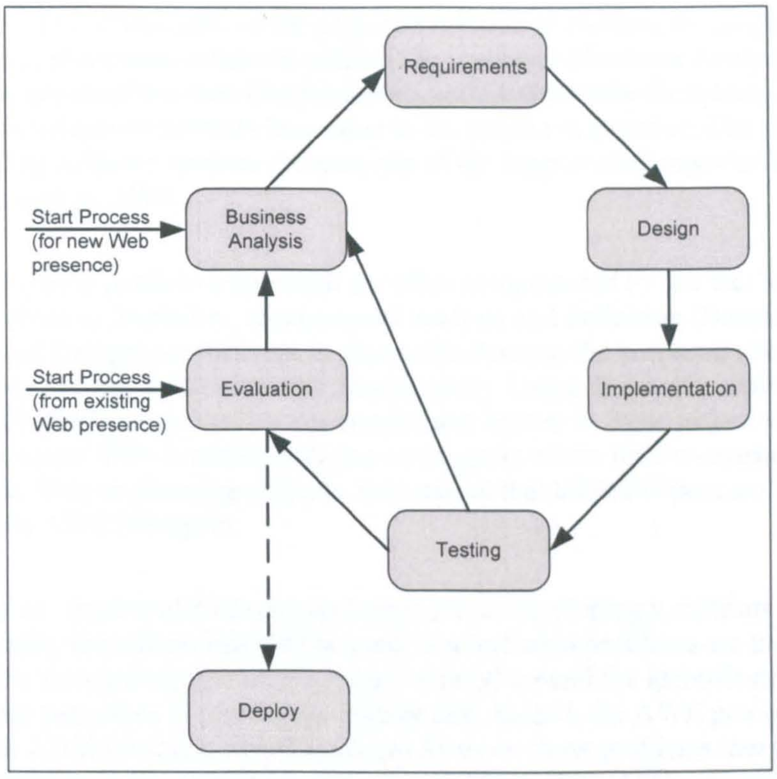

Requirements
Design
Implementation
Testing
Evaluation
Business Analysis
Deploy
Start Process (for new Web presence)
Start Process (from existing Web presence)

Figure 13. The AWE Process Life-cycle

In addition to communication through source code, XP advocates the use of metaphors to reflect architectural issues relating to project development. This indicates a relatively static and stable architectural environment. However, Web applications need to be highly scalable and portable, in addition to allowing easy replacement and upgrading of supporting software infrastructure (e.g. Web server, application server, database management system, etc.), if they are to ensure the longevity and growth expected by many of the different business models applied in Web engineering. The AWE Process therefore gives specific focus on architectural issues relating to Web application development.

Iterative and incremental development cycles are crucial to making the AWE process work successfully. Many Web development projects start out with a feasibility phase followed by a lengthy initial requirements analysis and definition phase. The predictive approach, where very early in the development life-cycle the problem and proposed solution are determined and remain largely static, is often useful from the perspective of describing project characteristics (e.g. budget, time-scales, resource allocation) that attempt to satisfy the types of initial demands often sought by senior stakeholders. However, this type of predictive approach often lulls organisations into the illusion that they understand everything that needs to be addressed by the proposed system at the beginning of the project. Usually on projects of this nature there follows the design and implementation of the proposed system, with little or no attention paid to re-addressing the problems (Business Analysis) and the definition of the proposed solutions (Requirements and Design) as the project stakeholders' understanding develops. Often, failings in the proposed solution are never properly identified until after the testing phase is completed, and even then, without a proper evaluation phase, end-user problems are often not noticed until the system is live in production.

The quicker a problem is identified and addressed during the life-span of a software system the more cost effective it is to address the problem (Boehm 1981). The longer development teams continue work on a project, the more their understanding grows about the problems and the deficiencies and strengths of the proposed solution in tackling the project's problem space. Ultimately, if the team cannot re-address the problems (Business Analysis) and the definition of the proposed solution (Requirements and Design) then the learning curve gained during development presents less value to the project in question. The predictive desire for building software systems presents one of the biggest challenges to the adoption of agile processes such as AWE.

The problems with the predictive approach are often compounded by the fact that those individuals involved in feasibility, requirements analysis and definition (Business Analysis, Requirements and Design) are different to those who develop the proposed solution (Implementation, Testing, Evaluation and Deployment). Using different people for project definition and development presents a communication barrier to those tasked with building the proposed solution. This is particularly true on projects where time-to-market pressures are high, such as Web engineering projects. It is crucial that all developers are involved in each phase of the AWE life-cycle.

When adopting an iterative and incremental approach to developing a software system it is essential to identify the criteria that will be used to select what problems are to be addressed first. Most of the selection criteria employed are centred around the identification of those problems that are perceived to present the highest risk. Indeed, the AWE process is no exception. Each AWE life-cycle iteration should focus on those problems that present the highest risk to project success should they not be achieved. Each AWE life-cycle iteration should focus on solving a subset of the problems that present the highest risk, ensuring that previous iteration's efforts are not compromised by the incremental increase in development scope. It is also prudent to remember that the simpler a solution is the easier it will be to change and enhance the overall proposed solution.

The customer community view in Web engineering projects make it essential to involve end-users in an evaluation phase before one can confidently determine whether the proposed solution will solve the problems identified in Business Analysis. Employing an evaluation phase with end-users can be expensive and time consuming (Constantine 2001). However, this does not mean that employing an evaluation phase with a representative sample of end-users is not cost effective. In addition, employing an evaluation phase on every iteration presents a serious conflict with the agile manifesto goal of addressing short development life-cycle times, and may not be practical or feasible due to the state of the deliverables. AWE therefore does not recommend an evaluation phase with end-users on each iteration of the AWE Process, see figure 14.

Evaluation should be used only when there is a substantial deliverable that can be measured against the identified problems and when the return on investment is seen to be worth the costs involved. AWE makes no predictions on how often the Evaluation phase should be included, with the exception of the last iteration before deployment, where it is essential to evaluate the deliverable in order to have confidence in the deployed solution. Figure 15 shows an iteration of the AWE life-cycle with the Evaluation phase included. It is the responsibility of the developers and organisations involved to judge how often to include the Evaluation phase.

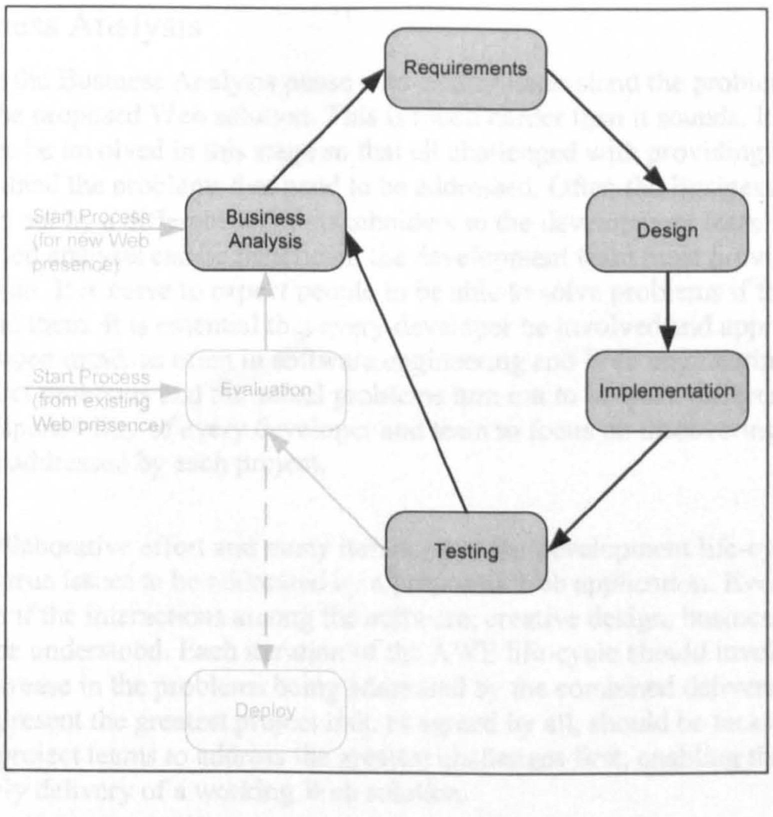

Requirements
Start Process (for new Web presence)
Business Analysis
Design
Start Process (from existing Web presence)
Evaluation
Implementation
Testing
Deploy

Figure 14. A Typical Iteration of the AWE Process Life-cycle.

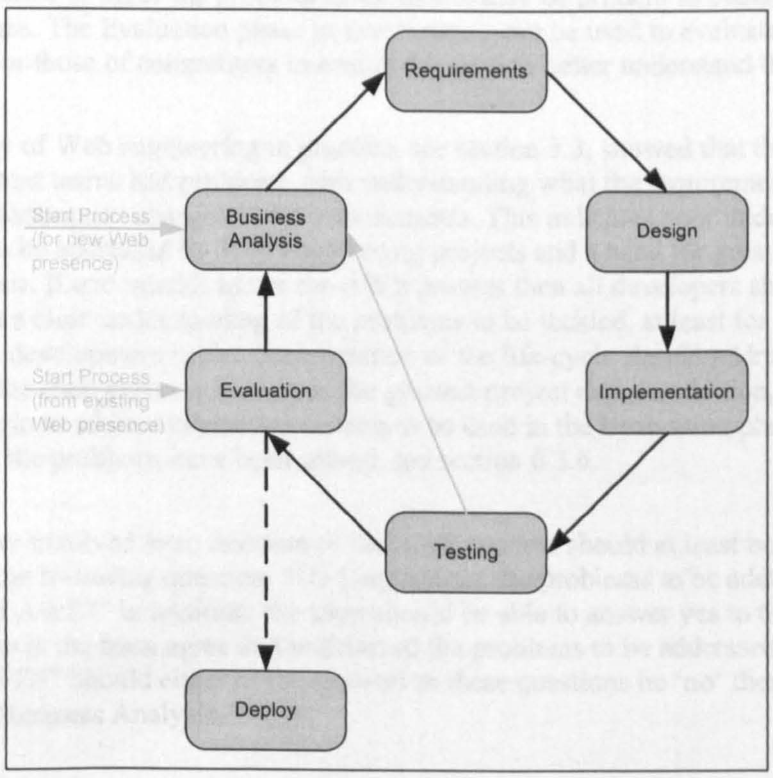

Requirements
Start Process (for new Web presence)
Business Analysis
Design
Start Process (from existing Web presence)
Evaluation
Implementation
Testing
Deploy

Figure 15. An Iteration of the AWE Process Life-cycle with an Evaluation Phase.

6.3.1 Business Analysis

The purpose of the Business Analysis phase is to clearly understand the problems to be addressed by the proposed Web solution. This is much harder than it sounds. It is crucial that every developer be involved in this stage so that all challenged with providing the proposed solution understand the problems that need to be addressed. Often the Business Analysis phase is carried out by a different set of stakeholders to the development team. While input from other skilled analysts can be beneficial, the development team must provide the proposed solution. It is naive to expect people to be able to solve problems if they do not fully understand them. It is essential that every developer be involved and approach this phase with an open mind, as often in software engineering and Web engineering the perceived project problems and the actual problems turn out to be quite different. AWE makes it the responsibility of every developer and team to focus on discovering the real problems to be addressed by each project.

It requires a collaborative effort and many iterations of the development life-cycle to understand the true issues to be addressed by a proposed Web application. Everyone must be willing to learn if the interactions among the software, creative design, business and domain models are to be understood. Each iteration of the AWE life-cycle should involve an incremental increase in the problems being addressed by the combined deliverables. The problems that present the greatest project risk, as agreed by all, should be tackled first. This will allow the project teams to address the greatest challenges first, enabling the adoption of change and early delivery of a working Web solution.

Most projects will enter the AWE process at the Business Analysis phase. If the team find it difficult to describe or agree the problem space then it may be prudent to backtrack to the Evaluation phase. The Evaluation phase in this instance can be used to evaluate an existing Web presence or those of competitors to enable the team to better understand the problems.

The first survey of Web engineering in practice, see section 3.3, showed that the majority of Web development teams had problems with understanding what the requirements were and in controlling continuous changes to the requirements. This indicates poor understanding of the problems to be addressed by Web engineering projects and a need for greater focus on business analysis. If one intends to use the AWE process then all developers should leave this phase with a clear understanding of the problems to be tackled, at least for the current iteration of the development cycle. Each iteration of the life-cycle should address the problems that the team perceive to present the greatest project risk. In addition, the team should also begin to address evaluation criteria to be used in the Evaluation phase to assess whether or not the problems have been solved, see section 6.3.6.

Every developer involved in an iteration of the AWE process should at least be able to answer yes to the following question: "Do I understand the problems to be addressed during this iteration of AWE?" In addition, the team should be able to answer yes to the question: "Does everyone in the team agree and understand the problems to be addressed during this iteration of AWE?" Should either of the answers to these questions be 'no' then more focus is required on Business Analysis.

6.3.2 Requirements

It is crucial that every member of the team agrees the problems to be solved before the requirements activities begin. The Requirements phase is about defining what the proposed solution will do (functional requirements), and what constraints are to be placed upon the proposed solution (non-functional requirements). In Web application development, there normally exist a large number of non-functional requirements that have to be properly addressed if the Web development process is to be successful. Great consideration should be given to addressing the non-functional requirements, in particular those associated with architecture, including those associated with the user interface and usability. For a more detailed discussion of architectural issues, see section 6.4.

After the requirements have been identified, the team should place focus on starting to create a test plan for use in the Test phase. The goal of this exercise is to enable the team to answer the question. Have we built the product right? It is essential that the entire team agree with the tests being devised to address this question. Otherwise, the project runs the risk of either falling short of agreed requirements or suffers from over engineering of the agreed requirements. See section 6.3.5 for a detailed discussion on the Test phase.

Every individual leaving the Requirements phase should be able to answer yes to the following question. Have we as a team defined the requirement(s) that need to be addressed during this iteration of the AWE Process life-cycle, if the perceived project objective with the highest risk is to be successfully addressed?

6.3.3 Design

Design involves understanding, co-ordinating and communicating all the major issues, before the implementation of a Web application. These issues should be independent of the lower-level implementation details. The AWE process when applied to large Web application development projects requires communication between similar types of developer in separate teams on the same Web presence. This communication serves to enable re-use, not only of deliverables but also of architectural framework. It is important to understand that if one requires a Web application to grow and mature quickly, and one wants to avoid serious portability and redesign problems, then great attention has to be paid to architectural concerns.

AWE does not impose any design techniques upon the development team. AWE puts the responsibility on every developer involved, to ensure that the team has an adequate grasp of the major higher-level issues associated with Web application development, in order to realise a potential solution. If the team feel that techniques help them address higher-level issues associated with Web application design, then they should use them. If they find that techniques are not helping then they should not use them. On large Web projects orthogonal uni-discipline developer communication and the coordination team are the mechanisms AWE employs to encourage usage of similar techniques across multiple teams and to resolve conflicts where problems occur. However, remember that the development team is responsible for the design of their Web application. The AWE process is ultimately responsible for making developers aware of the major phases they need to address during Web application development.

6.3.4 Implementation

AWE considers implementation, or design at the lowest level (Cockburn 2000), to be just as important as high-level design. While there are many differences between the Design and Implementation phases, both involve decisions that have a critical impact on project success. The agile process XP encourages developers in teams, mostly comprising software engineers, to communicate through the source code of the project deliverable. This is achieved through pair programming and extensive testing. Unfortunately, while communicating through source code may be suitable, such as HTML, for those representing roles from the software model and the creative design model (homogeneous communication), it is not practical for communication amongst all developers in a multidisciplinary team (heterogeneous communication). Instead, AWE recommends communicating through the browser experience. All developers should collaborate and focus their development efforts around the Web interface, using collaborative sessions to discuss and review the browser experience. Often it is beneficial to record the results of reviewing the browser experience, however this is left at the discretion of the teams, individuals and organisations adopting AWE. The browser experience should help teams focus on the issues that are crucial to end-users.

6.3.5 Testing

Testing is a crucial stage in any software activity. It is essential that all developers are aware of the objectives of the Testing phase. Testing objectives involve assessing whether or not what has been built has satisfied the project's requirements. All the developers should be asking the question. Have we built the product right?

Beyond just testing the functional requirements, it is essential that all developers understand the importance of testing non-functional requirements. The development team, should, as a minimum, address the following questions during non-functional requirements testing. Is our application compatible with our target browser audience and the constraints of our audience's environment (screen size, bandwidth, processor speed, memory, etc)? Can our application cope with the expected load? Will our system scale easily? Is our system portable? Does our system conform to the required performance constraints? Is our system secure?

Testing requires input from the Requirements, Design and Implementation phases. The requirements phase contributes very high-level tests, for example: given input X the system will respond with output Y; or the system will be able to cope with 400 concurrent users at one time. The design and implementation phases contribute tests and information for tests at a much lower level, for example, if $Z > 10$ during transaction type A, then the system will queue T for R seconds. Figure 16 highlights the phases involved in the creation of the Test Plan.

Developers should be aware that a good browser experience on their desktop does not guarantee a good browser experience on the desktop of their intended audience. Even if developers thoroughly test their deliverables during Design and Implementation it is essential that the entire team are involved in testing the deliverables in an environment that is as close as possible to their intended audiences. It is good practice to ensure that the development team have access to a minimally configured system (minimum browser specification, bandwidth connection, screen size, etc) for testing.

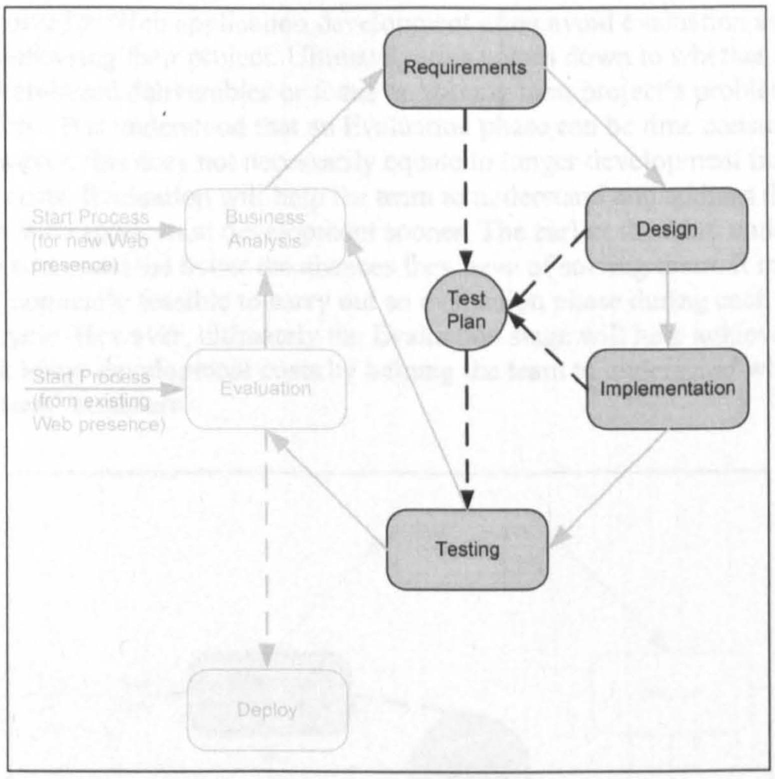


Figure 16. AWE Process Life-cycle showing the Phases Involved in the Creation of a Test Plan

6.3.6 Evaluation

The Evaluation Plan should be derived from the issues or problems identified in the Business Analysis phase. See figure 17 for a description of the phases involved in the creation of the Evaluation Plan. Essentially, the team must address the Evaluation phase by asking themselves the following question. Have we built the right product? It is imperative that the team objectively evaluate what has been delivered independently of design and implementation issues. Only then will everyone be able to assess whether or not the project is solving the problems to be addressed. Evaluation, when carried out by skilled developers measuring end-user usage, often leads to a greater understanding of the problems or issues to be solved. This new understanding of the problem space needs to be fed back into the Business Analysis and Requirements phases, see figure 17.

Understanding what is involved in carrying out an evaluation of Web-based applications is an area in its own right. There are many good sources of information to guide and assist a development team with this task (Lewis & Rieman 1994, Constantine & Lockwood 1999, Nielsen 2000, Spool et al. 2002 and Nielsen & Norman 2004). The intention of this sub-section is to make the reader aware of some of the issues involved in evaluating a Web application, not to present an exhaustive guide to doing so.

It is imperative that a representative sample of end-users is used to evaluate the deliverables. While end-user substitutes are a valuable resource, there is no replacement for evaluation with a representative sample of end-users. It is also important that the developers understand how to interact with end-users during evaluation, just asking them what they think or want is known to be flawed (Nielsen 2001a). It is imperative to observe usage of the deliverables by end-users in a situation that mimics their normal usage environment.

Developers involved in Web application development often avoid evaluation as they see it as an obstacle to delivering their project. Ultimately, this comes down to whether the team wish to just create Web-based deliverables or focus on solving their project's problems with Web-based applications. It is understood that an Evaluation phase can be time consuming and expensive. However, this does not necessarily equate to longer development times and greater project costs. Evaluation will help the team to understand and address the issues associated with Web application development sooner. The earlier the team understand the problems to be addressed the better the chances they have of solving them. It may not be practical or economically feasible to carry out an evaluation phase during each iteration of the AWE life-cycle. However, ultimately the Evaluation stage will help achieve shorter life-cycle times and lower-development costs by helping the team to understand what their project's end-users' needs are.

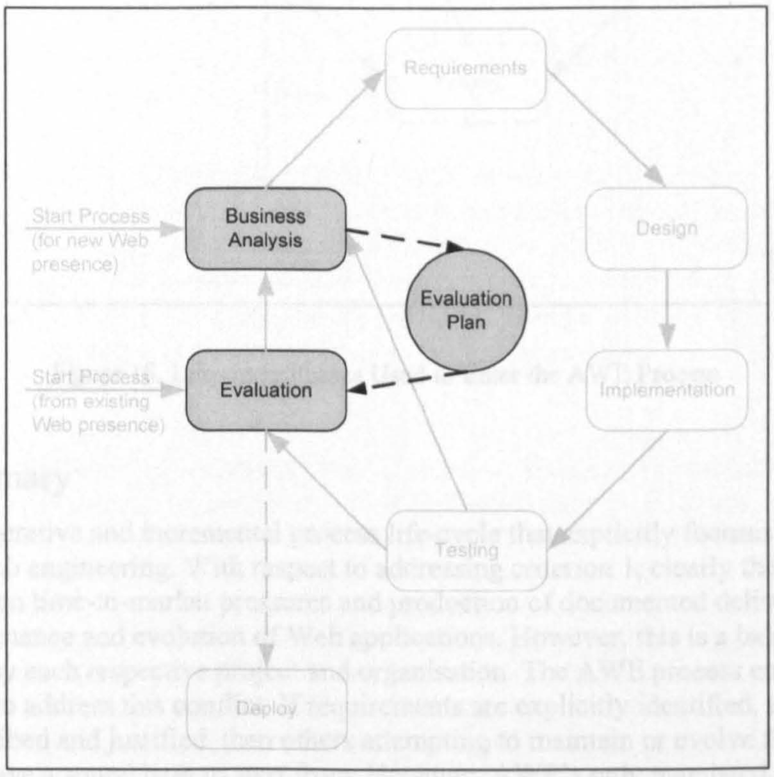


Figure 17. AWE Process Life-cycle showing the Phases Involved in the Creation of an Evaluation Plan

6.3.7 Beginning the AWE Process

There are two phases that can be used to start the AWE Process life-cycle: the Business Analysis phase and the Evaluation phase. When starting a new Web-based endeavour then you should enter the AWE Process life-cycle through the Business Analysis phase. However, it should be noted that greater problem understanding may be gained through Evaluation of competitors' Web applications or other related Web-based solutions. When evolving an existing Web application or presence then it is prudent to start the AWE Process life-cycle through the Evaluation stage where the existing Web application or Web presence is evaluated using previous project goals. Entering the AWE Process through the Evaluation phase when evolving an existing Web application will help tremendously with the Business Analysis phase. Figure 18 shows the entry phases to the AWE Process life-cycle.

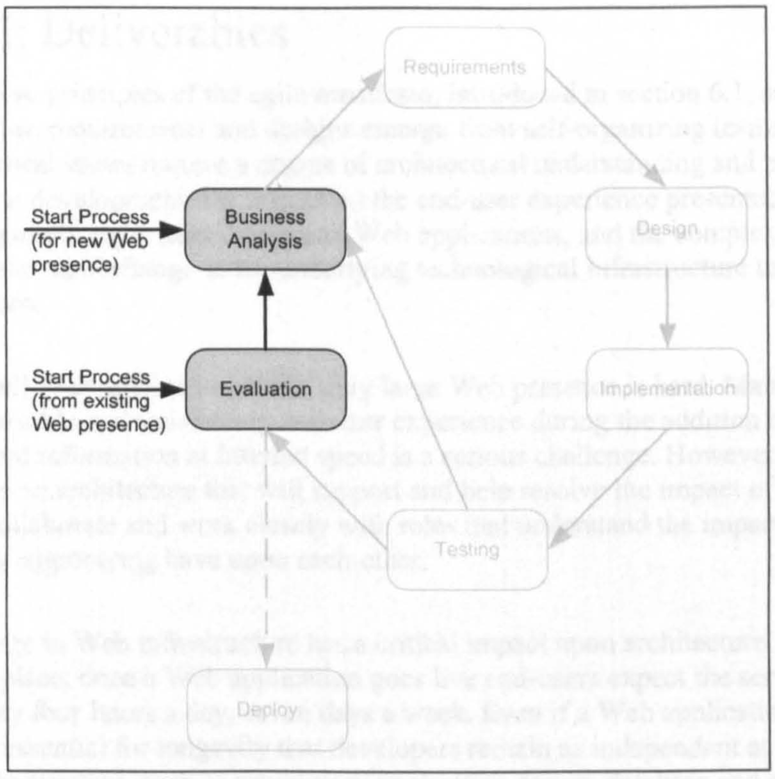

Requirements
Start Process (for new Web presence)
Business Analysis
Design
Start Process (from existing Web presence)
Evaluation
Implementation
Testing
Deploy

Figure 18. Life-cycle Phases Used to Enter the AWE Process

6.3.8 Summary

AWE has an iterative and incremental process life-cycle that explicitly focuses on the phases required in Web engineering. With respect to addressing criterion 1, clearly there is a conflict between time-to-market pressures and production of documented deliverables to support maintenance and evolution of Web applications. However, this is a balance that must be addressed by each respective project and organisation. The AWE process encourages those using it to address this conflict. If requirements are explicitly identified, and designs properly described and justified, then others attempting to maintain or evolve the deliverables have a sound base to start from. However, AWE's only mandated deliverable is the Web application itself. AWE is therefore ranked as showing strong support for criterion 1 'Short development life-cycle times'.

AWE explicitly focuses on the importance of the business analysis phase within the AWE life-cycle with a discussion of the objectives and potential activities carried out within this phase. AWE also focuses on an evaluation phase with a representative sample of end-users and encourages explicit focus on the importance of end-user involvement and the usability during architectural considerations. AWE is therefore ranked as showing strong support for criterion 5 'Analysis and Evaluation'.

AWE places explicit focus on the objectives needed to be addressed in both the requirements and testing phase during the AWE process. AWE is therefore ranked as showing strong support for criterion 6 'Requirements and Testing'.

6.4 AWE: Deliverables

One of the twelve principles of the agile manifesto, introduced in section 6.1, states, "The best architectures, requirements and designs emerge from self-organizing teams" (Beck et al. 2001). Two critical issues require a degree of architectural understanding and planning in Web application development: the impact on the end-user experience presented by the rapid evolution and growth experienced on many Web applications, and the complexity involved in adapting to the rapid change in the underlying technological infrastructure used to present Web applications.

Evolving a small Web application into a truly large Web presence is hard. Maintaining a consistent, enjoyable and satisfactory end-user experience during the addition of functionality and information at Internet speed is a serious challenge. However, in order to properly devise an architecture that will support and help resolve the impact of these issues, one needs to collaborate and work closely with roles that understand the impact each of the models in Web engineering have upon each other.

The rapid change in Web infrastructure has a critical impact upon architecture. The Web is a global market place; once a Web application goes live end-users expect the service to be available twenty four hours a day, seven days a week. Even if a Web application evolves at a slow rate, it is essential for longevity that developers remain as independent as possible from supporting infrastructure, such as proprietary application server, database and operating system features. Failure to do so may result in significant redevelopment efforts to ensure the continued usefulness of a Web-based system.

Calvert, Smith and Natis (2001) classify Web applications into one of two categories, opportunistic or systematic. Opportunistic Web applications are small, less significant, Web-based solutions that have little potential for expansion compared with systematic Web applications. An example of an opportunistic Web application would be a conference Web site, where structure and content may change, but at such a rate not to require systematic architectural planning continuously throughout the life time of the Web site. This does not mean that opportunistic Web applications can ignore the architectural issues, just that the significance of architectural issues is lessened in comparison with systematic Web-based systems. An example of a systematic Web application would be an Internet banking application, where underlying infrastructure will have to expand to support the exponential growth rate of users, functionality and information. In addition underlying infrastructure will in the vast majority of cases eventually be replaced to ensure lower operational costs as infrastructure support contracts near their end of life and to take advantage of evolution in technology.

The rest of this section goes on to discuss architectural issues within the context of AWE, outlining what development teams should address to help ensure the longevity, scalability, portability, reuse, and ultimately a satisfactory end-user experience of their Web-based deliverables.

6.4.1 Web Usability and the End-User Experience

Many Web sites have evolved into extremely large presences over the past decade (Nielsen 2000a). The exponential growth of information and functionality in many Web applications makes the task of managing each end-user experience extremely difficult. Ensuring that information is easy and intuitive to ascertain, and that applications provide functionality that satisfies end-users usage demands is something that requires a good deal of thought and effort particularly on large systematic Web application development projects. Developers must be aware of the importance of creating a user-friendly experience for every type of end-user of their Web application.

Understanding end-users' usage of Web Applications is crucial to successfully realising the potential of Web-based endeavours. A study by Nielsen (2001b) calculated that as much as 44% of potential e-commerce sales are not completed because users cannot use the site. In addition, Nielsen goes on to describe the lack of adherence to usability guidelines (Nielsen, Molich, Snyder & Farrell 2000), stating that most Web sites comply with only one third of published usability guidelines. The report suggests that if usability was improved on the average e-commerce site, there is a potential to increase sales by up to 76%.

It is important to get productive involvement with end-users in order to successfully adopt the AWE Process. It is only possible to truly achieve the potential of large Web-based solutions if the developers are able to understand and satisfy end-users' usage requirements. The Human Computer Interaction (HCI) field has produced a large collection of published literature, including Nielsen (2000a) and Spool et al. (2002), which can assist teams with the challenges of dealing with Web usability issues. However, HCI is a large field in its own right, and an exhaustive review of the literature is beyond the scope of the AWE process and this dissertation. AWE places the responsibility on the individual, team, and organisation to tackle the Web usability challenges presented by end-users, who are ultimately, the litmus test for their Web engineering project success.

End-users, if engaged and interacted with correctly (Nielsen 2001a), can contribute enormous benefits to Web engineering projects. However, it is important to bear in mind the costs involved in dealing with end-users. End-user validation is time consuming and can be expensive (Constantine 2001). Only a thorough understanding and a commitment from everyone as to the importance of end-user involvement will enable development teams to harness the true benefits of Web-based developments.

6.4.2 Web-based Software Infrastructure

Recently there has been a move towards Web-based enterprise development frameworks in the enterprise application development community. Examples include Sun Microsystems's Java 2 Platform Enterprise Edition (J2EE) platform (Sun 1999, Sun 2001a, Sun 2001b) and Microsoft's .NET framework (Microsoft 2001a, Microsoft 2001b). One of the major objectives of these initiatives is to provide a mechanism that will allow infrastructure independence from hardware, operating systems (excluding .NET) and to some degree Web-based software infrastructure (e.g. database, application server, Web server). If strictly adhered to, these enterprise frameworks can help ensure: portability; scalability; lower total cost of ownership; easier skills transfer amongst developers; vendor independence; and platform and software infrastructure independence. It should be noted, though, that to develop Web-based solutions that have these characteristics, other standards initiatives usually have to complement those of enterprise framework adoption. Mark-up language standards, e.g. XML, and client-side browser scripting technologies, e.g. permitting the use of Java Applets only, can be critical in ensuring the greatest possible browser compatibility, which ultimately allows one to deliver the solution to a wider target browser audience. In addition, Structured Query Language standards e.g. SQL92, can help ensure database flavour independence.

If the goal is to produce opportunistic Web applications then architectural issues are less of a concern, however, ignoring them altogether is extremely risky. For those who are embarking upon, or are involved in, systematic Web application development, these issues are critical to project success and tackling them effectively will help ensure that one can harness the benefits of architectural consideration and planning. It is not the author's objective to get into a religious debate regarding J2EE and .NET. The AWE Process will work with both these enterprise frameworks, although the techniques chosen to support AWE may vary depending on the enterprise framework. Figure 19 illustrates a 4-tier Web-based architecture, indicating the role of enterprise frameworks and guidance to help identify some of the previously discussed issues. It should be noted that the architecture described in figure 19 is a utopian one that is difficult to achieve in the enterprise. Often there are many legacy systems and multiple database flavours to contend with in addition to external systems that are often interfaced through messaging systems. Organisation and development teams are responsible for their architecture. Organisations and developers will get from their architecture what they put into it, as the META Group (2001) saying goes "An enterprise without architecture is like a ship without a rudder - it will move but you cannot guide it".

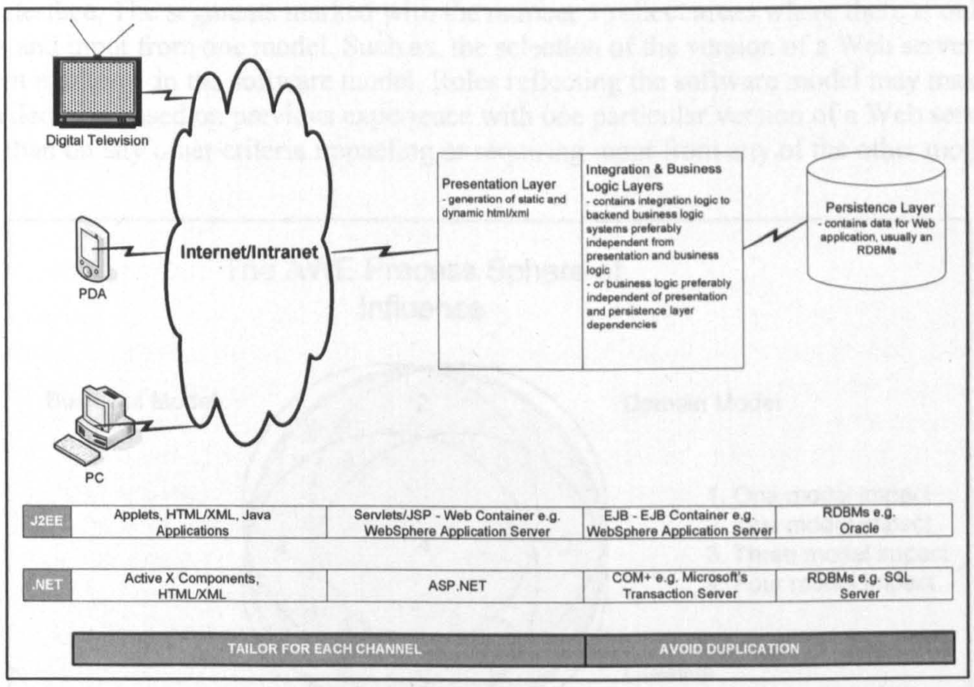

Digital Television
PDA
PC
Internet/Intranet
Presentation Layer
- generation of static and dynamic html/xml
Integration & Business Logic Layers
- contains integration logic to backend business logic systems preferably independent from presentation and business logic
- or business logic preferably independent of presentation and persistence layer dependencies
Persistence Layer
- contains data for Web application, usually an RDBMs
J2EE
Applets, HTML/XML, Java Applications
Servlets/JSP - Web Container e.g. WebSphere Application Server
EJB - EJB Container e.g. WebSphere Application Server
RDBMs e.g. Oracle
.NET
Active X Components, HTML/XML
ASP.NET
COM+ e.g. Microsoft's Transaction Server
RDBMs e.g. SQL Server
TAILOR FOR EACH CHANNEL
AVOID DUPLICATION

Figure 19. N-tier Architecture showing Placement of Web-based Software Infrastructure Components with Reference to J2EE and .NET

AWE explicitly addresses the architectural issues associated with Web engineering that are important to evolve and maintain Web applications. AWE is therefore ranked as showing strong support for criterion 7 'Maintenance'.

6.5 AWE: Activities

In the AWE Process Sphere of Influence, see Figure 20, all model areas reflected in the grey circle (AWE Process Sphere of Influence) impact upon and are impacted by the AWE Process. The areas marked 1 to 4 respectively indicate the number of developer roles and models that influence, and are impacted by, the AWE Process. Team leaders, see section 6.6.7, are the exception in that this role can influence and have an impact in all the numbered segments. There are very few segments in the AWE Process Sphere of Influence that do not impact two or more models. Thus, if the AWE Process is to be successfully adopted then there is a clear need to explicitly encourage communication and collaboration between the various models reflected in the AWE Process Sphere of Influence, segments marked 1 to 4.

Within the AWE Process Sphere of Influence the segment marked with the number 4 represents part of the development process that impacts all four models and requires input from all four models. There are a number of different issues and views to be addressed by the problems that are categorised in this segment; the resolution to these problems can have the most radical impact on all of the four models in Web engineering. The segments in the AWE Process Sphere of Influence marked with the number 3 represents the part of the AWE development process that impacts three models and requires input from the three models impacted. For example, the segment that is created by the intersection of the software, creative design and business models may produce scenarios where roles representing the software, creative design and business models are required to develop elements of the Web site structure. The segments marked with the number 2, reflect areas impacting and requiring input from two models. For example, the segment between the software and the creative design model may reflect issues to be resolved between usability and aesthetic design of the Web interface. The segments marked with the number 1 reflect areas where there is only impact and input from one model. Such as, the selection of the version of a Web server in the segment marked 1 in the software model. Roles reflecting the software model may make such a decision based on previous experience with one particular version of a Web server rather than on any other criteria impacting or requiring input from any of the other models.

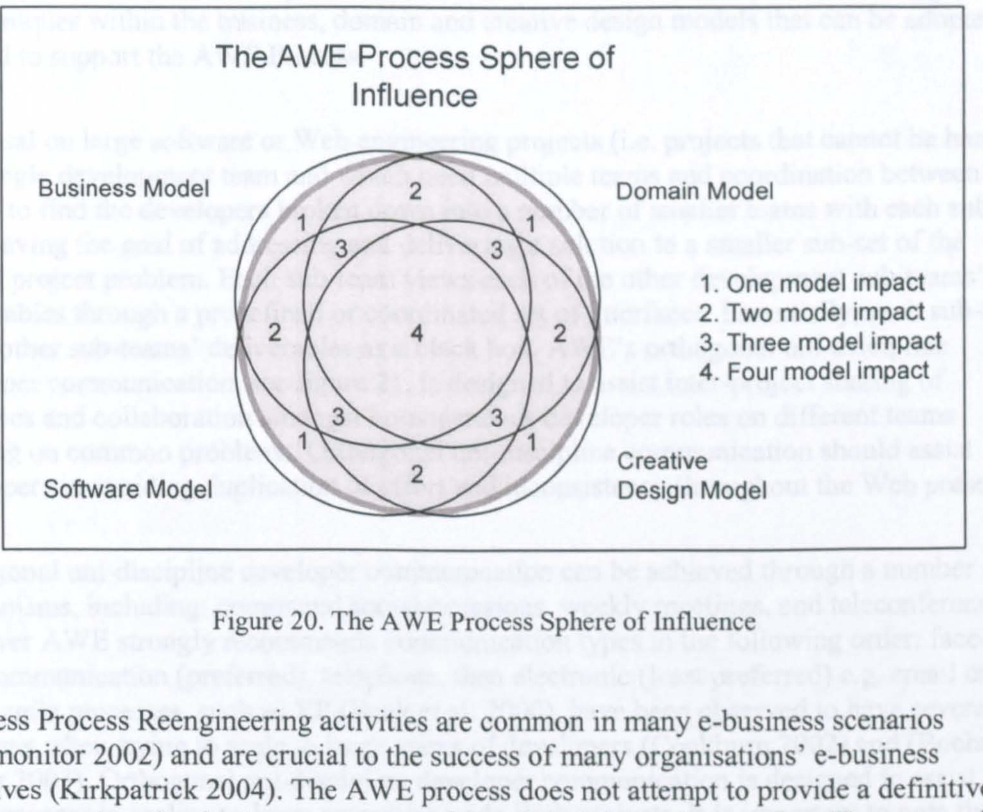


Figure 20. The AWE Process Sphere of Influence

Business Process Reengineering activities are common in many e-business scenarios (Datamonitor 2002) and are crucial to the success of many organisations' e-business initiatives (Kirkpatrick 2004). The AWE process does not attempt to provide a definitive guide to reengineering business processes, this task is left to others such as Hammer and Champy (2001). Instead, AWE, through its agile process approach, with its focus on people and their interactions, encourage developers, both technical and non technical, to give explicit focus on business process reengineering activities to benefit their project. This is illustrated by the AWE process sphere of influence, see figure 20, that encompasses the majority of the business, domain and creative design activities as well as the software focused activities found in traditional software engineering processes.

AWE identifies the models required in Web engineering e.g. the business, domain, software and creative design models. In addition, AWE defines the respective roles reflected from the models in Web engineering (refer to section 6.6 for a detailed discussion of AWE roles) and discusses how these roles should interact to ensure business and domain models embrace the impacts and benefits of the different business models found in Web engineering. AWE is therefore ranked as showing strong support for criterion 2 'Different business models'.

6.5.1 Multidisciplinary and Orthogonal Uni-Discipline Developer Communication

The multidisciplinary or heterogeneous nature of Web development prohibits the use of many existing software engineering techniques for multidisciplinary or intra-team communication due to their technical foundations and homogeneous nature. However, this is not to say that AWE discourages the use of existing software engineering techniques, quite the contrary. The software engineering community has spent many years striving to tackle problems in software development, many of these activities have produced well proven techniques that support common tasks in the software model in Web engineering. However, most of the software engineering techniques will only prove to be useful during uni-disciplinary communication sessions within the software model (homogeneous communication). This still leaves the problem of supporting multidisciplinary communication (heterogeneous communication) and other uni-discipline communication sessions (homogeneous communication) in the creative design, business and domain models. With respect to other uni-discipline communication mechanisms, there may exist approaches or techniques within the business, domain and creative design models that can be adopted or tailored to support the AWE Process.

It is usual on large software or Web engineering projects (i.e. projects that cannot be handled by a single development team and which need multiple teams and coordination between the teams) to find the developers broken down into a number of smaller teams with each sub-team having the goal of addressing and delivering a solution to a smaller sub-set of the overall project problem. Each sub-team views each of the other development sub-teams' deliverables through a predefined or coordinated set of interfaces. Essentially, each sub-team views other sub-teams' deliverables as a black box. AWE's orthogonal uni-discipline developer communication, see figure 21, is designed to assist inter-project sharing of resources and collaboration amongst homogeneous developer roles on different teams working on common problems. Orthogonal uni-discipline communication should assist developers in avoiding duplication of effort and inconsistency throughout the Web presence.

Orthogonal uni-discipline developer communication can be achieved through a number of mechanisms, including: communal social occasions, weekly meetings, and teleconferences. However AWE strongly recommends communication types in the following order: face-to-face communication (preferred), telephone, then electronic (least preferred) e.g. email or fax. Many agile processes, such as XP (Beck et al. 2000), have been observed to have severe problems when trying to scale to large teams of developers (Cockburn 2002) and (Boehm & Turner 2003). Orthogonal uni-discipline developer communication is designed to assist the AWE process in scaling to large enterprise wide Web projects. It is important to note that Figure 21 also includes a co-ordination team. The co-ordination team is responsible for guiding the inter-team communication process and for leading the inter-team initiatives that will enhance the objectives of the larger Web presence. For example, consider the banking organisation involved in developing the Web-based teller system. Since the Web-based teller system is intended to be one of many Intranet systems being developed to run on branch computers, it is going to be essential that there is orthogonal uni-discipline communication between similar development roles working on different Intranet projects. For instance, all programmers decide that they will meet once a week for lunch to discuss work related issues. During a lunch meeting it is discovered that certain types of users may be required to remember more than ten separate user names and passwords for different Intranet applications. Upon realisation of the negative impact this problem will have on the end-user experience, the co-ordination team members decide to raise this issue with senior management to try and co-ordinate a single sign on initiative. The single sign on initiative is sold as a cost and effort prevention initiative to team leaders and as a mechanism for improving the end-users' usage of the proposed Intranet systems to each of the clients of the respective Intranet projects.

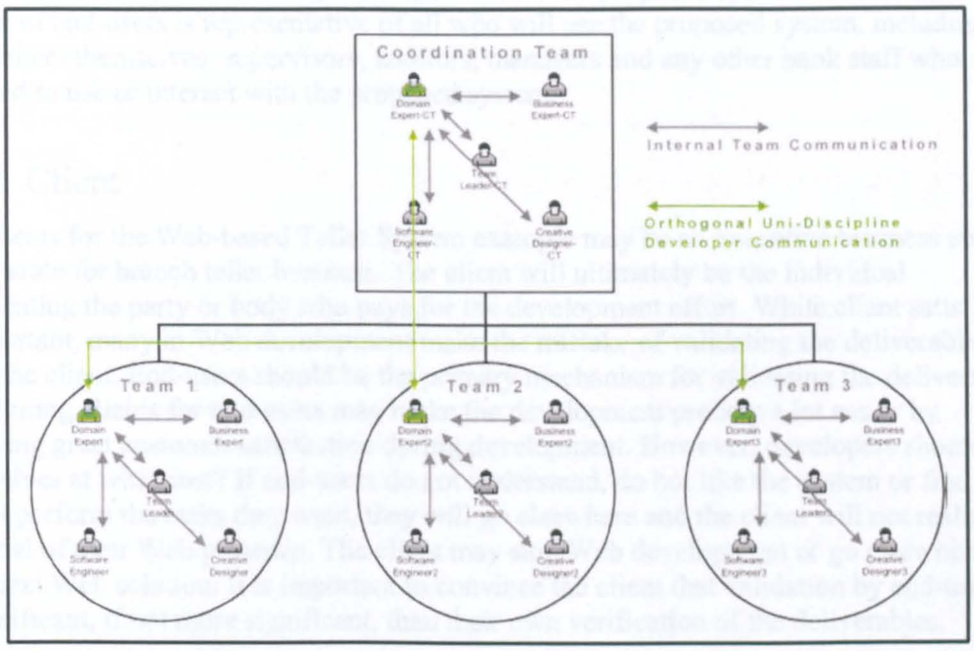


Figure 21. AWE Communication Channels on Large Web Projects

AWE's orthogonal uni-discipline developer communication model provides explicit support for scaling AWE to large Web engineering projects. AWE is therefore ranked as showing strong support for criterion 4 'Small development teams working in parallel on similar tasks'.

6.6 AWE: Roles

Within software processes there are a number of stakeholder roles that reflect different viewpoints or interests within the development framework, whether these are implicitly or explicitly acknowledged. Each of the roles represented below reflects an important viewpoint onto the AWE development process. There can be multiple stakeholders in each role. However, it is important that each role be filled by stakeholder(s) who have the ability to competently reflect the issues associated with each viewpoint.

6.6.1 End-User

End-users are the litmus test for success. The empirical evidence presented in chapter 3 indicates that many Web development efforts ignore or pay little attention to end-users. While many disciplines are involved in Web engineering it should never be forgotten that the end-users of the system ultimately decide the usefulness of the deliverables and therefore the primary successes of the project.

The AWE Process advocates close involvement with end-users, both in the Business Analysis stage to help developers understand the problem, and in the Evaluation stage to validate the deliverables against the problems or goals previously identified in the Business Analysis phase. Many Web projects only use clients to validate the deliverables. While end-user substitutes can be beneficial, substituting clients for end-users will not in the majority of cases help developers understand usage of the proposed system.

Consider our example Web-based Teller System. The end-users should be made up of a representative sample of banking staff who will use the proposed system. It is crucial that the sample of end-users is representative of all who will use the proposed system, including: bank tellers themselves, supervisors, auditors, managers and any other bank staff who will be required to use or interact with the proposed system.

6.6.2 Client

The clients for the Web-based Teller System example may be an executive business sponsor responsible for branch teller business. The client will ultimately be the individual representing the party or body who pays for the development effort. While client satisfaction is important, many in Web development make the mistake of validating the deliverables using the client. End-users should be the primary mechanism for validating the deliverables. Substituting clients for end-users may make the development process a lot easier by achieving great customer satisfaction during development. However, developers should ask themselves at what cost? If end-users do not understand, do not like the system or find they cannot perform the tasks they want, they will go elsewhere and the client will not realise the potential of their Web presence. The client may stop Web development or go elsewhere for their next Web solution. It is important to convince the client that validation by end-users is as significant, if not more significant, than their own verification of the deliverables.

6.6.3 Domain Expert

Domain experts provide data or content for Web applications. Web-based systems can present information relating to wheels for motor vehicles or information relating to a museum collection in the numismatics field. Unfortunately, domain experts cannot just create data and send it to the software engineering and creative design developers. All developers must collaborate to create an end-user experience that tackles the issues raised in the Business Analysis phase. This is difficult. It requires collaboration, compromise and a lot of hard work.

It should be noted that the majority of domain experts reside out with contracting organisations, and are most frequently found in different departments to the core Web development team, within in-house organisations. In the Web-based Teller System a domain expert may be a branch supervisor or deputy manager, who understands the teller activities and the activities associated with those of the teller, and the context within which these activities are carried out.

6.6.4 Business Expert

In addition to providing content, business experts are seen to provide guidance on achieving the business objectives of the Web application and are more involved in contributing to the overall structure of the Web presence. Business experts should provide a pivotal role in keeping the project focused on business objectives. Business experts also need to be able to approve the majority of decisions regarding the shift in focus many projects have to address as the project evolves.

It should be noted that the majority of business experts also reside out with contracting organisations, and are most frequently found in different departments to the core Web development team, within in-house organisations. The business expert in our Web-based

Teller Example would be a representative from head office who is able to make decisions regarding business objectives relating to the proposed solution and any associated Business Process Re-Engineering activities that need to be addressed.

6.6.5 Software Engineer

The software engineer is a role used to describe developers from a technical background who are responsible for development issues in the software model. These roles can usually be broken down into a number of sub-roles such as database developer and application programmer.

Software engineers not only have to be able to listen and solve the issues facing the business and domain models with software systems and software infrastructure, but they also have to advise on the impact Web and Internet technologies will have within the business and domain. For example, putting an email address on a Web site will only be effective if there is someone who is going to read and be able to effectively respond to the message. Often organisations do not have a culture of reading or being able to respond to the end-user's satisfaction via email. The software engineer must effectively communicate these issues to those concerned in the organisations in question. The software engineers involved in the Web-based Teller System will primarily comprise database developers and middleware programmers.

6.6.6 Creative Designer

Creative designers are from an artistic or graphic design background and are involved in creating the corporate design of the Web presence, and helping to brand the look and feel of the Web application. Often creative designers, due to their artistic background, lose sight of usability issues in favour of a more artistically creative design. All developers must be aware that what is being developed is a Web-based tool. While artistically enhanced Web presences can be a powerful asset in gaining early client satisfaction one should never forget that end-users will determine the success of the project. If end-users cannot perform the tasks they are trying to accomplish, with ease and satisfaction, they will not be happy. When building an Intranet site end-users may have no option but to use the system or go elsewhere for employment, but poor user interface design costs many organisations time and money. When developing an Internet application then the problem is compounded by the fact that your end-users are only seconds away from your competitors. Finding a balance between usability and creative design is a difficult task, but one that the AWE development team must address with the issues of the project and organisations involved, or risk the success of the project itself.

In our Web-based Teller System example the creative designer will be a Web developer primarily responsible for aesthetic aspects of the browser experience. It is essential that the creative designer is aware of the issues associated with building the user interface and can collaborate with the rest of the team to find the balance between function and form to achieve the most user friendly system given the project constraints. This will involve close collaboration with others in the AWE development team and a clear understanding of the usage of the proposed system through close observation of end-users.

6.6.7 Team Leader

The team leader is responsible for coordinating and directing the AWE development team towards the best solution available given the project constraints and characteristics. With such a wide diversity of developers to guide, being team leader is one of the most difficult jobs. Realistically they can only hope to be experienced in one of the other four developer roles: business expert; domain expert; creative designer or software engineer. With such a wide diversity of knowledge required to build most Web applications the team leader must respect all developers involved and appreciate the contribution that each different type of developer brings to the project. Where necessary though the team leader must ensure that the team make decisions regarding the development of the proposed system with the best interests of the project in question.

AWE shows encouragement of collaboration through the browser experience as a mechanism for multidisciplinary developers to communicate, and it identifies and discusses the roles required to represent the multidisciplinary nature of Web engineering. These roles include end-users, clients, business experts, domain experts, software engineers, creative designers and team leaders. AWE is therefore ranked as showing strong support for criterion 3 'Multidisciplinary development teams'.

6.7 AWE: Tools and Techniques

The AWE Process is technique and tool independent. This is not to say that AWE considers techniques unimportant. On the contrary, techniques are crucial to understanding many Web engineering project problems and realising Web-based solutions to those problems. However, many have researched and developed techniques that can support different tasks within Web application development. See chapters 18-20 in Pressman (2004) for an overview of techniques to support many of the phases in Web engineering. Techniques can be useful and powerful aids on many Web development projects. However, not all teams, organisations and projects may be suited to one particular technique. This is the case due to the wide diversity of projects and individuals involved in Web application development, and not necessarily due to any flaw in the techniques themselves. Instead, the AWE Process places the onus on the development team to discover, adopt, create and evaluate techniques to support the AWE Process where they feel they are required.

There are three primary purposes for techniques in any development process: to assist with understanding of the problem or proposed solution; as an aid to communication amongst stakeholders during project development; and as a mechanism to assist maintenance and ensure the evolution of the proposed system. The rest of this section goes on to discuss the issues that are important to consider when integrating techniques to support the AWE process and concludes with a guide to assist with integrating techniques to support AWE.

The diversity of solutions to which Web engineering is applied and the added complexity by the inclusion of the business, domain and creative design models in addition to the software model in Web engineering are the primary reasons for placing the onus on the development team who have to take on the responsibility of discovering, adopting, creating and evaluating techniques to support AWE. It is crucial that the AWE developers understand the responsibility that AWE places on their shoulders with respect to technique integration. Successful adoption of AWE without this understanding will only be applicable to small, opportunistic Web engineering projects. It is important that developers are aware that if their project shows any of the following characteristics: team members not in close working proximity to each other; development teams that are not going to maintain the project deliverables; numbers of developers greater than ten, then serious effort will have to be given to getting techniques that work for your project and to the successful adoption of the AWE Process.

6.7.1 Adopting, Creating, Tailoring and Evaluating Techniques

AWE supports iterative and incremental development life-cycles. Initially each project should develop a set of criteria that identifies all the project risks that can be reduced by techniques. Focusing initially during the first iterations on the highest risk objectives of the project, each AWE team and respective developer is responsible for contributing and collaborating on the task of successfully integrating techniques that will work with the AWE Process. The rapid change within the business, domain and software models within Web engineering, and the major impact change has between all models, requires constant review and evaluation of the techniques being used. It is essential that each team constantly review the contributions that each technique is making to the project, ideally this should happen upon completion of each AWE life-cycle iteration.

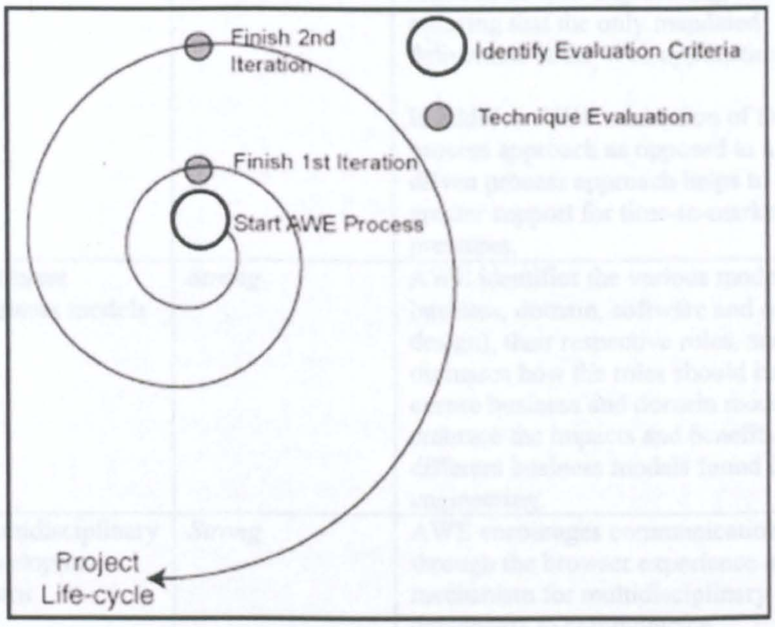


Figure 22. Evaluation Criteria and Technique Evaluation Stages on an Iterative and Incremental AWE Project Life-cycle

Figure 22 shows a typical iterative and incremental development AWE project life-cycle indicating where technique evaluation should be carried out. The iterative and incremental style of the AWE Process life-cycle is based upon Boehm's (1998) Spiral Model. See Section 6.3 of this dissertation for a more detailed description of the AWE Process life-cycle. Technique Evaluation is used to identify what techniques are to remain or be dropped depending on whether they are benefiting or hindering the project. This should be carried out according to the criteria previously identified in Identify Evaluation Criteria, and also to identify where new project techniques are required. For example, where a technique is observed to be working in intra team communication (amongst a single team), the team should continue to use it, ensuring that other teams are informed through inter team communication (between many different teams) to raise awareness of the benefits of the technique in question. If techniques appear to be problematic, review what works and what does not. If the technique can be altered and the team believes that the alterations indicate successful adoption within the project, continue to use it, else get rid of the technique and look for another to take its place if required. Again, remembering to inform other sub-teams through inter team communication channels of the teams decision to drop a technique and why.

6.8 AWE: Support for the Criteria for a Web Engineering Process

Table 8 below summarises AWE support for the criteria for a Web engineering process, using the criteria identified in chapter 3.

No.	Support	Comments
1. Short development life-cycle times	*Strong*	AWE shows strong support for criterion 1 'Short development life-cycle times'. This is primarily achieved through its iterative and incremental process life-cycle and explicit focus on the phases required in Web engineering, but ensuring that the only mandated deliverable is the Web application itself. In addition AWE's adoption of the agile process approach as opposed to a plan-driven process approach helps to ensure greater support for time-to-market pressures.
2. Different business models	*Strong*	AWE identifies the various models (e.g. business, domain, software and creative design), their respective roles, and discusses how the roles should interact to ensure business and domain models embrace the impacts and benefits of the different business models found in Web engineering.
3. Multidisciplinary development teams	*Strong*	AWE encourages communication through the browser experience as a mechanism for multidisciplinary developers to communicate. AWE identifies and discussed the roles required to reflect the multidisciplinary nature of Web engineering. These roles include end-users, clients, business experts, domain experts, software engineers, creative designers and team leaders.
4. Small development teams working in parallel on similar tasks	*Strong*	AWE's orthogonal uni-discipline developer communication provides explicit support for scaling AWE to large Web engineering projects.
5. Analysis and Evaluation	*Strong*	AWE explicitly focuses on the importance of the business analysis phase within the AWE life-cycle with a discussion of the objectives and potential activities carried out within this phase. AWE focuses on an evaluation phase with a representative sample of end-users and encourages explicit focus on the importance of end-user involvement and the usability during architectural considerations.
6. Requirements	*Strong*	AWE places explicit focus on the

and Testing		objectives that need to be addressed in both the requirements and testing phase.
7. Maintenance	*Strong*	AWE explicitly discusses the architectural issues associated with Web engineering that are important when evolving and maintaining Web applications.

Table 8. AWE Support for the Criteria for a Web Engineering Process

6.9 How Agile is AWE?

Visconti and Cook (2004) propose an ideal agile process model based on the principles established in the Agile Manifesto (Beck et al. 2001). See appendix 5 for the principles behind the Agile Manifesto. Using the agile practices defined by their ideal agile process model, Visconti and Cook provide a mechanism to assess an agile process to see what agile practices are missing or need improvement. The ideal process model is proposed as "the standard in assessing the *agility* of agile methods" (Visconti & Cook, 2004 p. 432). In this section the AWE process is assessed against the ideal process model proposed by Visconti and Cook. Table 9 shows this Ideal Agile Process Model.

Phase	Dimension	
	Core Process	Customer Satisfaction
Identify	• Deliver working software frequently • Constant interaction between customers and developers • Maximize the amount of work not done • Face-to-face communication in development team • Constant rate of development • Self-organized teams	• Satisfy customer • Welcome changing requirements • Continuous attention to technical excellence and good design • Provide environment and support needed
Monitor	• Early and continuous delivery of valuable software • Business people and developers work together daily • Trust developers to get the job done • Proper environment and support provided	• Early and continuous delivery of valuable software • Business people and developers work together daily
Measure	• Working software is primary measure of progress	• Working software is primary measure of progress
Feedback	• At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour	• Satisfy customer • At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour

Table 9. Ideal Agile Process Model. Reproduced from Visconti and Cook (2004, p. 435)

The Ideal Agile Process Model has two dimensions Core Process and Customer Satisfaction and four phases: Identify, Monitor, Measure and Feedback. The model shows the mapping of the agile principles to the phases and dimensions of the model. Using this model Visconti and Cook (2004) compare two popular agile processes, XP (Beck 2000) and Scrum (Schwaber & Beedle 2002) identifying areas where these processes do not explicitly address some of the principles of the Agile Manifesto, reproduced from (Visconti & Cook 2004, p. 438) in table 10.

Phase/Dimensions	Ideal Agile Practices	XP Practices	Scrum Practices
Identify/Core Process	Deliver working software frequently	Small releases and continuous integration	Sprints
	Constant interaction between customers and developers	Whole team and onsite customer	Scrum teams, product owner
	Maximize the amount of work not done	Planning game	Sprint planning meeting
	Face-to-face communication in development team	Whole team and pair programming	Daily scrum
	Constant rate of development	Sustainable pace/40-hour week and small releases	Sprint
	Self-organized teams	Whole team	Scrum team
Monitor/Core Process	Early and continuous delivery of valuable software	Small releases and continuous integration	Sprint review meetings, empirical management
	Business people and developers work together daily	Whole team and onsite customer	(4)
	Trust developers to get the job done	Pair programming, collective code ownership, metaphor	Scrum teams and sprint review meetings
	Proper environment and support provided	(2)	Scrum master
Measure/Core Process	Working software is primary measure of progress	Small releases, continuous integration and big visible chart	Sprint review meetings, empirical management and sprint backlog graph
Feedback/Core Process	At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour		
Identify/Customer Satisfaction	Satisfy customer	Testing	Product backlog, sprint review
	Welcome changing requirements	Small releases and continuous integration	Product backlog
	Continuous attention to technical excellence and good design	Testing, simple design, pair programming, refactoring and coding standards	(3)
	Provide environment and support needed	(1)	Scrum master
Monitor/Customer Satisfaction	Early and continuous delivery of valuable software	Small releases and continuous integration	Sprint review meetings, empirical management
	Business people and developers work together daily	Whole team and onsite customer	(4)

Measure/Customer Satisfaction	Working software is primary measure of progress	Small releases, continuous integration and big visible chart	Sprint review meetings, empirical management and sprint backlog graph
Feedback/Customer Satisfaction	Satisfy customer	Testing, small releases, continuous integration	Sprint review meetings
	At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour	Small releases, continuous integration and big visible chart	Sprint review meetings, empirical management and sprint backlog graph

Table 10. Ideal Agile Process Model – Mapping of practices for XP and Scrum. Reproduced from Visconti and Cook (2004, p. 438)

Table 10 highlights at points (1) and (2) that XP does not explicitly acknowledge the need to provide and monitor proper environment and support. Point (3) illuminates where XP does support continuous attention to technical excellence. Point (4) illustrates Scrum's poor support for customers who are not part of the Scrum development team. The interaction with customers is conducted through the 'product owner' in Scrum.

Table 11 shows the mapping of AWE's practices to the Ideal Agile Process Model proposed by Visconti and Cook (2004). The customer community discussed by Visconti and Cook does not distinguish between the client and the end-users, instead they just refer to customers. AWE provides explicit focus on the different stakeholder roles of the client and end-user in Web engineering providing a more precise customer community view for Web engineering. In addition, AWE also explicitly focuses on stakeholder roles e.g. business and domain experts that are often referred to as customers in other processes such as XP where the customer community view is coarsely defined. Point (5) in table 11 shows that AWE's practices do not explicitly support a constant rate of development. It is not that a constant rate of development is not achievable with AWE, rather that the AWE development team is responsible for the rate of development.

Phase/Dimensions	Ideal Agile Practices	AWE Practices
Identify/Core Process	Deliver working software frequently	Iterative and incremental lifecycle. Web-based deliverables being delivered after every iteration.
	Constant interaction between customers and developers	Collocated multidisciplinary team including business experts, domain experts, team leader, software engineers and creative designers. Explicit focus on end-user through the evaluation phase.
	Maximize the amount of work not done	Only mandated deliverable is the Web application itself
	Face-to-face communication in development team	Collocated multidisciplinary team including business experts, domain experts, team leader, software engineers and creative designers.
	Constant rate of development	(5)
	Self-organized teams	Team responsible for deciding deliverables beyond Web application. Team responsible for selecting techniques to support AWE.
Monitor/Core	Early and continuous	Each iteration focuses explicitly on addressing

Process	delivery of valuable software	the business objectives that present the highest risks to project success. Explicit focus on end-user through the evaluation phase. Developers communicate through the shared browser experience.
	Business people and developers work together daily	Collocated multidisciplinary team including business experts, domain experts, team leader, software engineers and creative designers
	Trust developers to get the job done	Team responsible for deciding deliverables beyond the Web application itself. The AWE development team monitors effectiveness and adjusts behaviour where appropriate to ensure success. In a large project using AWE with many sub-teams, each team communicates with others using co-ordination team and orthogonal uni-discipline developer communication.
	Proper environment and support provided	Explicit focus on the architectural and usability issues in Web engineering.
Measure/Core Process	Working software is primary measure of progress	The multidisciplinary AWE development team communicate through the shared browser experience. Explicit focus on the Testing phase in every lifecycle iteration.
Feedback/Core Process	At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour	Team responsible for deciding deliverables beyond the Web application itself. The AWE development team monitors effectiveness and adjusts behaviour where appropriate to ensure success. Team responsible for creating, evaluating and selecting techniques to support AWE.
Identify/Customer Satisfaction	Satisfy customer	Explicit focus on end-user validation of deliverables through the Evaluation phase to meet customer's project objectives.
	Welcome changing requirements	Requirements and Business Analysis are key phases in every lifecycle iteration
	Continuous attention to technical excellence and good design	Explicit focus on the Design and Testing phases in every lifecycle iteration.
	Provide environment and support needed	Explicit focus on the architectural issues in Web engineering.
Monitor/Customer Satisfaction	Early and continuous delivery of valuable software	The multidisciplinary AWE development team communicate through the shared browser experience. Explicit focus on Testing phase in every lifecycle iteration. Explicit focus on end-user validation of deliverables through the Evaluation phase to meet customer's project objectives.
	Business people and developers work together daily	Collocated multidisciplinary Team including business experts, domain experts, team leader, software engineers and creative designers.
Measure/Custome r Satisfaction	Working software is primary measure of progress	The multidisciplinary AWE development team communicate through the shared browser experience. Explicit focus on end-user validation of deliverables through the Evaluation phase to meet customer's project objectives. End-user performance in the Evaluation phase is the litmus test for project success.

Feedback/Customer Satisfaction	Satisfy customer	Collocated multidisciplinary Team including business experts, domain experts, team leader, software engineers and creative designers. Explicit focus on end-user through the evaluation phase.
	At regular intervals, team reflects on how to become more effective, tunes and adjusts its behaviour	The AWE development team monitors effectiveness and adjusts behaviour where appropriate to ensure success. Team responsible for creating, evaluating and selecting techniques to support AWE.

Table 11. Ideal Agile Process Model – Mapping of Practices for AWE

This lightweight approach indicates that AWE addresses all the principles of the Agile Manifesto (Beck et al. 2001). While quantifiable measurements would provide better feedback through empirical studies of each of the processes discussed in this section, the Ideal Agile Process Model uses standard process texts to assess the support an agile process has for the Agile Manifesto principles. It is evident from table 11 that AWE explicitly supports the vast majority of the agile manifesto principles. As to how well AWE addresses each of the principles is something that requires further research based on empirical analysis of AWE in practice. However table 11 strongly indicates AWE's support for the principles behind the Agile Manifesto and presents justification for its classification as an Agile Process.

6.10 Summary

This chapter describes the Agile Web Engineering (AWE) Process and summarises its support for the criteria for a Web engineering process. Unlike other agile processes AWE is designed for a particular classification of software activity, i.e. Web-based development. This is not to say that the agile route does not extend to other types of software activity. Just that the AWE process has only been developed to tackle the particular characteristics associated with Web engineering. The AWE process lends itself well to the development of Web-based systems, however, AWE is only one step towards tackling the problems in Web engineering. The individuals responsible for the development of Web-based systems are the most important factor in project success. The AWE Process can only hope to have a second order effect on project success. This is illustrated by the responsibility placed on the individuals, teams and organisations that must decide what techniques, tools and technologies are required to support them and AWE and help achieve success on each of their projects.

Everyone involved in contributing to the deliverables on a Web application, whether it be copy or text, graphics, business objectives or database designs are responsible for the success of their Web-based endeavours. It is essential that all involved realise the importance of their contribution and that of the other members of the team. This requires collaboration, compromise, education, understanding and a high degree of professionalism. Should these developer criteria be seriously lacking amongst those responsible for the development of Web-based systems, then one must question the effectiveness of this group applying the agile approach and AWE.

It is essential that those involved in selecting a process for the development of Web-based systems understand the differences between plan-driven and agile process approaches. Agile approaches are measurable, using project metrics, in completely different ways to predictive approaches. Essentially agile processes rely heavily on continuous delivery of working software as a measurement of project success. However, it is too early yet to discuss the effectiveness of measuring all types of software activity in such an adaptive manner. The environmental and cultural barriers that the adaptive approach must overcome clearly present one of the biggest challenges to the adoption of AWE and other agile approaches.

The development of Web applications requires multidisciplinary development teams. In order to scale AWE to large Web engineering projects it is essential that orthogonal uni-discipline communication be led by a co-ordination team. This should help to ensure a consistent end-user experience and the avoidance of effort duplication. Again this is one of the biggest challenges to the adoption of AWE on many Web-based projects, due to the geographical spread of developers and corporate inertia within many organisations.

End-users are the litmus test for success. Developing Web-based systems without close involvement with end-users is flawed. It is essential that all developers are focused on the end-users of the proposed system and try to provide the best usage experience given the project resources available. Indeed, the evolution of Web-based systems requires a great degree of architectural focus. Web applications require architectural focus not only on usability but also on the application design, to try to ensure a scalable, robust, reliable and evolvable deliverable.

This chapter provides a rationale for the creation of the Agile Web Engineering (AWE) process, describes AWE in detail, and compares it against the criteria for a Web engineering process. The following chapter compares four other commercial Web engineering processes against those criteria. Chapters 8, 9 and 10 then describe field trials of AWE in a commercial environment.

7 Other Web Engineering Processes Support for the Criteria

Over the past five years a small number of processes specifically for Web engineering, and Web engineering focused evolutions to traditional software engineering processes, have been proposed. Processes created specifically for Web engineering include Collaborative Web Development (CWD) (Burdman 1999) and Crystal Orange Web (COW) (Cockburn 2002). Web engineering focused evolutions to traditional software engineering process include Web OPEN (Haire, Henderson-Sellers & Lowe 2001) and the Rational Unified Process (Ward & Kroll 1999). This chapter describes each of these Web engineering processes and discusses the support shown by each approach to the criteria for a Web engineering process, listed in Table 12. Sections 2.1 and 2.2 discuss in detail the definition of process, the elements used to describe processes throughout this dissertation, and the ranking scheme used to illustrate a particular process's support for each of the criterion for a Web engineering process.

No.	Criteria for a Web Engineering Process
1.	Short development life-cycle times
2.	Different business models
3.	Multidisciplinary development teams
4.	Small development teams working in parallel on similar tasks
5.	Analysis and Evaluation
6.	Requirements and Testing
7.	Maintenance

Table 12. The Criteria for a Web Engineering Process

The important feature of the definitions presented in section 2.1 and 2.2 is that a process is much more than a set of techniques for developing Web-based applications. The existing literature on Web Engineering tends to focus on techniques for developing Web-based applications rather than covering the wider issues of the process life-cycle. Koch (1999) presents a good overview of methods for developing Web-based applications but states "Most of these methods focus on the design of hypermedia applications; only a few cover more aspects of the life-cycle, …". Gaedke and Gräf (2000) include a "Comparison of Process Models", however, the criteria they use does not cover all desirable aspects of a software development process.

The existing literature (ICWE 2002, ICWE 2003, ICWE 2004, JWE 2004) does not present any papers that clearly identify a set of criteria for evaluating Web engineering processes that cover all the relevant aspects of a software development process. Nor is there any attempt to systematically compare the emerging commercial processes and extensions to existing processes that claim to address the problem of developing Web based applications. Although all of the processes described in this chapter are based on different process engineers' experience and observations of Web engineering, there is no literature giving empirical evidence concerning the use of these processes. Therefore, as with chapter 5, the evaluations are based on a critical review of the literature describing the processes.

The rest of this chapter describes the four previously mentioned commercial Web engineering processes using the process elements discussed in section 2.2. As references to each criterion surface, through the description of process elements, they are given a ranking to show how strongly each process supports each of the criterion for a Web engineering process. The evidence and a rationale behind this ranking are also presented.

7.1 Collaborative Web Development (CWD)

Burdman (1999) wrote Collaborative Web Development (CWD) after gaining experience in over 20 Web engineering projects. Burdman comes from a technical and creative writing background, and her Web site development experience was primarily based on the development of informational Web applications. Thus, CWD represents an early attempt at defining a process for Web engineering that does not address some of the critical process issues concerning interactive Web application development. The rest of section 7.1 discusses the elements of CWD that are being used to describe the CWD process.

7.1.1 CWD: Process Life-cycle and Phases

CWD recommends a four phase Web development process life-cycle, see figure 23. The following quote from Burdman (1999, pp. 50-51) describes the four phases of the CWD process life-cycle:

> Phase I: *Strategy*. During this phase, either a strategic planner, account executive, or project manager and/or the client is determining the objective for the site based on a dedicated research effort. The culmination or deliverable of this phase is the creative brief, which clearly outlines the objectives, requirements, and key insights of the target audience. The creative brief provides a foundation for every team member's work;
>
> Phase II: *Design*. During this phase, the creative and technical teams are doing a parallel design of the site. The creative team is designing the user interface and interactions, and the technical team is designing the back-end applications that will be used on the site. The culmination of this phase is the functional and/or technical specification, site architecture, schematics, and designs of the site. Sign-off by the client is critical to proceed;
>
> Phase III: *Production*. During this phase, we are building the site. Any major changes in functionality and features have to be monitored closely. If the client requests a change at this point, a change order is issued. The culmination of this phase is, of course, the site which will go into Phase IV. A production guide is also created during this phase;
>
> Phase IV: *Testing*. During this phase we are testing the site and getting ready to publish it on the live, or production, Web server. Our QA manager develops a test plan based on the scope document and functional specification and tests against this plan. Bugs are reported and fixed. The site is ready to go live at the end of this phase.

The CWD process life-cycle is influenced heavily by early plan-driven software process developments. The life-cycle closely resembles the Stagewise (Bennington 1956) and Waterfall (Royce 1970) process life-cycles with at least one documented deliverable as an outcome of each CWD life-cycle phase. The life-cycle is neither iterative nor incremental does not adequately reflect the continuous and evolutionary nature of developing many Web-based applications.

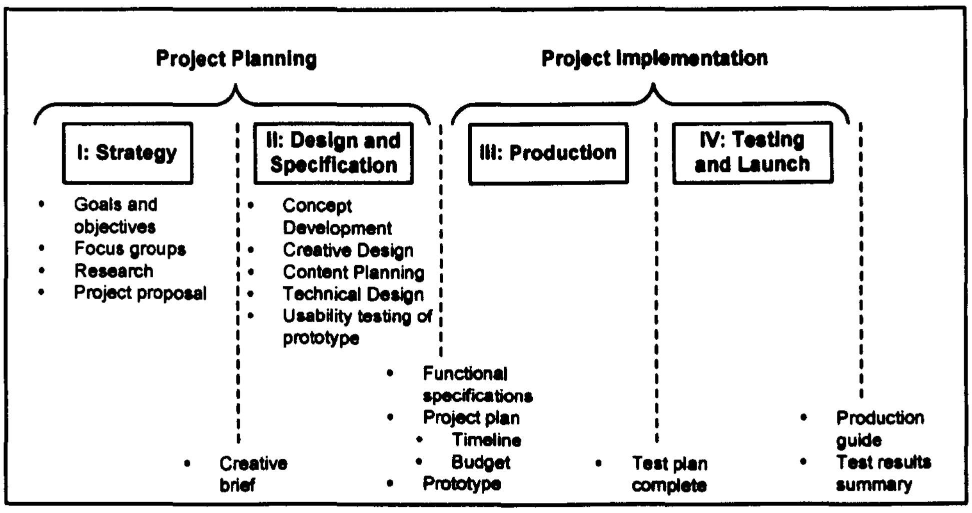


Figure 23. The CWD Web Project Development Life-cycle and Phases. Reproduced from Burdman (1999, p. 51)

CWD advocates focus on requirements and design collectively during Phase II. The criteria for a Web engineering process calls for explicit focus on requirements. The majority of the commercial Web engineering process approaches described in section 3.3 in the first survey of Web engineering in practice combined requirements capture and design within one phase. Criterion 6 which requires explicit focus on requirements within a Web engineering process was primarily derived from the problems observed with combining requirements with design phases. CWD's combination of requirements and design activities shows weak support for requirements.

The last phase of the CWD life-cycle focuses on testing the 'Web site' being delivered. CWD focuses particularly on the source executing in the browser tier of the Web application with little mention of testing within other tiers as the following quote (Burdman 1999, p.139) illustrates:

> Another form of testing, white-box or glass-box testing, is generally conducted by the programmer or coder. For Web sites, programmers use some type of HTML validator and also spend time executing and viewing JavaScript code in a browser.

The lack of focus on testing issues within the server tiers, and failure to discuss testing of business and domain processes that surround the Web application, suggest weak support for testing Web applications within CWD. The amalgamation of the requirements and design phase, and the reduced scope of testing within CWD, indicates weak support for criterion 6 'Requirements and Testing'.

CWD mentions usability testing in a small number of paragraphs dispersed throughout Burdman's (1999) book. However, there is little discussion of how, when and where usability testing should be carried out and integrated in CWD. The lack of explicit focus on evaluation and end-user involvement shows weak support for Evaluation in the CWD life-cycle. CWD lends itself poorly to the customer community view of Web application development and is symptomatic of the influence of the Waterfall and Stagewise processes that are suited more to the customer community view of traditional software development, (see section 4.5).

The following quote (Burdman 1999, p. 47) discusses the CWD approach to business analysis in the strategy phase:

> If you are going to invest in a Web site, the money you spend in the planning stage is the best money spent. It's the only way to ensure that your Web project will accomplish what it must in order to achieve your business goal. However, clients don't always want to pay for planning. Sometimes the client will have certain aspects of the Web site planned long before you are hired to build the project. It's up to you to decide at that point if that particular project is right for you and your team.

There is limited support for business analysis in CWD. This may be due to Burdman's experience as a contractor and the creative design background of companies she has worked for. The limited focus on business analysis and evaluation in CWD suggest a weak support for criterion 5 'Analysis and Evaluation'.

CWD recommends a fixed timeline and little changes to requirements during Project Implementation (Phases III and IV) without additional cost implications for the client. This is illustrated by the following quote from Burdman (1999, p. 51):

> In my contract, I note that if we exceed the fixed time line due to a client's change in initial requirements, there'll be an additional fee.

CWD shows partial support for criterion 1 'Short development life-cycle times'. The approach advocated by CWD to addressing time-to-market pressures in Web engineering is too predictive in nature. It is unrealistic to be able to predict everything required to make a Web engineering project successful upfront. CWD may prevent many client change requests that cannot be resourced, or are unrealistic given the time constraints of the project. However, the extreme strategy employed by CWD for addressing time-to-market pressures may prevent many simple and beneficial changes that occur as the client and team evolve their understanding of the project's problem space and proposed solution. The CWD strategy for addressing the short development times encountered in Web engineering projects conflicts with the very name given to this process, i.e. 'Collaborative Web Development'. Clearly collaboration between the client and the development team is not of primary focus, particularly during Project Implementation phases.

7.1.2 CWD: Deliverables

The deliverables recommended by CWD are primarily document centric and resemble the approach more regularly associated with the Waterfall model (Royce 1970). The CWD deliverables are listed as bullet points in Figure 23 and are described in detail in 'Collaborative Web Development' (Burdman 1999). The Production Guide deliverable shows focus on the traditional software engineering maintenance issues reflected from the software and creative design models, but is non existent with respect to maintenance for the business and domain models required in Web engineering. This is illustrated by the following quote (Burdman 1999, pp. 63-64):

> I use a production guide to keep track of how the site was built, and then I hand this guide to my client to serve as documentation of the site, from how the

graphics were created to what the files were named and why. Here are some important facts to include in your production guide:

- Directory structure ...
- File names ...
- Coding and scripting notes ...

Production art notes ...

There is no mention of the business processes that may be introduced, re-engineered or evolved by a Web engineering project or how the domain or business models will be impacted by the Web-based deliverables. The CWD Production Guide may be suitable for a static Web site serving to aid a marketing campaign but it is not sufficient to address the maintenance in the business and domain models within many Web engineering projects. For example, CWD would be poorly suited to many of the business and domain model maintenance issues associated with a business-to-business (b2b) e-commerce Web application. With respect to criterion 7 'Maintenance', CWD is ranked as providing partial support.

7.1.3 CWD: Activities

Burdman (1999) devotes Chapter 4 of her book to 'Communication Issues' in CWD. Figure 24 describes the 'chain of communication' as proposed for CWD project teams. The 'chain of communication' between the client and the development team primarily reflects one way impacts from the business and domain models to the software model, as is found in traditional software processes. CWD's 'chain of communication' shows weak support for impact back into the business and domain models as described in criterion 2. The following two quotes from 'Collaborative Web Development' (Burdman 1999, pp. 80-81) describe examples of communication within the chain:

> During a review with the account manager, the creative director, the producer, and the client requests a change. The producer communicates this change to the project team either by sending an e-mail to the entire team or by sending an e-mail to each of the respective team leaders, depending on the size of the team.
>
> The MIS manager tells a programmer that the server the team is using is flaky and must be shut down. The programmer tells his team leader, who tells the producer. If the client has to know, the producer tells the account manager, who talks to the client. The producer communicates the news to the rest of the project team through either the team leaders or global e-mail.

As illustrated in the second quote above, the majority of the CWD team members are up to three stakeholders removed from the client with respect to communication. The CWD development team seem to be deliberately removed from the client and any of the client's employees, with only the producer and account manager having regular direct face-face communication with the client. Figure 24 implies that only the core team members are involved in communication between sub-teams.

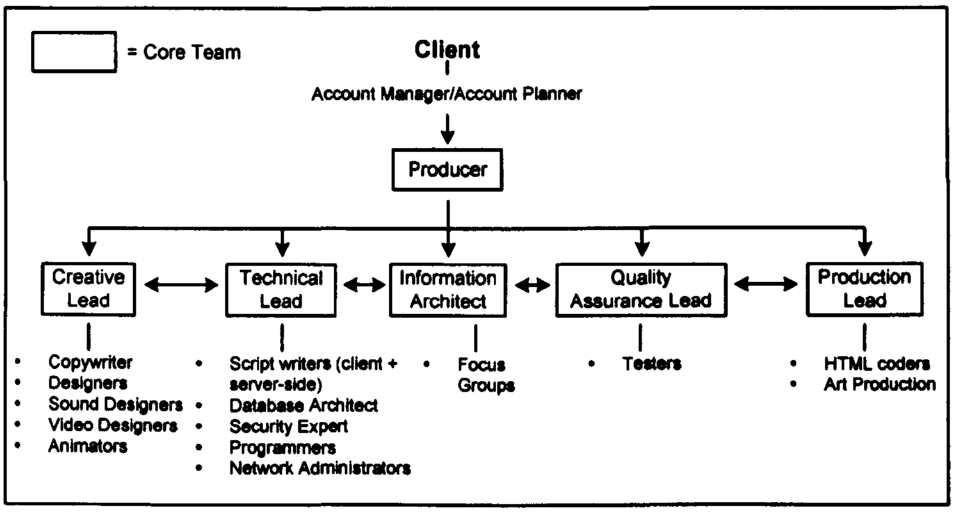


Figure 24. The Chain of Communication on a CWD Project Team. Reproduced from Burdman (1999, p. 80)

Burdman addresses large Web engineering endeavours in Chapter 6 'Multi-departmental and Large-Scale Sites'. With respect to assisting with scalability, the chapter devotes much attention to the issues surrounding scalability on large Web-based projects but has few recommendations on how to address these issues within a Web development team. The following quote form Burdman (1999, p. 118) describes CWD teams on large Web engineering projects:

> The roles for the Web team in large-scale Web site or intranet are the same, as you learned in chapter 2; however, there could be a lot more of them, and their roles could be more specialised.

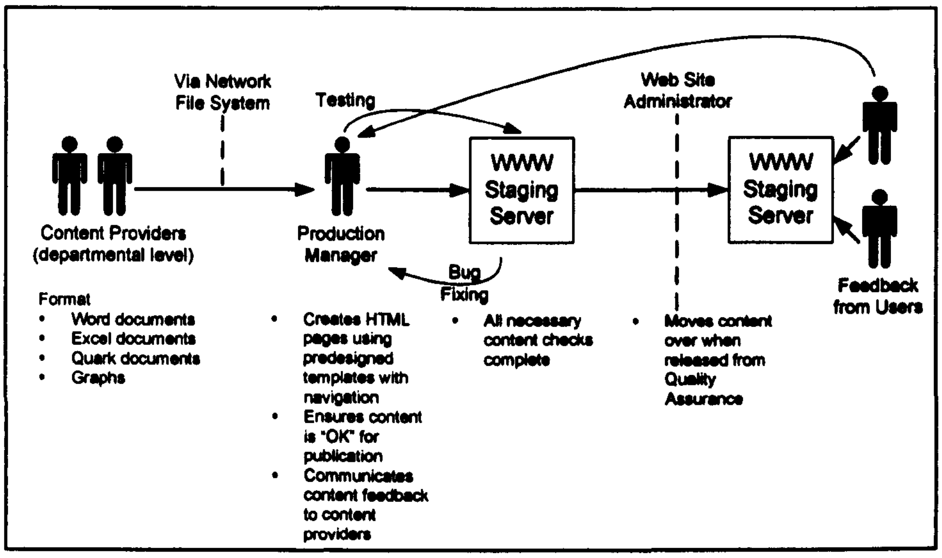

Via Network File System
Testing
Web Site Administrator
Content Providers (departmental level)
Production Manager
WWW Staging Server
WWW Staging Server
Bug Fixing
Feedback from Users
Format
Word documents
Excel documents
Quark documents
Graphs
Creates HTML pages using predesigned templates with navigation
Ensures content is "OK" for publication
Communicates content feedback to content providers
All necessary content checks complete
Moves content over when released from Quality Assurance

Figure 25. The CWD Publishing Process or 'Flow of Content' on Large-Scale Web Site or Intranet Project. Reproduced from Burdman (1999, p. 119)

CWD advocates the same team structure to small and large teams, in addition the roles more closely associated with business and domain are again removed from the CWD Web development team. CWD is ranked as showing weak support for criterion 4 'Small development teams working in parallel on similar tasks'.

Figure 25 describes the CWD publishing process on large Web engineering projects. Most of the focus of this process surrounds the creation, testing, bug-fixing and approval of static Web-based content (e.g. HTML pages) and is poorly suited to many of the dynamic technologies, such as J2EE, that are core to many large corporate Internet, Intranet and Extranet sites.

7.1.4 CWD: Roles

CWD describes a number of roles comprising a CWD development team. These are listed in Figure 24. CWD roles show a detailed understanding of the types of skills in the software and creative design models required to build many Web applications, although clearly the creative design roles are given prominence. For example, Tester and Programmer are reflected in CWD as non core team roles.

CWD classifies developers into a number of role types: core team members (C), extended team members (E) and specialist team members (S). The core roles are those found on all CWD projects. Extended roles are described as "people whose skills might not always be necessary or who might have cross functional roles" (Burdman 1999). Specialist roles are described as "people who are brought in to do work that is not part of your core or extended team but may become so" (Burdman 1999). The roles described in the boxes in Figure 24 represent the core CWD team members and those listed in bullet points are extended and specialist CWD team members. The roles described by CWD give a strong representation of those reflecting the creative design and software models, although the majority of roles representing the software model are classified as extended or specialist roles. Despite listing and describing over 20 roles in a CWD team, there is no mention of business and domain roles as part of the CWD development team. With respect to criterion 3 'Multidisciplinary development teams' CWD shows partial support with roles only reflecting the creative design and software models.

7.1.5 CWD: Tools and Techniques

CWD is tool and technique independent. Burdman makes many recommendations regarding tools and techniques that can assist with the development of Web applications but refrains from mandating any of them. Burdman's background in creative and technical writing exposes a number of limitations in CWD with respect to its view of the roles and activities within the software model. The following quote (Burdman 1999, p. 166) illustrates such an example with respect to Lightweight Directory Access Protocol (LDAP) (Wahl, Howes & Kille 1997):

> ... LDAP is the latest recommended user-access control protocol. LDAP allows you to keep a central directory of users and their access rights. This is critical because you want people to be able to see your site, but this openness is exactly what makes it possible for people to break in and cause trouble.

The above quote clearly shows limited understanding of a key technology related to securing access and control within many Web-based and traditional software systems. LDAP is one of the most widely used directory implementation standards, if not the de-facto standard, for retrieving access and control information in directories used in security scenarios. Like any technology, an LDAP implementation can result in an insecure solution, if implemented incorrectly, or with poor a understanding of the implications and constraints of the technology. However, it is the simple and open nature of LDAP, along with the security features provided in version 3, that have made it so popular and led to its wide use throughout the industry.

7.1.6 CWD: Support for the Criteria for a Web Engineering Process

Table 13, below, describes CWD's support for the criteria for a Web engineering process.

No.	Support	Comments
1. Short development life-cycle times	*Partial*	CWD prevents any changes being incorporated during the production phase without cost implications. This approach is only suitable within contracting projects with agreed fixed requirements.
2. Different business models	*None*	There is no support for new business models, i.e. this is reflected in the lack of support for impact into business and domain models from the software model.
3. Multidisciplinary development teams	*Partial*	CWD only supports the creative design and development roles, ignoring the business and domain experts.
4. Small development teams working in parallel on similar tasks	*Weak*	Team structure is the same regardless of the size of the project and there is no inter-team communication model for parallel development.
5. Analysis and Evaluation	*Weak*	There is some support for business analysis in the Strategy phase but a lack of explicit focus on Evaluation within CWD.
6. Requirements and Testing	*Weak*	The Design and Specification phase, combining requirements and design to produce a technical specification in CWD is known to be problematic [4, 5]. Conventional testing by a development team only and focused primarily on the browser tier, i.e. the source executing in the browser. Poor focus on testing the other tiers of the Web application.
7. Maintenance	*Partial*	CWD maintenance focus is on software and creative design models, and ignores business and domain models.

Table 13. CWD's Support for the Criteria for a Web Engineering Process

7.2 Crystal Orange Web (COW)

Crystal Orange Web (COW) (Cockburn 2002) is part of a family of agile processes developed by Alistair Cockburn, known as the Crystal family. Crystal processes are: "people- and communication-centric"; intended for development teams that are collocated within the one building and are not designed for safety critical systems. The following quote from Cockburn (2002, p. 204) 'Agile Software Development' describes the characteristics of projects that Crystal Orange (CO) was created to address:

> Crystal Orange is a methodology for D40 category projects: up to 40 people, sitting in the same building, working on a system that might cause loss of discretionary monies... The characteristics of such a project are:
>
> - It is staffed by 10 to 40 people total;
> - It is of 1 to 2 years in duration;
> - Time-to-Market is important;
> - There is a need to communicate with present and future staff, and a need to keep time and costs down;
> - It is not a life-critical system.

Crystal methodologies should be adjusted to fit a particular setting. COW is a specific adjustment of the Crystal Orange Process for Web engineering. COW was the Crystal methodology used to deliver a Web-based application for eBucks.com (2004). COW is described by Cockburn (2002, p. 206) in the following quote:

> Crystal Orange Web is a methodology we created for eBucks.com, a company delivering code to the Web in a continual stream. It differs from Crystal Orange in that this methodology does not deal with a single project but with a continuous stream of initiatives that require programming and with each initiative's results being merged with the growing code base being used by the public. This methodology is still in its trial run.

The following quote (Cockburn 2002, p. 207) describes the primary objective of COW, which is to ensure that the code that eBucks.com were putting live was as defect free as possible.

> The company [eBucks.com] had already established a Web presence. It was no longer driven by time-to-market pressures but was beginning to feel pressures imposed by the cost of defects. Customer calls, arriving in exponentially increasing volumes, could easily negate profit margins. Thus, the company was shifting from productivity to defect freedom as its top priority.

7.2.1 COW: Process Life-cycle and Phases

The following quote from Chapter 6 of 'Agile Software Development' (Cockburn 2002, pp. 207-208) illustrates the strong support of COW for addressing criterion 1 'Short development life-cycle times':

> ... Overall production runs in fixed length development cycles of two weeks. After each delivery, each team member may opt for the next delivery to be either two or four weeks, depending on what the team can deliver of use to the public.

While Cockburn encourages business owners to consider the business processes required for software failure or errors, there is no encouragement to discuss the potential benefits that may be derived by re-engineering business processes. The following quote from (Cockburn 2002, p. 208) illustrates the weak support of COW for criterion 2 'Different business models':

> A business owner writes a business use case and a system use case brief. The business use case illustrates the proposed new system features in operation, paying particular close attention to the manual business processes that are invoked when things go wrong. The brief is used by the technology group to estimate the work involved in creating the features.

The last sentence in the above quote illustrates that consideration of the opportunities presented by introducing new Web-based functionality is carried out solely by the business owner. COW does not support the technologist's potential to impact and educate on the potential benefits of the proposed solution, such as re-engineering or creating new business processes. The above quote also illustrates the lack of explicit encouragement of business process re-engineering activities in COW with little provision for impact back into the business and domain models from the software model.

Focus on requirements within COW is supported by the *detail business analysts* who "produce use cases and data descriptions, which go to the user interface designers, the server programmers, and the servlet programmers". COW emphasises support for testing by the developers and a separate testing team as illustrated by the following quote (Cockburn 2002, p. 208) describing the latter stages of the COW development process:

> The server and servlet programmers produce regression tests for their code and peer review the test cases. When the test cases are deemed good and the code passes the tests, the code is passed to the integration testers, who pester the developers to fix whatever remaining errors they find before the major deployment.
>
> The integration testers post the changes going out in the new release to the internal group and also to the call centre.
>
> For live code, the call centre returns bug reports to a special SWAT team whose sole purpose is to fix problems in production. The SWAT team is selected from the development group on a rotating basis every two cycles.

Although there is a strong emphasis on testing the functional requirements of the Web application, there is little mention of how to test the non-functional requirements of the system. COW is classified as showing partial support for criterion 6 'Requirements and Testing'.

COW does not mention evaluation or end-user involvement during the development life-cycle. In addition once the software is live, user feedback is managed through the intermediate of the 'call centre'. The lack of direct contact between end-users and the COW team will ultimately hinder the primary objective of COW in trying to address non-functional defects with respect to usability and smooth integration with business processes and shows no support for Evaluation. COW is classified as showing weak support for criterion 5 'Analysis and Evaluation'.

There is no mention of long term maintenance and evolution issues within COW. The focus is on rapid short-term evolutionary steps that are primarily focused on reducing defects with respect to the functional requirements of the Web application. COW's focus on testing and documented requirements should help assist projects using COW with maintenance issues. COW is therefore classified as showing partial support for criterion 7 'Maintenance'.

7.2.2 COW: Deliverables

The only enforced deliverable on COW development teams is that they "must deliver something useful to the public every four weeks" (Cockburn 2002). COW does not attempt to define acceptance criteria for what is useful. Instead this is left to the discretion of those adopting COW.

7.2.3 COW: Activities

One of the two base techniques that underpin the use of the Crystal family is reflection workshops. The Crystal family has a rule that states that "the team hold pre- and post-increment reflection workshops (with a strong preference for holding mid increment reflection workshops as well)". The following quote from Cockburn (2002, p. 208) describes a COW post reflection workshop:

> Post reflection workshop, suggestions visibly posted. At the end of every life-cycle, the company meets to discuss what worked well, what didn't work so well, and which ideas to try out on the next cycle. The outcome of the meetings is a posted list of things to keep.

The Crystal family encourages concurrent development in non-bottleneck activities. However on the project eBucks.com the bottleneck was programming and therefore there is no discussion of how to achieve concurrent development with COW. The following quote from Cockburn (2002, p. 210) illustrates the weak support within COW for criterion 4 'Small development teams working in parallel on similar tasks':

> ... the general absence of concurrent development, which is one of my favourite development speed up techniques. Concurrent development is missing because of the bottlenecks in the system.

7.2.4 COW: Roles

There are a number of roles mentioned throughout the description of COW although no definitive list of roles for COW is presented. The following quote (Cockburn 2002, p. 210) describes the eventual goal with respect to team structure and team communication in COW:

> Eventually, the programmers, user interface designers, testers, business owners, marketers, and so on should sit in cross-functional teams to maximize the effect of conversation around delivering initiatives across speciality boundaries and to minimize the effect of rumouring about others' specialties. This will have to be balanced with staffing levels and growing space needs.

One of the objectives of COW, as illustrated from the above quote, is that ideally, the programmers, user interface designers, testers, business owners, marketers, et al. should sit in cross-functional teams. COW therefore shows strong support for criterion 3

'Multidisciplinary development teams', although this was not actually implemented by the reported system being developed for eBucks.com (2004).

7.2.5 COW: Tools and Techniques

COW is tool independent and minimal in its recommendation of techniques, for example, two of the techniques discussed by Cockburn when beginning the COW process include: business use cases, and system use case briefs. The following two quotes from Cockburn (2002, p. 202-206) with respect to the Crystal family illustrate Crystal's approach to techniques:

> The two base techniques in Crystal are:
>
> - The methodology-tuning technique: using project interviews and a team; workshop to convert a base methodology to a starter methodology for the project
> - The technique used to hold reflection workshops.
>
> You are welcome to replace those two techniques with others if you have another way of accomplishing these goals.
>
> The techniques used by the individual roles are left entirely to the discretion of the individuals.

7.2.6 COW: Support for the Criteria for a Web Engineering Process

Table 14 below summarises Crystal Orange Web's support for the criteria for a Web engineering process.

No.	Support	Comments
1. Short development life-cycle times	*Strong*	COW recommends two-week development cycles, with a further recommendation that "Each team must deliver something useful to the public every four weeks".
2. Different business models	*Weak*	COW encourages "business owners" to consider the business processes required for software failure or errors. However, there is no explicit encouragement to discuss the potential benefits that may be derived by re-engineering business processes.
3. Multidisciplinary development teams	*Strong*	One of the objectives of COW is that ideally, the programmers, user interface designers, testers, business owners, marketers et al. should sit in cross-functional teams.
4. Small development teams working in parallel on similar tasks	*Weak*	There is an absence of support for concurrent development in COW, although other Crystal processes show strong support for this criterion.
5. Analysis and Evaluation	*Weak*	Business analysis involves the business owner writing a business use case and a

		system use case brief. COW does not mention end-user involvement during the development life-cycle before live delivery of software.
6. Requirements and Testing	*Partial*	Requirements are supported by business analysts producing detailed use cases and data descriptions. COW emphasises support for testing by the developers.
7. Maintenance	*Partial*	There is no mention of long term maintenance and evolution issues within COW, the focus is on rapid short-term evolutionary steps.

Table 14. COW's Support for the Criteria for a Web Engineering Process

7.3 Extensions to the Rational Unified Process (Extended RUP)

There are extensions for Web application development with IBM's Rational Unified Process (RUP) (Kruchten 2003), a well known, plan-driven software process product which is widely used for the development of object-oriented systems. A Rational white paper by Ward and Kroll (1999), describes how RUP can be extended for Web-based application development (hereafter referred to as extended RUP). This paper "focuses particularly on the front-end of the life-cycle, and how to integrate the creative design process with the software engineering process of the Rational Unified Process". Since the publication by Ward and Kroll (1999) the RUP product (Rational 2003) has come to incorporate a Conceptual Road Map entitled 'Developing e-business Solutions' that is primarily a reflection of the extensions proposed by Ward and Kroll (1999).

In assessing the suitability of extended RUP, consideration has been given to the explicit extensions to the basic process (Rational 2003) and also to where the extended process depends upon the foundations of the RUP process. Where there is any ambiguity, the unextended well-known version of RUP is referred to as 'vanilla RUP'.

7.3.1 Extended RUP: Process Life-cycle and Phases

The RUP is an iterative and incremental process that is influenced by Boehm's (1988) Spiral model. The RUP life-cycle has four phases:

1. *Inception*. The overriding goal of the inception phase is to achieve concurrence among all stakeholders on the life-cycle objectives for the project;
2. *Elaboration*. The purpose of the elaboration phase is to analyse the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the project's highest-risk elements;
3. *Construction*. During the construction phase, all remaining components and application features are developed and integrated into the product, and all features are tested thoroughly;
4. *Transition*. The purpose of the transition phase is to move the software product to the user community.

RUP includes business modelling workflow in the Inception phase before deriving software requirements. RUP does not focus heavily on an Evaluation phase with end-users or how to integrate support for Evaluation into the rest of the process, although RUP does mention usability testing. Extended RUP is therefore classified as showing partial support for criterion 5 'Analysis and Evaluation'.

Extended RUP places explicit focus on the capture of all types of requirements including functional (use case model) and non-functional (supplementary specification and creative design brief). It is worth pointing out that the RUP Concept: 'Developing e-business Solutions' (Rational 2003) recommends "less emphasis" on a number of RUP Workflow Details involved during the Requirements phase:

- Workflow Detail: *Analyse Problem* - The purpose of this workflow detail is to gain agreement on the problem being solved;
- Workflow Detail: *Understand Stakeholders' Needs* - The purpose of this workflow detail is to understand the needs of the primary project stakeholders by gathering information about the desired or envisaged product;
- Workflow Detail: *Define the System* - The purpose of this workflow detail is to begin converging on the scope of the high-level requirements by outlining the breadth of the detailed requirements for the system.

The Creative Design Brief places explicit focus on those non-functional requirements that are specifically relevant to the creative design model. The testing elements are contained within vanilla RUP and although there is no discussion of Web site testing by Ward and Krolls (1999) the RUP Product (Rational 2003) mentions usability testing, performance testing and Web site structure testing. Extended RUP therefore shows strong support for criterion 6 'Requirements and Testing'.

Extended RUP makes no explicit mention of maintenance issues. However, the number of documented deliverables that are produced within the creative design and software models. In addition, the iterative and incremental nature of the vanilla RUP process suggests partial support for criterion 7 'Maintenance' by extended RUP.

The RUP Process is too predictive and heavy-weight for Web application development. The integration of the creative design process alone (Ward & Kroll 1999) requires the additional development of a 'Full Web User Interface prototype', preceded by six documented deliverables, to be produced covering all use-cases, before the construction phase. See figure 26 for a diagrammatic representation of the workflow involved in producing the 'Full Web User Interface prototype'. Extended RUP does little to help projects address time-to-market pressures with the exception of recommending re-use of use cases from previous Web projects to address time-to-market. Extended RUP is therefore classified as having weak support for criterion 1 'Short development life-cycle times'.

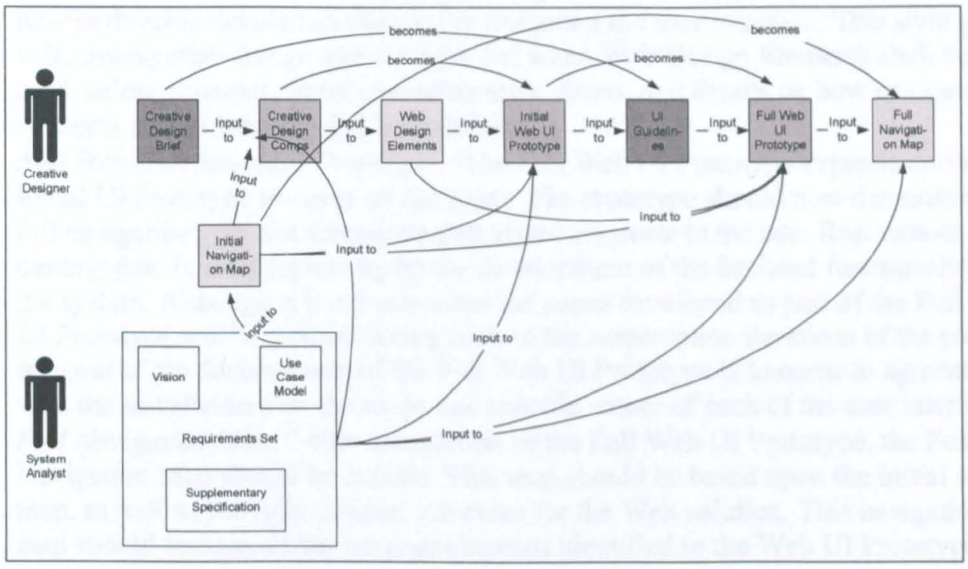


Figure 26. Integrating the Creative Design Process and the Rational Unified Process. Reproduced from Ward and Kroll (1999)

7.3.2 Extended RUP: Deliverables

Figure 26 illustrates the additional deliverables proposed to integrate the creative design model with extended RUP. These deliverables are described below with direct quotes from (Ward & Kroll 1999, pp. 3-7):

- *Creative Design Brief*. "In parallel to identifying actors and use cases, the initial user interface guidelines are developed; these high-level guidelines are often referred to as a Creative Design Brief in the Web community. The Creative Design Brief defines: the mood of the site (e.g. does the site convey authority, playfulness, or service? Is it conservative or provocative?); how users will be accessing the site (e.g. their connection speed); the browsers the users will be using; whether the site will use frames; any colour limitations the site will have; if applicable, a graphics standards guide (including standards on logos and all corporate colours); what sort of "bells and whistles" are wanted (e.g., mouse-overs, animation, news feeds, multimedia, etc.)";
- *Navigation Map*. "The navigation map is a view of the Web solution showing how users of the site will navigate it, represented in a hierarchical "tree" diagram. This sort of diagram is often referred to as a "site map", but we have chosen to call it a navigation map";
- *Creative Design Comps*. "The Creative Design Comps present to the stakeholders a number of visual options for their Web solution in order to "prime the pump" of the creative process of arriving at a truly compelling visual design for the Web solution";
- *Web Design Elements*. "Web Design Elements are the discrete graphical images that are assembled to build the Web pages for a site";
- *Initial Web User Interface Prototype*. "The Initial Web UI Prototype typically supports only portions of the system, based upon the most important and representative use cases. The development of the Initial Web UI Prototype allows the users and designers to better communicate on both the look-and-feel of the site, as well as the functionality to be delivered";

- *User Interface Guidelines.* "When the Initial Web UI Prototype is complete, it is time to develop detailed guidelines for designing the user interface. This style guide will, among other things, specify how and when Web Design Elements shall be used, colour schemes, fonts, cascading style sheets, and details on how navigational elements should function and be positioned";
- *Full Web User Interface Prototype.* "The Full Web UI Prototype expands upon the Initial UI Prototype to cover all use cases. The prototype should now demonstrate full navigation between screens and all visual elements to the site. Real data or dummy data is used depending on the development of the backend functionality of the system. Although it is expected that the pages developed as part of the Full Web UI Prototype will be refined during each of the construction iterations of the project, the goal of the development of the Full Web UI Prototype is to come to agreement with the stakeholders on the scope and specific nature of each of the user interfaces";
- *Full Navigation Map.* "After completion of the Full Web UI Prototype, the Full Navigation Map should be created. This map should be based upon the initial site map, as well as the fully defined use-cases for the Web solution. This navigation map should include all known pages/screens identified in the Web UI Prototype".

The deliverables listed above focus on integrating the creative design model with the software model and managing the impact between both models. There is no discussion of how RUP manages impact from the software model to the business and domain models. The following quote from Kruchten (2003, pp. 141-142) describes the purpose and rationale behind the use of software engineering techniques to model the business:

> ... It facilitates understanding of how something described in the business domain might relate to something belonging in the software domain...
>
> Historically, we have seen that modelling techniques developed and matured in the software domain inspire new ways of visualising an organisation. Because object-oriented visual modelling techniques are common for new software projects, using similar techniques in the business domain comes naturally.

The first sentence of the above quote shows that the primary purpose of business modelling is to understand how the business model impacts the software model. There is no discussion of how the software model should impact the business or domain models. The impact reflected within the RUP process is from the business and domain model to the software and creative design models, as opposed to just the software model as in vanilla RUP. There is no mention of impact back into the business and domain models from the software model. Extended RUP is classified as showing no support for criterion 2 'Delivery of bespoke solutions and different business models'.

7.3.3 Extended RUP: Activities

There is poor focus on parallel development activities or coordination mechanisms to support many small teams in vanilla RUP (Rational 2003) and (Kruchten 2003) or its extensions for Web application development (Ward & Kroll 1999). Although Ward and Kroll (1999, p. 2) explicitly mention its importance as illustrated by the following quote:

> Building Web solutions involves a variety of stakeholders more so than other software application development. These stakeholders will usually include business executives, marketing, creative design, customer support, and the technology development team, among others. Having a process that facilitates communication among these varied stakeholders is critical to success.

Ward and Kroll only recommend a *Glossary* and *Navigation Map* to facilitate communication between stakeholders. Extended RUP therefore shows weak support for criterion 4 'Small development teams working in parallel on similar tasks'.

7.3.4 Extended RUP: Roles

Extended RUP focuses on the increased visibility of the creative design role when building Web solutions. Ward and Kroll (1999) explicitly mention the wider diversity of stakeholders required to build Web solutions than in traditional software engineering. However, they present no definitive list of roles and there is little discussion of how these roles integrate with the development process outside of the creative design role. Extended RUP therefore show only partial support for criterion 3 'Multidisciplinary development teams'.

7.3.5 Extended RUP: Tools and Techniques

RUP is very tightly integrated into a number of Rational Computer Aided Software Engineering (CASE) tools. However, there is no discussion of integration with tools to support the creative design model. With respect to formal techniques, RUP is tightly integrated into the Unified Modelling Language (UML) (Rumbaugh, Jacobson & Booch 1998), and the Rational CASE tools that are integrated with RUP provide strong support for addressing issues within the software model and the impacts into the software model.

7.3.6 Extended RUP: Support for the Criteria for a Web Engineering Process

Table 15 describes the analysis of extended RUP support for the criteria for a Web Engineering Process.

No.	Support	Comments
1. Short development life-cycle times	*Weak*	Process is too predictive and heavy-weight, requiring the development of a 'Full Web User Interface prototype', (six documented deliverables), to be produced covering all use-cases, before the construction phase. Recommends re-use of use cases from previous Web projects to address time-to-market.
2. Different business models	*None*	The impact reflected within the process is from the business and domain model to the software and creative design models, as opposed to just the software model as in vanilla RUP. There is no mention of the impact back into the business and domain models from the software model.
3. Multidisciplinary development teams	*Partial*	Focus on the increased visibility of the creative design role when building Web solutions. There is an explicit mention of a wider diversity of stakeholders required to build Web solutions than in traditional software engineering, but no integration of these roles into the development process.
4. Small development teams working in parallel on	*Weak*	No mention of parallel activities or coordinating many small teams.

similar tasks		
5. Analysis and Evaluation	*Partial*	RUP includes business modelling workflow before deriving software requirements. No explicit mention of evaluation with end-users or how to support this activity within the process.
6. Requirements and Testing	*Strong*	Explicit focus on capture of all types of requirements including functional (use case model) and non-functional (supplementary specification and creative design brief). Creative design places explicit focus on those non-functional requirements that are specifically relevant to the creative design model. The testing element is strong within vanilla RUP but there is no explicit mention of Web site testing.
7. Maintenance	*Partial*	No explicit mention of maintenance issues. However, a number of documented deliverables are produced within the creative design and software models.

Table 15. Extended RUP Support for the Criteria for a Web Engineering Process.

7.4 Extensions to OPEN (Web OPEN)

Web OPEN is based on Object-oriented Process, Environment and Notation (OPEN) (Henderson-Sellers 2004). OPEN is an object-oriented process framework developed and maintained by over thirty five members of the OPEN Consortium. A recent paper (Haire, Henderson-Sellers & Lowe 2001) describes how OPEN can be extended to include activities and tasks for Web-based application development; it presents sixteen new Tasks and one new Activity to enable OPEN to "fully support the new demands of Website construction and the delivery of business value on the Web". In assessing the suitability of Web OPEN, consideration has been given to the explicit extensions to the basic process (Haire, Henderson-Sellers & Lowe 2001) and also where the extended process depends upon the foundations of the OPEN process framework.

7.4.1 Web OPEN: Process Life-cycle and Phases

Table 16 reproduces the levels of software method understanding that are presented by Boehm and Turner (2003a) which are based on work by Cockburn (2002). These are designed to help sort out what various levels of people can be expected to do within a given method framework.

OPEN is a meta process that requires process engineers to create a process instance particular to their project or organization. There is therefore a strong dependency on a skilled process engineer, Level 3 under the levels of software method understanding and use, to ensure that the process is sufficiently tailored to deal with the time-to-market pressures experienced in Web engineering without compromising the criteria for a Web engineering process. It is unlikely that most Web projects will have access to such a skilled Level 3 process engineer, therefore Web OPEN shows weak support for criterion 1 'Short development life-cycle times'.

The OPEN framework includes business modelling but it is not clear how this relates to business analysis. Web OPEN increases the focus on designing the user interface, with a number of sub tasks based on Usage Centred Design (UCD) (Constantine & Lockwood 1999). Web OPEN does not mention Evaluation or the end-users of the proposed deliverables during development. However, OPEN provides a number of work products for Usability Testing, although the author could find no discussion of where, how or when these should be applied during Web development either in the OPEN or Web OPEN literature. Web OPEN is therefore classified as showing partial support for criterion 5 'Analysis and Evaluation'.

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation.
2	Able to tailor a method to fit a precedented situation
1A	With training, able to perform discretionary method steps (e.g. sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With training, can become level 2
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Table 16. Levels of Software Method Understanding and Use (after Cockburn). Reproduced from Boehm and Turner (2003a)

OPEN provides a lot of focus on requirements engineering and Web OPEN provides specific focus on testing through the Web Site Testing Task. Web OPEN is therefore classified as showing strong support for criterion 6 'Requirements and Testing'.

7.4.2 Web OPEN: Deliverables

A Task in OPEN is defined as "as a functionally cohesive operation (reified as a work unit) that is performed by a direct producer (role or tool). A single responsibility of the producer will be fulfilled by the execution of one or more tasks. A task results in the creation, modification, or evaluation of a version of one or more work products." The following sixteen tasks and four sub-tasks have been proposed by Web OPEN as extensions to OPEN (Haire, Henderson-Sellers & Lowe 2001) to enable support for Web application development:

1. *Build White Site;*
2. *Create content (on Website);*
3. *Create navigation map for Website;*
4. *Define acceptance criteria for Website;*
5. *Define Website testing strategy;*
6. *Design and Implement content management strategy;*
7. *Design and Implement personalization strategy;*
8. *Design Website architecture;*
9. *Design Website standards;*
10. *Develop a brand identity;*
11. *Develop Data Standard;*
12. *Integrate Content with User Interface;*
13. *Prototype the human interface;*
14. *Undertake content management;*

15. *Undertake market analysis;*
16. *Undertake testing of Website.*

In addition, the following subtasks are also proposed by Web OPEN (Haire, Henderson-Sellers & Lowe 2001):

1. *Choose Architectural Pattern for Website* (subtask of Create a System Architecture);
2. *Create the UCD role model* (subtask of Design User Interface);
3. *Create the UCD task model* (subtask of Design User Interface);
4. *Create the UCD content model* (subtask of Design User Interface).

The tasks listed above indicate clear focus on assisting with the production of deliverables within the software and creative design models. There is limited focus on deliverables within the business and domain models. The Web OPEN process does not address impact from the software model back into the business and domain models nor does it mention business process re-engineering. OPEN recommends UML (Rumbaugh, Jacobson & Booch 1998) and OML (2004) as techniques to support the creation of deliverables. Web OPEN provides no explicit focus on applying different business models as found with many Web developments.

The OPEN process framework does, however, provide a Phase within the Enterprise Life-cycle known as Business Reengineering. However, there was no available discussion as to how the Enterprise life-cycle and Project life-cycle should interact with respect to Web-based application development. Web OPEN is therefore classified as showing weak support for criterion 2 'Different business models'

7.4.3 Web OPEN: Activities

There is explicit focus in Web OPEN on a new Activity known as Web Site Management dealing with the bringing together "all the issues regarding the development, maintenance and management of a corporate Website which may or may not include access to back-end transaction processing systems". This specific focus within Web OPEN shows strong support for criterion 7 'Maintenance'.

There is no discussion within OPEN or Web OPEN of how teams are coordinated or should work together within a large Web engineering project. Indeed, there is no discussion with respect to small teams either. Web OPEN, therefore, shows no support for criterion 4 'Small development teams working in parallel on similar tasks'

7.4.4 Web OPEN: Roles

Web OPEN discusses the creative design and developer roles but there is no mention of the business or domain expert roles. OPEN or Web OPEN do not provide a definitive list of Web development team roles and responsibilities incorporating the business, domain software and creative design models. Web OPEN is, therefore, classified as showing partial support for criterion 3 'Multidisciplinary development teams'.

7.4.5 Web OPEN: Tools and Techniques

OPEN and Web OPEN are tool and technique independent although support for UML (Rumbaugh, Jacobson & Booch 1998) and OML (2004) is explicitly discussed within the OPEN literature.

7.4.6 Web OPEN: Support for the Criteria for a Web Engineering Process

Table 17 summarises the analysis of Web OPEN support for the criteria for a Web Engineering Process.

No.	Support	Comments
1. Short development life-cycle times	*Weak*	The OPEN framework requires process engineers to create a process instance particular to their project or organization. There is, therefore, a strong dependency on a skilled process engineer, Cockburn level 3, to ensure that the process is sufficiently tailored to deal with the time-to-market pressures experienced in Web engineering. It is unlikely that most Web projects will have access to such a skilled process engineer.
2. Different business models	*Weak*	The Web OPEN process does not address impact from the software model back into the business and domain models nor does it mention business process re-engineering. The OPEN process framework does however provide a Phase within the Enterprise Life-cycle known as Business Reengineering.
3. Multidisciplinary development teams	*Partial*	Web OPEN includes the creative design and developer roles but there is no mention of the business or domain expert roles.
4. Small development teams working in parallel on similar tasks	*None*	There is no mention of how teams are coordinated or work together within a large Web engineering project.
5. Analysis and Evaluation	*Partial*	The OPEN framework includes business modelling but it is not clear how this relates to business analysis. Web OPEN increases the focus on designing the user interface, with a number of sub tasks based on Usage Centred Design. Web OPEN does not mention Evaluation or the end-users of the proposed deliverables during development. However, OPEN provides a number of work products for Usability Testing, although the author could find no discussion of where, how or when these should be applied in Web development.
6. Requirements and Testing	*Strong*	OPEN provides a focus on requirements engineering and Web OPEN provides specific focus on testing through the Web Site Testing Task.
7. Maintenance	*Strong*	There is explicit focus in Web OPEN on a new Activity known as Web Site Management dealing with the bringing "together of all the issues regarding the development, maintenance and management of a corporate Website

		which may or may not include access to back-end transaction processing systems"

Table 17. Web OPEN Support for the Criteria for a Web Engineering Process.

7.5 Summary

This chapter used the criteria for a Web engineering process, see section 3.4 for more detail, to evaluate support by a number of Web engineering processes using the available literature. This chapter does not attempt an empirical evaluation of these processes in practice and, while this is desirable, it would be extremely difficult to do. Table 18 summarises the support shown by each of the four processes evaluated for the criteria for a Web engineering process. The four Web engineering processes evaluated were 'Collaborative Web Development' (CWD), 'Crystal Orange Web' (COW), the extensions proposed by Rational and Context Integration to the Rational Unified Process (Extended RUP) and the proposed extensions to OPEN (Web OPEN).

No.	CWD	COW	Extended RUP	Web OPEN
1. Short development life-cycle times	*Partial*	*Strong*	*Weak*	*Weak*
2. Different business models	*None*	*Weak*	*None*	*Weak*
3. Multidisciplinary development teams	*Partial*	*Strong*	*Partial*	*Partial*
4. Small development teams working in parallel on similar tasks	*Weak*	*Weak*	*Weak*	*None*
5. Analysis and Evaluation	*Weak*	*Weak*	*Partial*	*Partial*
6. Requirements and Testing	*Weak*	*Partial*	*Strong*	*Strong*
7. Maintenance	*Partial*	*Partial*	*Partial*	*Strong*

Table 18. Summary of CWD, COW, Extended RUP and Web OPEN Support for the Criteria for a Web Engineering Process

Analysis of table 18 indicates the following with respect to other Web-based development approaches support for the criteria for a Web engineering process:

1. **Short development life-cycle times**. With the exception of Crystal Orange Web, the commercial Web-engineering processes evaluated showed poor support for assisting with the time-to-market pressures experienced in Web-based projects. COW is the only process that explicitly addresses the crucial criterion of short development life-cycles;
2. **Different business models**. There is clearly a need for stronger support for different business models and business process re-engineering in the commercial Web engineering processes evaluated. If organisations are to harness the full benefits of e-business initiatives then Web-based development processes need to support the

different business models that are found in Web engineering that reflect and provide explicit support for impact back from the software model into the business and domain models;

3. **Multidisciplinary development teams**. With the exception of Crystal Orange Web the commercial Web-engineering processes evaluated showed partial support for the multidisciplinary nature of Web development teams. COW is the only process to incorporate the wide range of development roles required in Web engineering including business and domain experts;
4. **Small development teams working in parallel on similar tasks**. None of the processes provide a mechanism to support scalability to a number of small teams working in parallel. There is a clear need for further research assisting with scaling processes for Web engineering to large development teams;
5. **Analysis and Evaluation**. There is a need for a stronger focus on addressing the customer community view within commercial Web engineering processes evaluated. In particular the Web engineering processes evaluated will all benefit from greater focus on end-user participation throughout development and evaluation phases;
6. **Requirements and Testing**. With respect to requirements and testing, the extensions to traditional software engineering processes provide stronger support because of their foundations;
7. **Maintenance.** Like requirements and testing, maintenance has stronger support in the Web engineering processes that extend traditional software engineering processes.

This chapter compared four other commercial Web engineering processes against the criteria for a Web engineering process. The next chapter describes and then compares a Fortune 500 financial service sector company's in-house process that is used to develop Web-based projects and all other company technology projects, against the criteria for a Web engineering process. The chapter then discusses the results of a company sponsored survey of stakeholders using the in-house process on projects where time-to-market pressures are critical.

8 Developing Web Applications in a Fortune 500 Financial Service Sector Company

The Agile Web Engineering (AWE) Process, see chapter 6, was developed during 2001 as a Web-based process that specifically addresses the criteria for a Web engineering process, summarised in table 19. From October 2001 to September 2002, the author undertook a one year Ph.D. Internship with a Fortune 500 global financial services company with the goal of exploring the use of AWE in a commercial environment. The company was listed as one of the world's top 50 financial services companies by revenue in the July (2002) edition of Fortune Magazine, and is within the 30 most profitable financial services organisations in the world (Fortune 2003).

No.	Criteria for a Web Engineering Process
1.	Short development life-cycle times
2.	Different business models
3.	Multidisciplinary development teams
4.	Small development teams working in parallel on similar tasks
5.	Analysis and Evaluation
6.	Requirements and Testing
7.	Maintenance

Table 19. The Criteria for a Web Engineering Process

The author started work as an IT architect in the IT strategy department, in return for the opportunity to explore his research ideas in a commercial setting. While there was no guarantee that the company would adopt any of the author's ideas, the company seemed very positive and encouraging of the author's research ideas and of the proposed Internship approach. In order to ensure that the results were going to be publishable, it was agreed to refer to the company and employees in any publications anonymously.

The first section of this chapter describes and compares the company's in-house process that is used to develop Web-based projects and all other technology projects against the criteria for a Web engineering process. The second section discusses the results of a company sponsored review of the current in-house process before AWE's first commercial pilot on a retail Internet banking application, described in chapter 9. The review of the in-house process was carried out through a survey to establish how the company, with extensive experience of software development, was coping with the changing demands of developing Web-based applications and other software projects where time-to-market pressures are a major driver. The last section summarises the conclusions that can be drawn regarding the company's in-house process, its suitability to Web engineering, and the opinions held by a sample of project development stakeholders with respect to the in-house process.

8.1 In-house Process Support for the Criteria for a Web Engineering Process

In this section the company's in-house process is described using the process elements discussed in section 2.2. As references to each of the criteria for a Web engineering process, summarised in table 19, surface through the description of the in-house process elements, they are given a ranking to show how the in-house process supports each of the criterion for a Web engineering process. The evidence and a rationale behind this ranking are also presented.

The author approached the technology department who owned the in-house process seeking information regarding the company's process. As a result of this engagement the only information resource offered, was to guide the author to use the in-house process database. The in-house process database is a Notes Domino (2004) database of all information relating to the company's in-house process, including workflows and document templates. This database is made available to all employees within the technology division of the company. The primary source of information, used within this chapter, regarding the company's in-house process was extracted from the company's in-house process database.

In order to ensure that the anonymity agreement between the financial service sector company and the author is not compromised, some of the organisation specific words and terminology have been replaced with generic prose. The rest of this section describes the in-house process in sections 8.1.1-8.1.5 using the process elements discussed in section 2.2 and concludes with a summary of the company's in-house process support for Web engineering in section 8.1.6.

8.1.1 Process Life-cycle and Phases

The financial service sector company's in-house process is a plan-driven mainframe centric software development process, influenced primarily by the Stagewise (Benington 1956) and Waterfall (Royce 1970) models. The in-house process is divided into two separate processes, one covering the business projects and one covering technology projects, see figure 27.

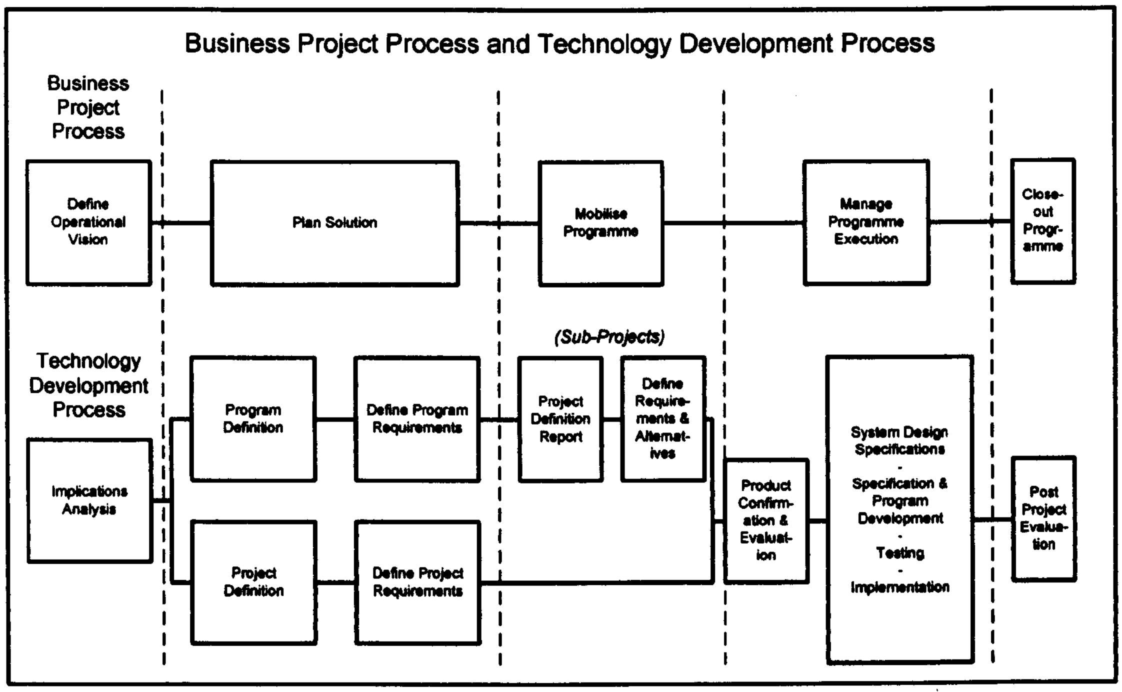

Business Project Process and Technology Development Process
Business Project Process
Define Operational Vision
Plan Solution
Mobilise Programme
Manage Programme Execution
Close-out Programme
(Sub-Projects)
Technology Development Process
Implications Analysis
Program Definition
Define Program Requirements
Project Definition Report
Define Requirements & Alternatives
Project Definition
Define Project Requirements
Product Confirmation & Evaluation
System Design Specifications - Specification & Program Development - Testing - Implementation
Post Project Evaluation

Figure 27. In-house Company Business Project Process and Technology Development Process. Reproduced from In-house Documentation

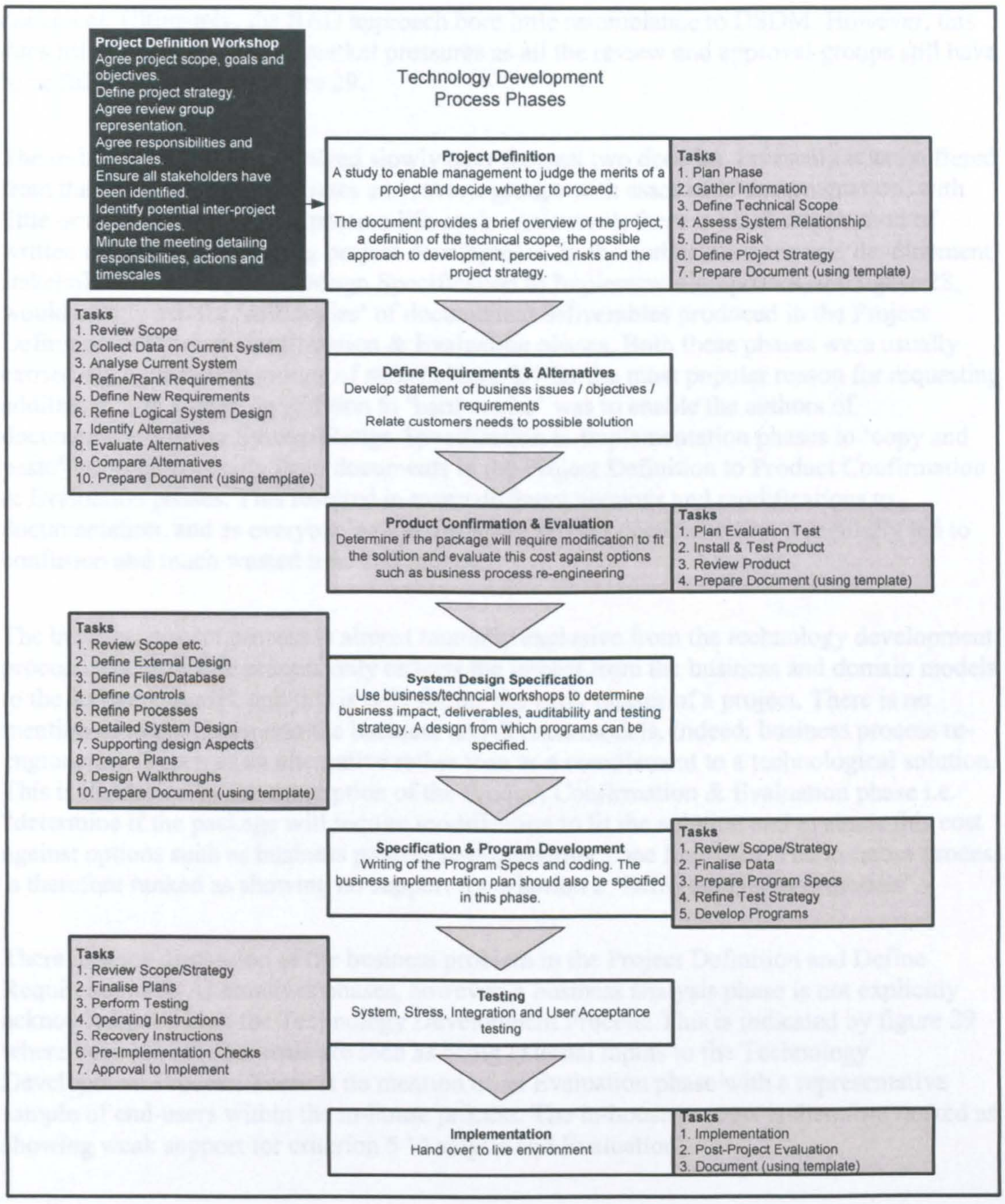

Project Definition Workshop
Agree project scope, goals and objectives.
Define project strategy.
Agree review group representation.
Agree responsibilities and timescales.
Ensure all stakeholders have been identified.
Identify potential inter-project dependencies.
Minute the meeting detailing responsibilities, actions and timescales
Technology Development Process Phases
Project Definition
A study to enable management to judge the merits of a project and decide whether to proceed.
The document provides a brief overview of the project, a definition of the technical scope, the possible approach to development, perceived risks and the project strategy.
Tasks
1. Plan Phase
2. Gather Information
3. Define Technical Scope
4. Assess System Relationship
5. Define Risk
6. Define Project Strategy
7. Prepare Document (using template)
Tasks
1. Review Scope
2. Collect Data on Current System
3. Analyse Current System
4. Refine/Rank Requirements
5. Define New Requirements
6. Refine Logical System Design
7. Identify Alternatives
8. Evaluate Alternatives
9. Compare Alternatives
10. Prepare Document (using template)
Define Requirements & Alternatives
Develop statement of business issues / objectives / requirements
Relate customers needs to possible solution
Product Confirmation & Evaluation
Determine if the package will require modification to fit the solution and evaluate this cost against options such as business process re-engineering
Tasks
1. Plan Evaluation Test
2. Install & Test Product
3. Review Product
4. Prepare Document (using template)
Tasks
1. Review Scope etc.
2. Define External Design
3. Define Files/Database
4. Define Controls
5. Refine Processes
6. Detailed System Design
7. Supporting design Aspects
8. Prepare Plans
9. Design Walkthroughs
10. Prepare Document (using template)
System Design Specification
Use business/techncial workshops to determine business impact, deliverables, auditability and testing strategy. A design from which programs can be specified.
Specification & Program Development
Writing of the Program Specs and coding. The business implementation plan should also be specified in this phase.
Tasks
1. Review Scope/Strategy
2. Finalise Data
3. Prepare Program Specs
4. Refine Test Strategy
5. Develop Programs
Tasks
1. Review Scope/Strategy
2. Finalise Plans
3. Perform Testing
4. Operating Instructions
5. Recovery Instructions
6. Pre-Implementation Checks
7. Approval to Implement
Testing
System, Stress, Integration and User Acceptance testing
Implementation
Hand over to live environment
Tasks
1. Implementation
2. Post-Project Evaluation
3. Document (using template)

Figure 28. Overview of In-house Technology Development Process Phases and Associated Tasks. Reproduced from In-house Documentation

The in-house process is a one size fits all approach to building software systems regardless of the type of development activity or the size or scale of the project, the notable exception being technology projects with a total estimated cost of less than £100,000 and a man-day effort less than 50. These projects are exempt from using the Technology Development Process at the discretion of a senior manager. Within the Technology Development Process there exists a rapid application development (RAD) approach which merely involves combining the phases Define Requirements & Alternatives, Product Confirmation & Evaluation and System Design Specifications and their respective documented deliverables, see figures 27 & 28. The Rapid Application Development (RAD) approach was supposedly based upon DSDM (2004a). However, when investigated further the author discovered that there was little difference between the full-blown process and the RAD approach. The only observable difference was the amalgamation of a number of documents to one single document. Ultimately, the RAD approach bore little resemblance to DSDM. However, this does little to reduce time-to-market pressures as all the review and approval groups still have to be fully engaged, see figure 29.

The in-house process has evolved slowly over the past two decades. Primarily it has suffered from the addition of many phases and review groups with associated documentation, with little or no deletion from the process life-cycle. As a result there is much duplication of written documentation during project development. In the author's experience, development stakeholders in the System Design Specification to Implementation phases, see figure 28, would openly ask for 'soft copies' of documented deliverables produced in the Project Definition to Product Confirmation & Evaluation phases. Both these phases were usually carried out by different groups of stakeholders. By far the most popular reason for requesting additional 'soft copies' in addition to 'hard copies' was to enable the authors of documentation in the System Design Specification to Implementation phases to 'copy and paste' information easily from documents in the Project Definition to Product Confirmation & Evaluation phases. This resulted in many different versions and modifications to documentation, and as everyone would distribute project documentation it regularly led to confusion and much wasted time and money.

The business project process is almost mutually exclusive from the technology development process. The in-house process only reflects the impact from the business and domain models to the software model, and this is only during the early phases of a project. There is no mention of impact back into the business and domain models. Indeed, business process re-engineering is seen as an alternative rather than as a complement to a technological solution. This is illustrated by the description of the Product Confirmation & Evaluation phase i.e. "determine if the package will require modification to fit the solution and evaluate this cost against options such as business process re-engineering", see figure 28. The in-house process is therefore ranked as showing no support for criterion 2 'Different business models'.

There is some discussion of the business problem in the Project Definition and Define Requirements & Alternatives phases, however, a business analysis phase is not explicitly acknowledged within the Technology Development Process. This is indicated by figure 29 where business requirements are seen as being external inputs to the Technology Development Process. There is no mention of an Evaluation phase with a representative sample of end-users within the in-house process. The in-house process is therefore ranked as showing weak support for criterion 5 'Analysis and Evaluation'.

Within the technology development process there is explicit focus on the requirements and testing phases. The in-house process is therefore ranked as showing strong support for criterion 6 'Requirements and Testing'.

There is no mention of a maintenance phase in the technology development process. The Technology Development Process ends when the project goes live. The in-house process is therefore ranked as showing no support for criterion 7 'Maintenance'.

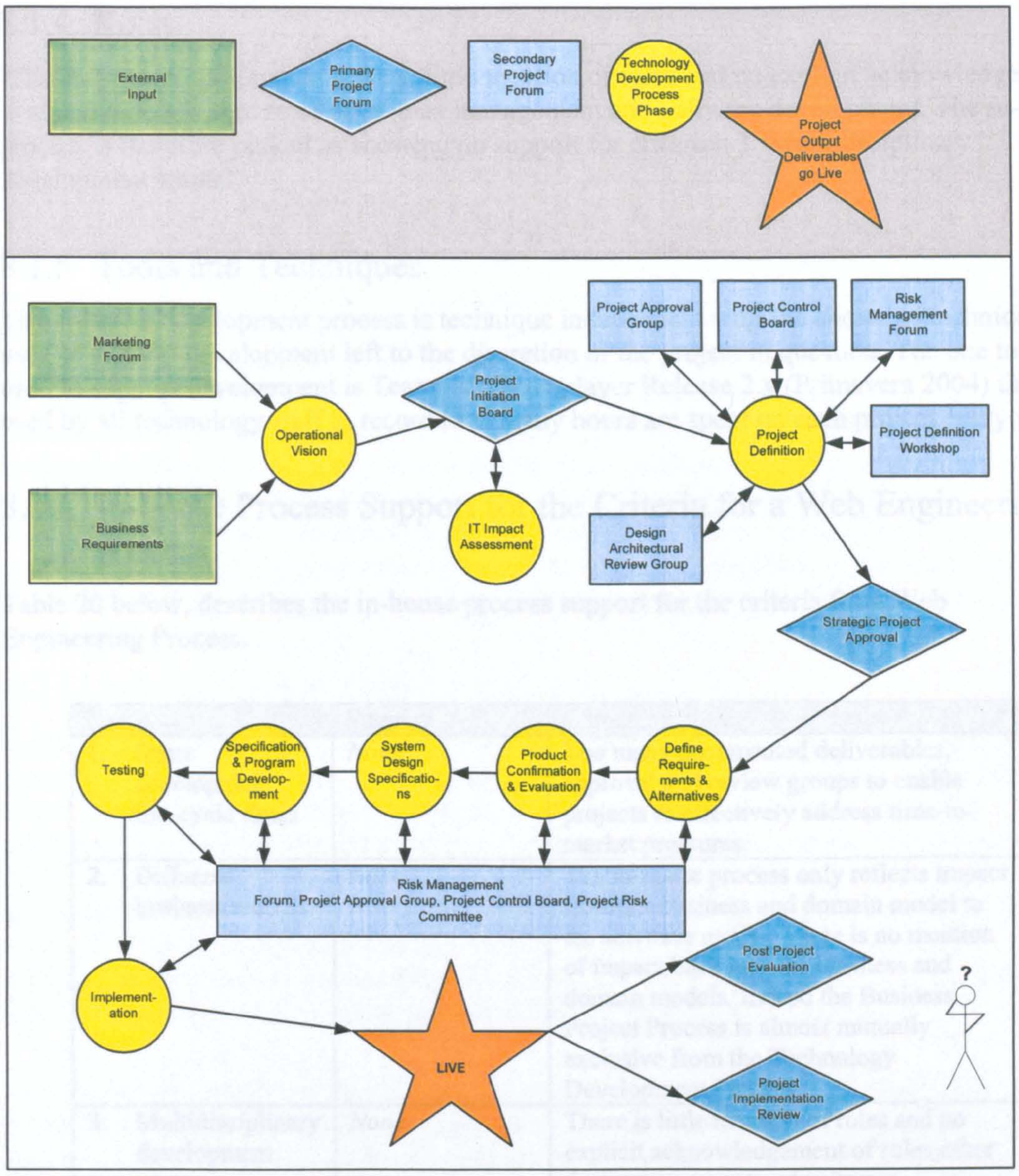

External Input
Primary Project Forum
Secondary Project Forum
Technology Development Process Phase
Project Output Deliverables go Live
Marketing Forum
Business Requirements
Operational Vision
Project Initiation Board
IT Impact Assessment
Project Approval Group
Project Control Board
Risk Management Forum
Project Definition
Project Definition Workshop
Design Architectural Review Group
Strategic Project Approval
Define Require-ments & Alternatives
Product Confirmation & Evaluation
System Design Specificatio-ns
Specification & Program Develop-ment
Testing
Risk Management Forum, Project Approval Group, Project Control Board, Project Risk Committee
Implement-ation
LIVE
Post Project Evaluation
Project Implementation Review
?

Figure 29. Technology Development Process Phases, Approval and Review Groups. Reproduced from In-house documentation, (key on the top added by the author)

8.1.2 Deliverables

The in-house process is too predictive in nature. The vast majority of projects are forced to produce too many documented deliverables and pass through too many review and approval groups to enable projects to effectively address time-to-market pressures. The in-house process is therefore ranked as showing no support for criteria 1 'Short development life-cycle times'.

8.1.3 Activities

Within the in-house process there are many review and approval groups that each project must interact with. There is no explicit mention of how different teams on the same project should communicate with one another. Although this is a required task during the Project Definition phase of the technology development process, see figure 28.The in-house process is therefore ranked as showing no support for criterion 4 'Small development teams working in parallel on similar tasks'.

8.1.4 Roles

Within the in-house process there is little mention of roles and no explicit acknowledgement of stakeholders within roles other than management and software development. The in-house process is therefore ranked as showing no support for criterion 3 'Multidisciplinary development teams'.

8.1.5 Tools and Techniques

The in-house development process is technique independent with the choice of techniques used to support development left to the discretion of the project in question. The one tool used to support development is Teamplay Teamplayer Release 2.x (Primavera 2004) that is used by all technology staff to record how many hours are spent on each project every week.

8.1.6 In-house Process Support for the Criteria for a Web Engineering process

Table 20 below, describes the in-house process support for the criteria for a Web Engineering Process.

No.	Support	Comments
1. Short development life-cycle times	*None*	Too many documented deliverables, approval and review groups to enable projects to effectively address time-to-market pressures.
2. Different business models	*None*	The in-house process only reflects impact from the business and domain model to the software model. There is no mention of impact back into the business and domain models. Indeed the Business Project Process is almost mutually exclusive from the Technology Development Process.
3. Multidisciplinary development teams	*None*	There is little mention of roles and no explicit acknowledgement of roles other than management and software development roles within the in-house process.
4. Small development teams working in parallel on similar tasks	*None*	There is no explicit mention of different teams on the same project should communicate with one another.
5. Analysis and Evaluation	*Weak*	There is some discussion on the business problem in the Project Definition and Define Requirements & Alternatives phases however a business analysis phase is not explicitly acknowledged. There is no mention of an Evaluation phase with a representative sample of end-users.
6. Requirements and Testing	*Strong*	There is explicit focus on the requirements and testing phases.
7. Maintenance	*Weak*	No explicit mention of maintenance issues outside the software model. However, a number of documented deliverables are produced within the software models. The Technology

		Development Process ends when the project goes live.

Table 20. In-house Process Support for the Criteria for a Web Engineering Process.

8.2 Pre-AWE Pilot Survey

During 2001/2002 a number of observations were made within the company regarding the suitability of the in-house software development process. In particular concern was raised regarding the in-house process's applicability in dealing with time-to-market pressures, particularly those associated with projects requiring the introduction of new software and hardware infrastructure, such as Web-based projects. Given ten man-days to investigate these concerns and try to articulate them more clearly, the author and another employee within the company[7] were asked to investigate. Both employees decided to interview a small representative sample of the stakeholders involved in projects where time-to-market pressures were critical. The objective of this investigation was to see what stakeholders' perceptions of the in-house process were and to assist with the author's understanding of the challenges facing the company's in-house process with respect to time-to-market pressures.

8.2.1 Survey Methodology

During the last two weeks of February 2002 six stakeholders were interviewed who were typically involved in projects where time-to-market pressures were critical and new software or hardware infrastructure was introduced. The interviewees comprised two project managers, a representative of the company's telecommunications supplier, an in-house business customer, an operations manager, and a representative from the in-house department who 'owns' the company's in-house development process. Software developers were not included in the sample as they have no say and little direct involvement in influencing the in-house process through approval and review groups. In addition, software developers normally only become involved after the Project Definition phase.

The interviews were conducted in a qualitative manner using an in-depth one-to-one interview technique. Each interview took about an hour. All the answers were recorded on paper by both interviewers conducting the survey. After the interview, the answers were written together by both interviewers. In order to help ensure there would be no access to potentially politically sensitive information it was decided to present the results of the interviews anonymously by individual within the company itself and in this dissertation. The author felt that it was important to present the results anonymously internally in the company as the results were a reflection upon the company's process and not of any individual interviewee.

Due to the nature of each interviewee's role, the interviewers tailored each interview to suit the different responsibilities of each stakeholder. Appendix 6 gives an illustrative example of the types of questions asked. Each interview tried to elicit information regarding the subject's understanding of the company's process. The following topics were covered:

- Asking the interviewee to explain the company's process in detail. In order that the interviewers could assess the interviewee's level of understanding of the process;

[7] The other employee in question has given permission for this work to be discussed without specific reference to him. This is to enable the author to comply with the anonymity agreement between the company and the author.

- Investigating each interviewee's understanding of their role and involvement at each stage/phase in the company's process;
- Where relevant to the interviewee, questions pertaining to the Rapid Application Development (RAD) life-cycle of the company's process were asked, in order that the interviewers could assess the different views of how the RAD process was understood;
- Aspects of the company's process the interviewee considered to be successful;
- Aspects of the company's process the interviewee considered to be poor in practice;
- Asking each interviewee to identify the various stakeholders involved in a typical project using the company's process. This was to aid the interviewers' assessment of what the interviewees views of the company process were, and at what stage/phase within the in-house process life-cycle different stakeholders were required to be involved;
- Each interviewee was also asked to describe any known limitations to the company's process. This helped comprehend the aspects of the in-house process that were seen to be absent.

8.2.2 Survey Results

The answers given in this survey were felt to be too specific in terms of organisational terminology and in-house jargon to be presented in an open publication without compromising the identity of the company. In addition, attempts to make the specific responses anonymous were felt to render meaningful interpretation to third parties useless. The rest of this section therefore presents the analysis of the data gathered and the conclusions that were drawn from the pre-AWE Pilot survey in general terms.

8.2.3 Architectural Evolution

Minimal evolution has occurred within the company's current architecture over the past ten years. For example, the network capacity found in bank branches had not changed in the vast majority of instances over the past decade. As a result there is poor support in the in-house process for projects introducing new software and hardware to the enterprise. This is primarily due to the fact that most projects carried out during the 1990s had not needed to introduce new architectural designs, but had instead run on existing software and hardware infrastructure.

Over the past few years there has been strong inertia shown to minor evolutions in the company's process. As a result little focus on architectural issues that arise in Web-based technology projects, such as those associated with Internet applications, are found within the in-house process. This was the case due to the lack of change associated with the infrastructure used in mainframe-centric application development and stakeholders' inexperience with new technologies, particularly in Web-based development.

8.2.4 Stakeholders' Understanding of the In-house Process

Over and above the limitations particular to the company's process itself, this survey showed that the company's process seems to be poorly disseminated and understood by the relevant stakeholders. This was illustrated by the confusion and difference between interviewees' answers over when to engage different types of stakeholder during the life-cycle, and the purpose of stages/phases contained within the process life-cycle. The process fails to identify when to correctly engage stakeholders. For example, there is no mention of operations stakeholders who are crucial in successfully introducing new software and hardware infrastructure. Business representatives also seemed to have a poor understanding of the relevance and workings of the phases within the development process, even the phases where their participation was essential for the project to gain 'official approval' to progress to the next stage/phase of the process.

The in-house process literature only mentioned stakeholders within technology, i.e. management and software development stakeholders. The many contradictions amongst interviewees as to how the company's development process should be used illustrated the need for greater understanding of and education about the process. Indeed the interviewers, who had both recently joined the company in the past six months, had received no formal exposure through training or induction to the company's process. To illustrate:

- Interviewee 1 (representative from the company's telecommunications supplier) had no exposure to the process at all;
- Interviewee 2 (representative from the in-house department that 'owns' the company's in-house development process) stated that the process should be used as a framework, whereby Project Leaders should ignore or add to the process phases to suit the project needs. This interviewee also stated that a major drawback of the process is the fact that stakeholders do not understand or use the process in this fashion. It was also stated that too many stakeholders view process stages as set in concrete;
- Interviewee 5 (Development Project Manager) stated that many people in the company have a poor understanding of the process;
- Interviewee 6 (Operations Manager) also stated that whilst he is aware of the process, he did not fully understand it.

The fact that the in-house process contains poor stakeholder representation beyond internal bodies is a significant obstacle to project success. This is a critical problem when third parties are involved. For example, there is no official mention of a third party telecommunications supplier or any other third party stakeholder within the process. This is a severe problem, as the company's data network and telephonic infrastructure installation and maintenance is outsourced. Third party stakeholder involvement is essential in most projects introducing new technologies, such as Web engineering projects. Failure to represent third parties within the in-house process severely impacts the ability for projects introducing new technologies to successfully address time-to-market pressures.

8.2.5 Time-to-Market Pressures

The process is seen as being too bureaucratic and too slow. So much so, that it is not suitable for projects where time-to-market pressure is crucial to success. Often it can take three months to get to the approval stage before relatively small projects can be started. Development effort and time required for many projects is less than the effort and time required to obtain approval. Indeed, this was illustrated by one of the interviewees, who as part of a project team, designed, built, tested and put into production a software system, before the Project Definition was officially approved to proceed by the Strategic Project Approval group, see figure 29.

Another stated example of the bureaucratic nature of the in-house process is the time taken to cost the installation of an additional telephone line. Following this process takes about six weeks to obtain both price and installation date figures for this work before one can even place an order for an additional telephone line. Therefore, a Telecommunications Supplier's involvement significantly increases time-to-market. In contrast, during the same period in which the survey was conducted the same third party telecommunications supplier would guarantee installation of a new telephone line within 14 days to the consumer and small business market.

The company's process is not suited to infrastructure projects or projects where speed-to-market is important. The in-house process involves too much unnecessary documentation, and this can double the length of the project in the opinion of some of the interviewees. The company's process documentation primarily supports and suits legacy development activities. The predictive nature of one of the initial phases, where the project definition is agreed, also seems to add significant time delays. The company's process is a one-size-fits-all approach to building software that is poorly suited to the new types of development activity facing the company. It is also worth mentioning that the company's process was never designed to operate in an environment where 'buy before build' is the organisational strategy.

8.2.6 Pre-AWE Survey Conclusions: Preliminary Recommendations and Impact of Survey

The conclusions of the internal report (written by the author and another employee) recommended a 400 man-day project to revise the company's in-house process. The recommendations of this project were: to interview a wider range of stakeholders in order to capture and detail more information regarding the challenges facing the in-house process (40 man-days); review the current in-house process, categorise the types of projects being planned over the next few years, and evolve the process with different process life-cycles specific to each major project category (160 man-days); develop a plan for increasing process ownership and understanding internally and within the company's important third party suppliers (50 man-days); introduce training and education surrounding the new process approach on a specific category of projects and assess the impact and success of the overall recommended initiative (150 man-days). It was then anticipated that the training and education could be extended given an indication of success in the last recommendation.

A number of executive stakeholders reviewed this document and while many business and technology stakeholders were supportive of our recommendation there was no direct action as an outcome of this review.

8.3 Summary

This chapter evaluated the in-house company process within a Fortune 500 financial service sector organisation against the criteria for a Web engineering process in section 8.1. The ranking of the in-house company process support for the criteria for a Web engineering process is exceptionally poor, and weaker than the other processes evaluated in chapters 5, 6 and 7. The in-house process shows no support for criteria 1, 2, 3, 4 & 7, weak support for criterion 5 with only criterion 6 showing strong support. The evidence presented in this chapter indicates strongly that the company is facing the Web-based process challenges discussed in chapters 3 and 4.

Section 8.2 describes the pre-AWE pilot survey. The pre-AWE pilot survey showed that the in-house process was heavily bureaucratic and provided extremely limited support for addressing time-to-market pressures. In addition, the process is poorly understood by critical stakeholders and has had little evolution during the 1990s. When presented with evidence of the problems surrounding the in-house process and a suggested approach for addressing the in-house process problems no action was taken by the company.

The following chapter discusses the first commercial pilot of AWE on a retail Internet banking channel within the financial service sector company. It then presents the results of a second survey, the post-AWE pilot survey, to assess company stakeholders' perceptions of the AWE process. The chapter then uses Boehm & Turners (2003a) home ground analysis to validate the results of the post-AWE Pilot survey and to indicate whether or not an agile approach is best suited to Web engineering within the company.

9 AWE's First Commercial Pilot

The last chapter presented evidence to suggest that the financial service sector company employing the author during his Ph.D. internship is facing many of the Web engineering process problems identified in the first survey of Web engineering in practice, see chapter 3. The first section of this chapter discusses the first commercial pilot of AWE on a retail Internet banking channel within the financial service sector company. The second section presents the results of a second survey, the post-AWE pilot survey, of staff members within both the business and the technology sectors of the financial services company. The goal of the second (post-AWE pilot) survey was to assess company stakeholders' perceptions of the AWE process. The third and penultimate section uses Boehm & Turner's (2003a) home ground analysis to validate the results of the post-AWE Pilot survey and to indicate whether or not an agile approach is best suited to Web engineering within the company. This is followed by a chapter summary in section 9.4.

9.1 Evolving and Maintaining a Retail Internet Banking Application

From October to December 2001, the author approached a number of groups and individuals within the company to discuss Web engineering research ideas, primarily the AWE Process, to promote interest in sponsoring work to support the usage of AWE on a commercial project. This route had not led to any great success and the fact that AWE was not proven in a commercial environment seemed to be a key stumbling block for senior stakeholders who were approached to support the use of AWE on one of their commercial projects. In addition, with over a decade invested in the current development process, it was not surprising to find that proposals for change were being met with significant inertia.

During the internship year the author met a part-time M.Sc. IT student who was working in the specialist Web development area in the same company employing the author on his Ph.D. Internship. The M.Sc. student's full-time job was as systems analyst/programmer maintaining and evolving the latest retail Internet banking application to go live within the group in Europe. The M.Sc. student undertook her M.Sc. project, supervised by the author, which allowed her to combine the application of usability techniques being guided by the AWE process with her normal work. The rest of this section describes this first pilot of AWE in more detail.

9.1.1 The Application

The Web application developed and maintained by the M.Sc. student's team had gone live during 2001 for two banks within the group. During the development of this retail Internet banking application there had been three approaches taken to addressing usability issues:

1. Using throw away and evolutionary prototyping with a combination of business experts (end-user substitutes) and software engineers during the Specification & Program Development phase of the in-house development process, see figure 28;
2. Usability experts acting as end-user substitutes from a consultancy group specialising in software usability, after final integration testing;
3. Post implementation focus groups, involving software engineers and business experts asking actual end-users what they thought of their experiences with the live usage of the Web application.

Simply asking end-users what they think of an application (Nielsen 2001a) is often flawed because end-users often do not remember what makes them perform a task or series of actions. This can often result in end-user willingness to please the interviewer(s), and lead them to fail to properly articulate their real opinion regarding the system(s) being evaluated.

9.1.2 The Pilot

In January 2002, the pilot started using the AWE process focusing on improving the end-user experience during the evolution and maintenance of the retail Internet banking application. As the Web application was already live and in the maintenance and evolution phase of the project, the pilot started using the AWE process (McDonald & Welland 2001c) at the Evaluation Phase, as described in chapter 6.

Recent commercial (Murphy 2001, Nielsen 2001a, and Spool et al. 2002) and academic (Lowe & Eklund 2002, McDonald & Welland 2001c) literature has strongly advised those involved in usability testing, or the Evaluation Phase as it is known in AWE, to observe a representative sample of end-users carrying out tasks and to get them to 'think aloud' when carrying these tasks out. Given the author's experience with Think Alouds (Lewis & Rieman 1994) as a teacher and the M.Sc. student's familiarity with Think Alouds as a student at the University of Glasgow, the team decided that this technique would be a good one to support the Evaluation Phase. During 2002 the M.Sc. student carried out a series of Think Aloud evaluations over two person-days with the part-time assistance of the author, one person-day.

Selecting a sample size required the team to balance a number of issues, including cost, time-scales and a sample size of critical mass (Nielsen 2000b) to draw solid conclusions regarding the usability challenges of the Web application. Rogers (2003) categorises adopters of technology into:

- Innovators (2.5% of the total audience) are those who are the first to adopt;
- Early Adopters (13.5% of the total audience) are the second to adopt;
- Early Majority (34% of the total audience) are third to adopt;
- Late Majority (34% of the total audience) are fourth to adopt;
- Laggards (16% of the total audience) are last to adopt;

The AWE development team decided to select a sample size of twenty individuals, offering each subject ten pounds sterling for approximately 45 minutes of their time to carry out a Think Aloud experiment. A number of individuals were approached off the street outside the company's premises, and as a result the team managed to get 10 subjects to represent the general public, potentially covering all types of technology adopters. The development team also selected ten students at random from the course attended by the M.Sc. student to represent early adopters and early majority technology adopters.

It is important to note that AWE is not just a usability focused Web engineering process; AWE accepts that usability alone is not the only metric for project success. AWE encourages validation of the deliverables to ensure the project is achieving its business objectives. Costs, timescales, achieving business objectives, internal company politics all have to be balanced to ensure project success. AWE assists development teams in addressing this balance through a number of features described in more detail in chapter 6.

The Think Alouds were carried out on the test system that was used for development within the company. Each subject was asked to perform six tasks that were representative of typical tasks undertaken by customers. A summary of the results of this first evaluation is shown in the top part of Table 21. No special usability lab was available or required for the team to carry out the Evaluation Phase.

Each interviewee was asked to fill in a short questionnaire immediately after the Think Aloud experiment to give subjective feedback on their experience of using the system. Subjects were asked to respond to a number of questions on a five point categorical scale, ranging from strongly agree to strongly disagree. More than 50% of responses agreed with the statement: 'using the system was very frustrating' and more than 50% disagreed with the statements about the system: 'it was easy to learn how to use it', 'easy to find my way about' and 'very pleasant to work with'. Perhaps the most significant response was that only 10% of respondents agreed with the statement that 'the system is very pleasant to work with'.

Task		1	2	3	4	5	6
First Evaluation	Completion rate	80%	25%	35%	45%	20%	75%
	Average time to complete successfully	2 min. 36 sec.	2 min. 12 sec.	4 min. 36 sec.	4 min. 36 sec.	5 min. 30 sec.	2 min. 12 sec.
Second Evaluation	Completion rate	95%	50%	85%	80%	75%	90%
	Average time to complete successfully	1 min. 48 sec.	2 min. 42 sec.	2 min. 54 sec.	3 min. 48 sec.	3 min. 24 sec.	2 min. 12 sec.

Table 21. Results of the First and Second Set of Think Aloud Evaluations

The findings were written up in an internal report that was presented to various stakeholders within the company. The writing up exercise took four person-days in total, bringing the total number of person-days to seven for the Evaluation phase. The total cost of the Evaluation phase including person-days and the £200 given to the 20 experimental subjects came to £2,500. The authors of the internal report then set about promoting this new approach to the company's Web development process and fed it back into the development team stakeholders for use in AWE's Business Analysis phase.

The report highlighted a number of usability issues, some of which had previously been identified from the earlier usability approaches described in section 9.1.1. However, the majority had not arisen from any of these earlier efforts. The development team produced a list of thirty-seven usability issues with the Web application. One month later business stakeholders produced a report detailing the results from data from helpdesk calls to the helpline supporting the project for the first six months since going live. The company's executive responsible for Internet and Telephonic Channels for the group in Europe stated that one issue identified in the report, if it had been addressed during development, would have saved the addition of four extra helpdesk staff supporting the helpline. He therefore communicated to other stakeholders within the company his personal endorsement and support for the rest of the pilot.

The first Evaluation phase indicated that only 47% of tasks were being completed successfully. From Table 21 it can be seen that completion rates for individual tasks were highly variable from 20% to 80%. These results were close to the figures gained from similar experiments carried out by Nielsen (2001b) using ecommerce Web sites. Clearly there was huge room for improvement and cost reduction to support the live Web application.

The development team then set about addressing some of the usability issues using AWE that were not being addressed by colleagues using the existing maintenance process. During AWE's Business Analysis Phase the team decided to document proposed solutions using QOC (Questions Options Criteria) diagrams from Design Space Analysis (MacLean et al. 1991). The developers only had a limited time to implement changes to the application and so only a small number of these issues were to be addressed during the pilot. However, all the issues were documented for future reference. In AWE's Requirements Analysis phase, the team chose the issues to address based on a number of criteria: the business case for the change; the simplicity of the change and the effort required to make it; the consistency of the proposed change with the existing system; and its impact on usability. To a large extent the team identified changes that would give 'quick wins' in usability terms, and also to ensure that at least one problem from each task was addressed.

Many of the issues raised were trivial to address in the development phases. For example, the HTML fields for entering monetary amounts were blank with no surrounding text. This resulted in a number of users entering a pound sign (£) before the monetary amount, which caused one of the server tiers to throw an error. The team attempted to resolve this issue by placing a pound sign (£) to the left of these fields, and writing some JavaScript to ensure that only valid numbers were sent from the client. During the second Evaluation Phase this problem ceased to occur. Other examples of changes involved: improving the feedback to the user, particularly with respect to null errors resulting from end-user misunderstanding of parameter settings; modifying the navigation structure (changing the location of certain application functions); and explaining banking terminology more clearly to the end-user.

During the pilot, the team addressed sixteen of the usability issues identified previously from the first series of Think Alouds. This was all that was possible within the given time frame, passing through the Design, Implementation and Testing Phases. The team then carried out the same series of Think Alouds with a different representative sample of end-users using the same approach described previously. The results of this second evaluation are shown in Table 21. Subjects were also asked to complete the same post Think Aloud questionnaire as the original group of subjects.

To the amazement of many sceptical colleagues the development team had managed to increase task completion rate from 47% to 79% without addressing the majority of the documented issues due to time constraints. It was also interesting to note that five of the six tasks now had completion rates above 75%, with tasks 3, 4 and 5 all showing major improvements. Even task 2 had shown a significant improvement although, obviously, this would need careful consideration in any further enhancement of the system, especially as the average completion time for this task increased.

Analysis of the questionnaires also indicated that the changes made during the pilot affected the subjects overall assessment of the system. Only 10% of subjects felt that 'using the system was very frustrating', while at least 60% agreed with the statement 'it was easy to learn how to use the system', only 10% of the subjects disagreed with the statement that 'the system was very pleasant to work with'. Navigation had improved; 40% of subjects disagreed with the statement: 'it was easy to find my way about', compared with 70% disagreement during the first series of Think Alouds. However, this was obviously still a significant problem area that needed to be addressed.

The iteration of the AWE process from initial Evaluation through to Testing took a total of sixteen person days:

- Evaluation (7 person days) – carrying out Think-Alouds and analysing results;
- Business Analysis (4 person days) – analysing usability issues (37) and creating QOC diagrams;
- Requirements Analysis (1 person day) – selecting the changes (16) to be implemented;
- Design, implementation and testing (4 person days) – developing and testing the selected changes.

The second evaluation required a further 7 person days to conduct the Think-Alouds, analyse and write up the results.

The total cost of the pilot study was £8,000, including person days and payments to experimental subjects; this comprised an iteration of the AWE development life-cycle plus the additional Evaluation Phase. For this modest outlay, the AWE pilot identified a large number of usability issues, addressed just under half of these issues, and demonstrated that a development team working on a real world commercial project could achieve a significant improvement in end-user performance and satisfaction.

9.1.3 AWE Pilot Status

This pilot raised the profile of AWE within the company. After the pilot the senior IT officer at board level for the group globally, gave his endorsement to the pilot and sent an encouraging and supportive email to the development team. In conclusion, the first pilot proved very positive. However, it is difficult to make exhaustive conclusions regarding the commercial applicability of AWE in its entirety after only one pilot study. However, the pilot did establish that the Agile Web Engineering (AWE) Process focus on end-user involvement through its Evaluation phase has enabled an increase in end-user task completion rate from 47% to 79% on a commercial project. The introduction of AWE's Evaluation phase with a representative sample of end-users and its linkage to AWE's Business Analysis Phase has been shown to be successful on its first commercial trial. Other literature has stated the benefits for collaboration between business experts, domain experts, software engineers and creative designers within Web development processes. Primarily this involves verifying the deliverables (Constantine & Lockwood 1999, Lewis & Rieman 1994, McDonald & Welland 2001c and Ward & Kroll 1999). This pilot demonstrated the additional benefit of an Evaluation Phase with a representative sample of end-users to validate the deliverables as demonstrated through the AWE process. Thus, our business sponsors have seen the benefit of validating Web-based deliverables with end-users in addition to just verifying Web-based deliverables within a multidisciplinary development team.

9.2 Post-AWE Pilot Survey

At the end of the one year Ph.D. internship the author had established that the organisation, see chapter 8, was facing the same Web development process challenges, discussed in chapters 3 and 4, that the AWE process was designed specifically to address. In addition, AWE had been successfully piloted to help address the correct focus on the customer community view for one of the bank's Internet facing Web-based applications. Yet despite these achievements formal adoption or heavy influence of the AWE process for building Web-based systems within the company still remained elusive. In order to try and understand the hurdles to the successful adoption of AWE within the company, the author carried out a survey, known forthwith as the post-AWE pilot survey to help illuminate these issues. The rest of section 9.2 describes the post-AWE pilot survey in detail.

9.2.1 Survey Methodology

The post-AWE Pilot survey was carried out in September 2002. Eight different stakeholders, referred to respectively as interviewees 1-8 were involved. Each interviewee had been involved during the promotion of the AWE Process and had read the AWE Process technical report (McDonald & Welland 2001c). The survey was conducted in a qualitative manner using an in-depth one-to-one interview technique. Each interview took approximately three quarters of an hour. Each interviewee was asked 16 questions, see appendix 7. All of the answers were recorded on paper by the interviewer and were then tabulated[8], see appendix 7. In order to help ensure anonymity of the company and the employees the results of the interviews are presented anonymously. There was not enough time to present the results of this survey within the company itself, this dissertation therefore serves as the first airing of this survey in published form.

9.2.2 Interviewee Sample Demographics

The interviewees were drawn from both the technology and business areas of the company. In total the interviewees had amassed 25 years experience in Web application development although not all within the company. The average interviewee involvement in Web development was approximately 4 years. Most of the interviewees had gained their formal training in traditional IT development, primarily mainframe-based software development. Interestingly, interviewee 1 did not consider 3 years Web-based development experience to count as experience in IT! Analysis of the current responsibilities of the interviewees showed that our sample represented stakeholders from business, software development, operational support, IT security, design and architectural roles.

9.2.3 Comparing Web-based and Traditional IT Development Experience

In response to question 4, 'In your experience do you notice any major differences between Web-based development and tradition IT Projects?' six of the eight interviewees explicitly noted a difference between Web-based and traditional IT development, with the other two interviewees commenting on the lack of difference within the company itself. More detailed analysis of the interviewees' answers to question 4 reveals the following common observations with respect to differences between Web-based development and traditional software development:

- Time-to-market pressures are greater on Web-based projects, interviewees 1, 3, 4 and 8;
- There is a differences in the type of development approach and the stakeholder skills required to succeed on Web application development projects, interviewees 1, 3, 4, 6, 7 and 8;
- Poor in-house skills for Web application development in comparison to traditional IT development in both business and technology departments, interviewees 1, 5 and 6;
- Lack of mature architectural patterns and increased architectural complexity in Web-based development, interviewees 4, 6, 7 and 8.

[8] The answers to question 16 are not presented as they were considered too specific in terms of organisational terminology to present without compromising the anonymity agreement between the author and the company.

The first two points strengthen the argument that the company needs a Web development process that addresses time-to-market pressures and supports multidisciplinary stakeholders to successfully develop Web applications. The last two points indicate a lack of skills and experience within the company in the field of Web engineering.

9.2.4 Perception of the Current In-House Process for Web-based Application Development

The interviewees' answers to question 6, 'What do you think of the organisation's current development process and its applicability to Web-based development?' highlight the following issues with the company's in-house process:

- Too slow, does not address time-to-market pressures, interviewees 1, 3, 5 and 7;
- Too heavy, bureaucratic and costly, interviewees 1, 2, 3, 4, 5, 7 and 8;
- Not appropriate for Web development, interviewees 1 and 4;
- Too predictive in nature, not adaptive enough, interviewees 1, 2, 5, 6, 7 and 8;
- Suffers from inertia, interviewees 2 and 8;
- Does not address the relevant stakeholders and is poorly understood by many stakeholders, interviewees 1, 2, 4, 5, 6 and 7.

The first two points strongly support the argument that the current company in-house process is not lightweight enough to address the time-to-market pressures required to successfully develop Web applications. The third, fourth and fifth points indicate that the in-house process is too rigid and predictive, or not agile enough, to support the adaptive nature of Web engineering. Indeed, interviewee 4 stated "I would be amazed if we could kick a Web-development project off and follow our process without serious pain". The last bullet point strengthens the observation regarding the poor suitability of the company process in addressing the more diverse multidisciplinary nature of stakeholders required to successfully develop Web applications.

9.2.5 Perceptions of AWE

All interviewees responded "Yes" to question 7, 'Have you read the AWE Process Technical Report?'; this question was included simply as a check on interviewees' awareness of AWE. Question 8 'What do you perceive as the strengths of the AWE Process?' and question 9 'What do you perceive as the weaknesses of the AWE Process?' aimed to derive the perceived strengths and weaknesses of AWE. AWE was observed by the interviewees to have a wide variety of strengths. AWE was perceived by the interviewees to be able to assist with time-to-market pressures and to improve cost reduction on Web engineering projects, interviewees 1 and 8. The adaptive nature of AWE was observed by interviewees 3, 4 and 8 as an advantage. AWE's specific focus on Web development was seen as a strength by interviewees 4 and 7. Not surprisingly given the success of the AWE's first commercial pilot, which increased end-user task completion on a Retail Internet banking application from 47% to 79%, see section 9.1 for more details, six of the eight interviewees mentioned the involvement of end-users during AWE's Evaluation phase as being a strength. Other features of AWE that were perceived as strengths included:

- Stakeholder involvement at every phase of the life-cycle, interviewees 2 and 5;
- Flexible and Adaptive nature, interviewees 3, 4 and 8;
- Collaborative nature, interviewee 5;

- AWE's approach to managing risk, interviewee 5;
- Technology, Tool and Technique independence, interviewees 6 and 7;
- Empowerment, interviewee 8.

The features of AWE listed in the bullet points above have a dependency upon the culture of the company and its stakeholders. While they were observed by some interviewees to be a perceived strength, they were also perceived to be a weakness of AWE. While interviewee 8 considered empowerment an advantage of the AWE process, interviewees 1, 2 and 6 considered empowerment in the hands of lower skilled developers to be a potential weakness. Interviewees 3, 7 and 8 observed the need for quality development stakeholders to successfully adopt AWE. The need for good quality developers in order to adopt AWE is not a surprise, as it is a common observation with respect to the successful adoption of agile processes (Boehm & Turner 2003a).

Half of the interviewees observed a need for a change in company culture before AWE could be adopted. The fact that AWE was not fully proven was also observed to be a weakness, interviewees 4 and 5. Other observed weaknesses included poor understanding of the differences between the domain and business models, interviewee 6, and an observation by interviewee 7 that AWE was not scalable to large projects. One of the major perceived weaknesses of AWE was its poor alignment with the company's culture. The following quotes from interviewees emphasise this point:

- "Another weakness is that AWE requires a cultural shift, which is easier to write about in paper than to happen in reality", interviewee 4;
- "Needs cultural and organisational change in order for an organisation like this to adopt it", interviewee 5;
- "Need to have a company that is forward thinking and is willing to change. Need to have an employee culture where employees will want to change. Not suitable within the financial service sector", interviewee 8.

In response to question 10, 'What obstacles/environmental conditions do you see to adopting the AWE Process within this organisation?' six of the eight interviewees believed that inertia was the biggest hurdle to the company's official adoption of AWE for Web engineering projects. Interviewees 2, 3 and 8 also mentioned ignorance as being a major barrier to AWE. The company's culture was again observed by interviewees 4, 5, 7 and 8 to be a major hurdle as AWE would require a significant change in culture before it could be successfully adopted. Other observed obstacles/environmental conditions to the adoption of AWE included: the company's 'buy before build' strategy and the predictive nature of the company's approach to development, interviewee 5; poor senior management buy-in, interviewees 2 and 7; the impact of the learning curve required for any new process such as AWE may have on project deadlines, interviewees 4 and 5; and the fact that AWE was specific to Web application development and not designed for a wider classification of project types, interviewee 6.

9.2.6 Perceptions of AWE's First Commercial Pilot

The next two questions (11 and 12) explored the perceptions of the *AWE Pilot*[9] carried out by the author and a colleague. The following points highlight the interviewees' impressions of the AWE Pilot:

- Six of the interviewees believed that the AWE Pilot was a valuable and beneficial piece of work. Interviewees 6 and 8 confessed to not knowing enough about the Pilot to comment;
- Interviewees 1, 2 and 7 commented that the Pilot focussed on important aspects of Web engineering that the company's in-house process does not address;
- Interviewees 2, 3 and 7 mentioned that the AWE Pilot showed how to incorporate usability into the company development process;
- The AWE Pilot was observed to be cost effective, interviewees 5 and 7;
- Interviewee 7 was quoted as saying that the AWE Pilot "Highlighted and confirmed reasons for using the AWE Process".

We followed this by asking 'Why do you think this[10] was not carried forward and formalised within our Development Process?'. Interviewee 1 stated that the business would be happy to adopt the AWE process features explored in the pilot, and that the problem lay within the company's technology arm. Three of the interviewees mentioned that adopting the features of AWE explored in the Pilot would require change and that inertia was one of the main hurdles. Four of the interviewees mentioned issues surrounding buy-in from senior management as a reason for the lack of willingness to adopt AWE. Interviewees 5 and 7 observed a lack of focus and ownership of the development process in general as a reason for the features of AWE successfully proven in the Pilot not being formally adopted.

9.2.7 Reflections on the In-House Development Process

The final group of questions were concerned with the current in-house development process and provided additional verification of the findings of our pre-AWE Pilot survey. Question 13 was: 'What changes, if any, would you like to see to our development process?' Four of the eight interviewees mentioned that they would like to see the adoption of agile processes. Interviewee 2 expressed a wish for the company to adopt the AWE process. Interviewees 1, 3 and 4 mentioned the need for a development process approach that helps projects address time-to-market pressures. Four of the interviewees mentioned the need for streamlining to remove the bureaucratic nature of the current in-house process. Interviewees 5, 6 and 7 expressed concern surrounding the one-size fits all nature of the current in-house process and expressed a desire for processes specific to different types of development activity. In addition, interviewee 5 observed a need for the current in-house process to focus more on usability issues as carried out during the AWE Pilot. Interviewee 8 again reiterated a desire for more empowerment within the current process. The need for greater training and education was also mentioned by interviewee 4.

We then asked two questions about the perceived strengths and weaknesses of the organisation's process. With respect to the strengths of the current in-house process interviewee 1 mentioned the benefits of the risk averse nature of the in-house process, although acknowledged that this was also a weakness. Five of the eight interviewees

[9] *AWE Pilot* replaces the in-house company name for this project, used in question 11, to ensure compliance with the anonymity agreement between the company and the authors.
[10] This question refers to the features of AWE successfully explored during the AWE Pilot.

mentioned the rigid structure and document-centric nature of the in-house process as an advantage. It was also perceived to be good for legacy systems development where the technologies were well understood and familiarity with similar projects had occurred in the past. Other advantages of the current in-house process mentioned by interviewees included:

- "That they have a process at all. ... That it's still alive and living. It does deliver something", interviewee 4;
- "Not prescriptive in the techniques to be adopted", interviewee 7;
- "Good where end-users are not heavily involved", interviewee 6. It should be noted that in the experience of the author 80% of the company's projects deliver systems where end-users are key to success, either in staff or customer facing channels. The main exceptions being projects carried out by the operational support department, who carry out projects to upgrade and introduce new software and hardware infrastructure upon which end-user facing systems run.

When asked to express their opinion regarding the weaknesses of the current in-house process four of the eight interviewees stated that the process duplicates effort or was not cost effective. Interviewees 2, 4, 7 and 8 highlighted the bureaucratic nature of the current in-house process as being a weakness. Interviewees 4, 7 and 8 mentioned that the process was in their opinion poorly understood. Again the one-size fits all nature of the current process was observed by interviewees 4, 7 and 8 to be a problem. Interviewee 7 stated that "... the current in-house skills are not strong enough to adopt an agile approach. You would also need a mentor within the teams and in the co-ordination team for AWE to be adopted successfully". This again strengthens the perception of poor skill levels in Web engineering within the company and resonates with similar research (Fichman & Kemerer, 1997) into the importance of mentors during the adoption of object oriented technologies.

Our last question (16) 'Describe the major phases involved in the organisation's development process' was intended to provide a final check on interviewees' awareness of the current in-house process. The answers to this question were considered too specific in terms of terminology to present in any detail in this dissertation without compromising the anonymity agreement between the author and the company. In general terms, the answers reflect some of the important issues that have been discussed previously in chapters 8 and 9:

1. The interviewee's answers varied considerably. For example, the number of phases identified by each interviewee ranged from 5-15. Interviewee's 1-8 identified 10, 5, 11, 8, 15, 9, 8, 6 phases respectively. Indeed interviewee 8 can be quoted as saying that "I don't know what the purposes of half these meetings/phases are!";
2. Interviewees 2 and 8 never mentioned the testing phase;
3. Interviewees 1, 2, 6, 7, 8 never mentioned a maintenance or evolution phase.

9.2.8 Post-AWE Pilot Survey Conclusions

The majority of those involved in Web-based development within the company acknowledged a difference between Web engineering and traditional software engineering projects. In particular, time-to-market pressures, the type of development approach required, and developer skill set were acknowledged as being different. The interviewees also highlighted the perceived lack of skills and inexperience in Web engineering within the company.

The current in-house company process is perceived by many to be too slow to address time-to-market pressures on Web engineering projects. The predictive nature of the process is not suited to the adaptive nature required in Web-based projects. In addition, the in-house process was observed to fail to address the correct stakeholders required to successfully deliver Web-based solutions.

The interviewees perceived AWE to have a number of strengths including: helping to address time-to-market pressures; an adaptive nature; and specific focus on the challenges facing Web Engineering projects. However, there were a number of characteristics of AWE dependent upon the culture of the company and staff adopting it, that were seen by some as strengths and others as weaknesses. These strengths and weaknesses included: empowerment of the development team, dependency on quality developers, and the need to change company culture before one could successfully adopt AWE. The major hurdles to the official adoption of AWE as part of the company's approved process for Web engineering projects seemed to be inertia, ignorance and poor suitability to the existing company culture.

The first commercial Pilot of AWE was regarded by six of the eight interviewees to be beneficial, the other two interviewees were not aware of the Pilot. The Pilot was seen to address areas of the company's process that do not focus on usability issues, which are perceived as being crucial to success. However, the company did not officially adopt AWE or those features of AWE explored within the pilot. Inertia, lack of senior management buy-in and poor ownership of the in-house process were seen as the major reasons for the lack of formal adoption.

When asked what changes the interviewees would like to see to the in-house process, half the interviewees mentioned the adoption of agile approaches. Three of the eight interviewees mentioned changes to the one-size fits all approach of the current process, with specific processes tailored to specific types of development activity. AWE was perceived by many in the interviewee sample to be better suited to Web engineering projects than the current in-house process. A number of features of AWE were acknowledged by the interviewees' to be better suited to Web engineering projects, including addressing time-to-market pressures and the incorporation of the Evaluation phase. However, according to the interviewees, the lack of formal adoption by the company seemed to derive from inertia, the need for a cultural change in order to adopt AWE, and a senior management desire for a one-size fits all development process.

In conclusion, the AWE pilot and the post-AWE pilot survey indicate that the AWE process is better suited than the company's in-house development process for building Web-based applications. Yet significant cultural and environmental hurdles exist which seem to prevent the company from adopting the AWE process. Others (Boehm & Turner 2003a) have commented on the cultural barriers that often hinder the adoption of agile processes. However, the empirical research presented in this chapter indicates that this company's desire for a one-size fits all development process is also a major stumbling block to the adoption of the AWE process. This is not because it is an agile process but because it is a process specific to one type of development activity only. Other organisations who have the desire for a one size fits all development process, spanning many different development activities, will struggle to harness the benefits from the Web engineering community and other communities focused on specific types of development activity. The following section uses home grounds analysis to indicate how the agile nature of AWE affects its adoption within the company.

9.3 Is the AWE Process Suitable for this Company?

Boehm and Turner (2003a & 2003b) explore the use of home grounds analysis to determine whether an agile or plan-driven process approach is most suited to a particular project or organisation. The home grounds for agile and plan-driven processes are 'the set of conditions under which they are most likely to succeed'. Boehm and Turner argue that the more a particular organisation or project's conditions differ from the home ground conditions, the more risk there is in using one approach in its pure form and that more value is gained in blending in some of the complementary practices from the opposite approach. This section explores the use of home grounds analysis to help illuminate the cultural and environmental factors that prevent the company adopting AWE for the development of their Web-based systems.

The home grounds for agile and plan-driven processes use five decision factors: size; criticality; personnel; dynamism and culture, to determine the relative suitability of agile or plan-driven process in a particular project. The home ground polar chart provides a mechanism for visibly evaluating a project along each of the five axes, thus allowing the determination of a project's location in this decision space. Figure 30 illustrates a blank home ground polar chart, reproduced from (Boehm & Turner 2003a).

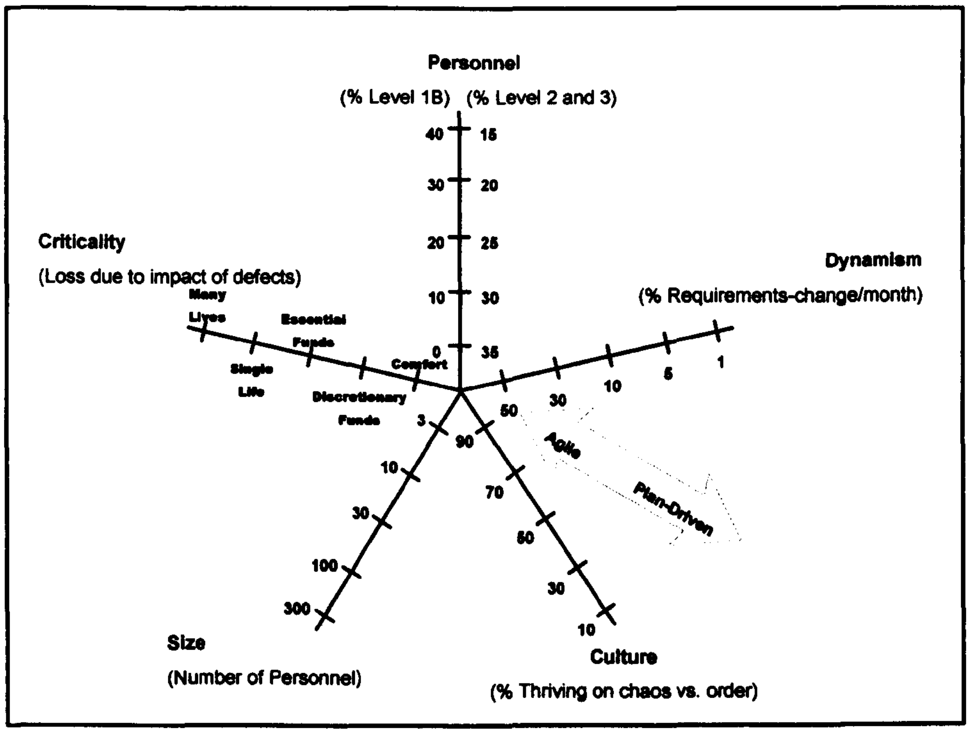


Figure 30. Home Ground Polar Chart. Reproduced from Boehm and Turner (2003a)

Once a project has been rated with respect to the five decision factors, and mapped to its home ground polar chart, it is possible to show the home grounds relationship graphically. Boehm and Turner describe three broad categories of project that can be observed with the home grounds approach. Each of these categories is labelled here as follows:

1. *Agile home grounds*. All the ratings are near the centre. Therefore, the project is most suited to an agile process;

2. *Plan-driven home grounds*. All the ratings are at the periphery. Therefore, the project is best suited to a plan-driven process;
3. *Exception territory*. A project rating is mostly in one or the other home grounds. Boehm and Turner (2003a) recommend treating the exceptions as sources of risk, and have devised risk management approaches to address them.

Boehm and Turner (2003a) describe a five-step, risk-based method to assist those who find themselves in projects categorised here as the 'exception territory'. The five-step risk-based method assists projects in category three, exception territory, to incorporate both agility and plan-driven process features in proportion to a particular project's needs.

Using the author's own experience of working within the company the author decided to determine the home grounds location of a typical project and a Web-based project. The rest of this section gives a brief overview of the five decision factors for determining the home grounds for agile or plan-driven processes. Followed with a home ground polar chart position for a typical project and a Web-based project within the financial service sector company. This is then followed by the conclusions.

9.3.1 The Five Decision Factors

The following list gives a short description of the five decision factors as used by Boehm and Turner. A rationale and more detailed explanation of each decision factor can be found in Boehm and Turner's recent book (2003a) and conference publication (2003b):

1. *Size* refers to the number of stakeholders involved during the development on a software project. Boehm and Turner argue that agile projects' home ground is more suited to smaller size teams, generally less than 10 people, with larger sized teams, generally greater than 40 people, being more suited to plan-driven processes;
2. *Criticality* is used to describe the loss or damage incurred from undetected defects. Criticality can be considered as impacting many different organisational or project characteristics, including: money, time-to-market, loss of life etc. Boehm and Turner present a scale for criticality starting with the lowest level of criticality rating, comfort. They then proceed with an increase in criticality through loss of discretionary funds, loss of essential funds, loss of a single life to the most severe critical rating loss of many lives. Boehm and Turner argue that agile projects' home ground is more suited to projects where criticality is considered low, with plan-driven processes being more suited to projects where criticality is considered high;
3. *Personnel* describes the quality of the team members using the respective process approach. Cockburn's method skill rating system (Cockburn 2002) is the foundational theory used to underpin this home ground factor. Table 22 shows the five levels of software method understanding and use (after Cockburn) employed by Boehm and Turner. Boehm and Turner argue that plan-driven projects' home ground is suited to a mix of high- and low-level skills, with agile projects requiring a greater percentage of higher-level skills;
4. *Dynamism* describes the rates of change experienced by projects. Boehm and Turner argue that plan-driven projects' home ground is suited to low rates of change, with agile projects being suited to both high and low rates of change;
5. The *Culture* axis reflects the flexible or chaotic nature within the organisation(s) in which a project is developed. Boehm and Turner argue that agile processes are more suited to environments where chaos thrives, with plan-driven processes being more suited to environments where order thrives.

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation.
2	Able to tailor a method to fit a precedented situation.
1A	With training, able to perform discretionary method steps (e.g. sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With training, can become level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Table 22. Levels of Software Method Understanding and Use (after Cockburn). Reproduced from Boehm and Turner (2003a)

9.3.2 Agile or Plan-Driven: The Best Suited Company Process Approach for Web engineering?

In this subsection a brief discussion is presented describing a typical company project and a Web-based company project with respect to each of the five decision factors used in the home grounds analysis. The reason for choosing a typical project derived from the financial service sector company's goal to have a one size fits all process approach to cover all information technology projects including Web-based projects. The reason for evaluating a Web-based project is that the AWE Process was only created for Web engineering and therefore can only be applied using home grounds analysis to determine the suitability for Web engineering projects within the financial services company. The author (McDonald & Welland 2001c) and others (Cockburn 2002 and Glass 2002) do not agree with the view that a one size fits all approach is practical or optimal in large enterprise technology environments where many different types of development activity occur. The following points present the home grounds analysis and justification for a typical project in the company:

1. *Size (Average of 25 people per project).* The author's experience would suggest that there are on average 25 people per project. It should be noted that the existing process approach is plan-driven and like many plan-driven approaches not all people are involved in the project throughout the life-cycle. Indeed few, with the exception of the Project Manager and Business Sponsor, will see the project from inception to decommissioning. It is also worth noting that the range of people on any of the projects currently in development would be from 10-250+;

2. *Criticality (Between Essential and Discretionary Funds, closer to Essential Funds).* It is rare indeed for established financial services companies, especially banks, to return anything but a profit to the shareholder. The size of the return and the direction of the profit margin from year to year is normally the major point of discussion. Loss of life is exceptionally rare as a direct impact of project failure. In addition, the conservative nature of the financial service sector and banking industry means that little funding is released without an approved business case from strategic funds;

3. *Personnel (40% Level 1B) (15% Level 2 & 3).* This level is based upon the author's experience working on a number of projects within the company;

4. *Dynamism (5% requirements change per month).* The majority of requirements analysis on typical projects within the company is carried out upon systems that are well understood and can follow existing patterns from previous projects;

5. *Culture (25% thriving on chaos vs. order).* The bureaucratic nature of the organisation has resulted in tomes of written documentation and procedures for even the most basic tasks. In many cases it is often simply not humanly feasible to follow the official process, never mind attempt to understand all the procedures one must follow. For example, the author came across the template for a mandated document, used during the official in-house company software development process that required fifty plus stakeholders' approval and sign-off before the project could proceed. However, the author could find no project that had actually gained the sign-off of all fifty plus stakeholders.

The following 5 points present the home grounds analysis and justification for a Web-based project in the company:

1. *Size (Average of 17 people per project).* Most Web engineering projects are relatively new to the company when compared to other types of projects. As a result less people are normally involved or sufficiently skilled to operate on such projects. Most stakeholders involved within Web-based projects spend a greater percentage of time on these new endeavours;

2. *Criticality (Discretionary Funds).* The majority of Web engineering projects result in new channels to access business logic that resides within mainframe based environments. Essentially Web engineering projects provide presentation and integration logic tiers to business logic tiers. There is generally less money invested in Web engineering projects when compared to projects involved with the business logic tiers of the company's architecture. This is primarily due to the criticality of such projects. For example, the availability and downtime associated with an Internet Banking channel is less critical than the availability and downtime associated with business logic tiers. This is primarily due to the fact that the availability and downtime associated with the business logic channel will prevent all channels from operating, including telephonic, branch, mobile as well as Internet. As a result the monies invested in Web engineering projects are less critical and are discretionary. Although, as the adoption of Web-based and Internet technologies grows throughout the enterprise, the funding for Web-based projects is increasing and thus these funds are becoming more essential;

3. *Personnel (10% Level 1B) (30% Level 2 & 3).* The inexperience in Web-based endeavours within the company and the growing importance and adoption of Web-based and Internet-based technologies results in higher skilled stakeholders being involved in the development of Web engineering projects;

4. *Dynamism (20% requirements change per month).* The relative inexperience of the company in Web engineering, the faster time-to-market pressures and more volatile nature of e-markets all contribute to a greater change in requirements on Web-based projects. The experience of the author suggests that the rates of change on Web-based projects that occur within the non-functional requirements are generally greater than those experienced within the functional requirements. However, the rate of change of functional and non-functional requirements is significantly greater for Web-based projects than is found in traditional projects;

5. *Culture (35% thriving on chaos vs. order).* The adaptive nature of Web engineering attracts stakeholders who are more comfortable with a greater level of chaos in their working environment. Often these stakeholders have a background outside the traditional financial services sector, such as graphic design. However, they are in the minority overall, although in a greater percentage than is found in typical projects.

See Figure 31 for the home ground polar chart rating for a typical company project and a Web-based company project.

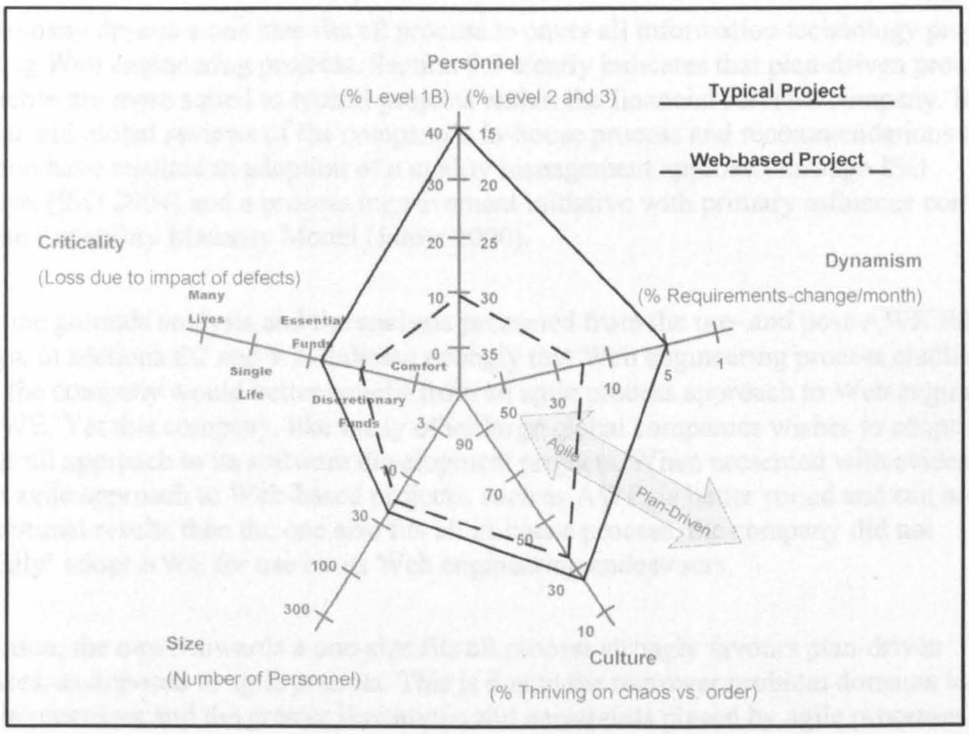


Figure 31. Home Ground Polar Chart for a Typical Project and a Web-based Project

Boehm and Turner argue that the home grounds tell whether or not a project is suited to an agile or plan-driven approach. Of the five decision factors used to determine the home grounds: size, personnel, culture, dynamism and criticality, only the latter two, dynamism and criticality, reflect features of the project problem to be solved by the project using the process. The first three decision factors: size, personnel and culture, describe features of the organisation(s) that will have to solve the problem(s) using a plan-driven or agile process, or a hybrid process of agile and plan-driven approaches.

While organisational characteristics are important, the ultimate goal of a process is to help a team and organisation solve a problem or series of problems. Therefore, advocates of agile methods believe that process is secondary to the people involved in the team. Indeed, this is an important point, if you don't have the right people your chances of solving your problems are severely reduced regardless of what process approach you adopt. However, there is a general consensus of opinion (Boehm & Turner 2003a) that with a plan-driven process approach you are more likely, than with an agile process approach, to get something that works. This is believed to be especially true with poorer skilled developers, even if this deliverable is not an optimal solution, or in the worst case, does not solve your project's problems at all!

Given the company's goal of a single one size fits all process approach, this section has mapped the company's home ground polar chart and the rationale for the company's position for typical projects with respect to the five decision factors, see figure 31. Boehm and Turners' home grounds analysis gives a strong indication that plan-driven processes as opposed to an agile processes are most suited to typical projects within the financial services company. However, home grounds analysis indicates that Web-based projects within the financial services company are suited to a more agile process than typical projects.

9.3.3 Home Grounds Analysis Conclusions

The company desires a one size fits all process to cover all information technology projects including Web engineering projects. Section 9.3 clearly indicates that plan-driven process approaches are more suited to typical projects within the financial services company. Recent regional and global reviews of the company's in-house process and recommendations for its evolution have resulted in adoption of a quality management approach through ISO standards (ISO 2004) and a process improvement initiative with primary influence coming from the Capability Maturity Model (Jalote 2000).

The home grounds analysis and the analysis presented from the pre- and post-AWE Pilot Surveys, in sections 8.2 and 9.2, indicate strongly that Web engineering process challenges facing the company would better benefit from an agile process approach to Web engineering like AWE. Yet this company, like many other large global companies wishes to adopt a one-size fits all approach to its software development projects. When presented with evidence that an agile approach to Web-based projects, such as AWE, is better suited and can achieve more optimal results than the one size fits all in-house process, the company did not 'officially' adopt AWE for use on its Web engineering endeavours.

In addition, the move towards a one-size fits all process strongly favours plan-driven processes, as opposed to agile process. This is due to the narrower problem domains tackled by agile processes and the greater limitations and constraints placed by agile processes on their adopters. Ultimately though, companies such as the one involved in this research collaboration would seem to benefit their Web engineering endeavours by using an agile oriented process as opposed to a one size fits all process covering all technology projects.

9.4 Summary

The post-AWE pilot survey supports and strengthens the claims made in chapter 8 that indicate that the financial services company is facing many of the process challenges in Web engineering described in the first survey of Web engineering in practice. In particular, time-to-market pressures are not being effectively addressed by the current in-house process approach. The AWE pilot and the post-AWE pilot survey suggests that an agile approach, such as AWE is better suited to addressing these process challenges than the current in-house company process approach. Yet when presented with evidence to support this argument AWE or even elements of it were not officially adopted by the company for Web-based development projects.

The post-AWE pilot survey indicates three major hurdles that arose when trying to get a process specifically for Web engineering, based on AWE, officially adopted within the company for Web-based projects:

1. *Inertia.* The author experienced great resistance to doing anything differently in Web engineering projects, even though many company employees

acknowledged the logic of adopting an agile approach for Web-based projects. They also acknowledged the benefits realised by using AWE;

2. *Culture*. The company's culture was perceived to be a poor fit with AWE and agile process approaches in general;
3. *One Size Fits All Process vs. Specific Processes*. There exists a conflict between the company aspiration for a one-size fits all process, as opposed to processes specific to different types of development activity such as Web-based projects. This conflict ultimately presents a significant research challenge to the Web engineering community who (Murugesan et al., 2001) recommend different software engineering approaches including process approaches for Web engineering.

Much of the agile literature observes the dependency on culture required to successfully adopt agile processes, such as Beck (2000) and Boehm and Turner (2003a & 2003b). However, a challenge of equal proportions exists in gaining acceptance within many large organisations to address different categories of development activity using different process approaches, whether agile or plan-driven. The desire for a one size fits all approach to developing software systems is understandable. However, as the very existence of the Web engineering community testifies, there are advantages to be gained by focusing on specific types of development activities, such as Web engineering. Nonetheless, it is naive to assume that Web engineering is just a pure subset of software engineering. If large companies are to successfully improve the development of their Web-based systems, then they must accept that their Web engineering approach may have to be different from other development activities.

Finally, the results from sections 9.2 are validated using Boehm and Turners' home grounds analysis. Home grounds analysis strongly indicates that the company's Web-based projects are better suited to an agile process like AWE, than to a plan-driven process, like the company's in-house process that is a one-size fits all process for all information technology projects.

The next chapter describes the influence AWE had upon an Intranet development stream, one of thirteen projects, within a major Intranet development programme. The chapter then discusses the results of a survey of the Intranet team and their impressions of the influence the AWE process had upon the project.

10 AWE's Second Commercial Usage

This chapter describes the influence AWE had upon an Intranet development stream, one of thirteen projects, within a major Intranet development programme. The Intranet development stream was challenged with integrating an enterprise wide Identity Management (IM) solution for the Intranet programme and all future Web-based projects. The following definitions, reproduced from in-house company documentation, serve to help the reader understand the high-level objectives of the IM programme:

Identity defines an individual through data that is used to link authentication and authorisation to enable access to back end data which constitutes a user's total relationship (e.g. Identification Number, Single Sign-on). The *Identity Management (IM)* programme comprised four major streams described as follows:

1. *Directory* holds the access and control data for enterprise applications, e.g. user name, password, address, secure certificate, business roles, email address;

2. *Permissions Management Infrastructure (PMI)* is a rules engine, which uses the data in the directory to enable other enterprise applications to externalise their access and control functionality. This permits role based access and control rules to be changed once without having to rewrite/configure each enterprise system individually;

3. *User Provisioning (UP)* provides a federated mechanism to allow the delegation of access and control administration, enabling the reduction of the number of user names and passwords that need to be remembered by each end-user and the number of administrators required across the company;

4. *Public Key Infrastructure (PKI)* enables secure authentication of users and systems and permits the organisation to place a very high degree of confidence in identity of individuals and where necessary external customers.

The Identity Management Programme is responsible for the development and implementation of effective systems, process, and procedures for the company across the following areas:

- *Provisioning*. The process of associating an identity with one or more "accounts" on IT systems that provide services to authorised users;

- *Authentication*. The means by which an entity's identity is verified;

- *Access Management*. The mechanism and process used to enforce resource access policies based on identity and entitlements.

The business scope is to deliver a base infrastructure across the company that includes *Identity*, *Access Management* and *Authentication* components. These three components are together referred to as the 'IM Core' and will be the foundation of the common shared infrastructure.

The first section describes the IM project in more detail and highlights important aspects of the project that were influenced by AWE. The second section discusses the results of a survey of the IM Intranet team and their impressions of the influence the AWE process had upon the success of the project.

10.1 Using AWE on an Identity Management Intranet Stream

At the end of the one year Ph.D. internship the author was approached by a senior development manager to help build an Identity Management team. The author and was retained as an external consultant two days per week from October 2002 until December 2003. The senior development manager was one of four within the European Development Centre and was responsible for over ninety technology development staff across a number of areas including bank payments and ATM machines. The senior development manager had read the AWE process technical report (McDonald & Welland 2001c) written during the author's Ph.D. internship year (October 2001-September 2002).

10.1.1 The Task

The Identity Management (IM) project was a part of a much larger Intranet project deploying across three European countries in three different banks. The primary objective of the Identity Management project was to reduce the number of user ids and passwords being given to bank-based users of the Intranet system. The organisation had decided to buy and customise a number of Web-based 'component off the shelf' solutions, within the Intranet project, and an initial estimate indicated that the roll out of the Intranet project would require the addition of nineteen different user ids and passwords for end-users.

During the last months of his Ph.D. internship the author had proposed the use of the enterprise identity management infrastructure to reduce the number of user ids and passwords across the Intranet project. This included providing Web-based single sign-on for:

- a Customer Relationship Management System used to sell mortgages and loans implemented in Siebel 7.5.x;
- an account maintenance and enquiries system implemented using the J2EE platform in WebSphere 5.x;
- and the integration of a Web-based email and calendar system in Notes Domino 6.5.x.

In addition, the organisation also desired the same authentication credentials to be used across a number of systems including Active Directory 2.0 for log-on to Windows XP and a number of backend legacy systems written in COBOL running in a CICS environment.

The Intranet project had been running for one year using the in-house process and had yet to define its requirements and scope to the satisfaction of those controlling the funding of this major development programme. Identity Management was purely an architectural vision within the group globally, and at this point had not been used on any projects within the group.

The author's task was to help build an Identity Management team from scratch and to realise the Identity Management architectural vision to underpin the Intranet project within Europe. The objective was to provide Web-based single sign-on (authenticate once to all Web-based systems) and same sign-on (use the same authentication credentials) across the Web and other enterprise systems such as PC log-on and legacy system access. The Intranet project had, at its peak, over 300 stakeholders working in parallel on the programme and a budget of over 50 million pounds spread over 24 months. The IM project was allocated 3 million pounds sterling and had to ensure that the systems were delivered for phase one of the Intranet project within seven months.

10.1.2 Collocating the Development Team

The team comprised two business analysts, four software engineers and three team leaders in the European IM project. In addition, two product experts, who were the only stakeholders who had experience of developing an Identity Management solution, were employed full time by the project. The two business analysts and one team leader were based in a different country to the four software engineers and the other two team leaders. Both the product experts were based in the same country as the software engineers.

Given the extreme time pressures and the scale of the project the senior development manager gave his support to collocating the team as recommended in the AWE process. It was therefore decided to collocate the team into one development area within the building used by the software engineers. This resulted in the entire team working together for two or three days per week during phase one of the Intranet project. The team used a number of mechanisms to communicate with each other including face-to-face communication around white boards and by focusing the realisation of the solution through the browser experience. Due to the rapid pace of development and the number of teams the IM project was interacting with (thirteen in total) the team decided to have half hour meetings every day at 9am. At these meetings everyone discussed their work for the day and each of the developers could raise issues and agree engagement with other team members outside of this daily meeting slot that was strictly time-boxed. In addition, the business analysts and software stakeholders collaborated to produce a number of documents and presentations to be used as a mechanism to communicate with other areas and projects within the Intranet project and to the wider organisation.

A great deal of resistance was encountered to collocating the software engineers who were involved in the IM project by their old line managers and it took three months to gain approval to create a space for the new team within one of the development floors. As a result the team collocated in the basement which was also used as the European ATM development area and contained fourteen ATM machines!

In spite of the less than ideal set up the team managed to run the fastest selection of a global enterprise product within recent times to select an enterprise user provisioning system which was the last major component in the Identity Management global product suite.

10.1.3 Business Process Re-engineering

The AWE Process emphasises the importance of re-engineering business processes when introducing new communication mechanisms through Web-based technologies. This will allow organisations to maximise the benefit from their Web-based investment. Despite the emphasis placed on re-engineering processes for user provisioning, the IM project experienced a great deal of inertia within the organisation and strong resistance from the Intranet project to changing the existing user provisioning processes. Ultimately this failure was not due to the IM team's understanding of the importance of business process re-engineering or any emphasis placed on this activity by AWE, but primarily due to organisational inertia. For phase one it prevented the adoption of the automated enterprise user provisioning product selected by the European IM team during the first three months of the project and resulted in the delayed deployment of this automated solution until phase four. The expense in time and money of the manual provisioning process was seen to hinder the growth of the Intranet project beyond phase two and it was only at this time the changes to the existing business process were finally accepted.

10.1.4Employing the AWE Evaluation Phase

Despite the successful pilot of AWE's evaluation phase within the company's European Retail Internet Banking Channel, described in section 9.1, the Intranet project offered great resistance to adopting this beneficial element of the AWE process. A proposal to bring in a usability expert to carry out Think Alouds during the first few weeks after the launch of phase one of the Intranet project was rejected by the Intranet programme director. The official reason given was due to the potential impact the results might have in terms of time scales and costs. Unofficially the author was informed that the senior programme leaders were concerned about the potential findings of the AWE evaluation phase and were fearful that adverse findings may prevent the Intranet project from gaining funding to continue.

10.1.5IM Project Achievements

The IM project managed to achieve Web-based single sign-on and same sign-on user-id credentials across seventeen enterprise systems, although this was delivered for phases one, two and three using a manual paper based provisioning process. Phase four of the Intranet project will deliver the automated user provisioning system (late 2004) and associated new business processes that will reduce the number of fixed term employees in the manual paper based process by over twenty. It will significantly reduce the cost of supporting the Intranet system and allow the end-user's password to be the same across all systems, like the user-id.

The AWE process was not employed in its entirety. This was primarily attributed by the 'Technology Sponsor' (interviewee 7) to cultural factors as reflected by the following answer to question 8 "... There are many cultural barriers that would need to be overcome in order to fully adopt AWE in this organisation". Although the AWE process was not employed in its entirety, a number of successful elements influenced the IM project including:

- self organising team;
- collocation of the business and technology developers;
- and communication through the browser experience.

The success achieved by the IM project was reflected within the organisation during the early part of 2004, when the European IM project won the global technology prize for value add within the company.

10.2 Impressions upon an Intranet Development Team

The survey was carried out in May 2004. Eight different stakeholders, referred to respectively as interviewees 1-8, were involved. The interviewees comprised the business and technology development stakeholders who were responsible for delivering the first project to use the Identity Management infrastructure in Europe.

10.2.1 Survey Methodology

The survey was conducted in a qualitative manner using an in-depth one-to-one interview technique. Each interview took approximately three quarters of an hour. Each interviewee was asked 19 questions, all of the answers were recorded on paper by the interviewer and were then tabulated, see appendix 8. In order to help ensure anonymity of the company and the employees the results of the interviews are presented anonymously. The results of this survey have not been presented within the company itself, this section serves as the first airing of this survey in published form.

10.2.2 Team Demographics

The sample interviewees included two development team leaders (one from the business team and one from the technology team), one technology sponsor, one engineering room manager who resourced the project with internal and external technical staff, one business expert and three software engineers. The interviewees average four and a half years experience of developing Web-based applications with only interviewee 5 having no previous experience of Web-based development before the Identity Management project.

There were a number of interesting observations from the interviewees' answers to question 4 'In your experience do you notice any major differences between Web-based development and Tradition IT Projects?'. These included:

1. The increased importance of the end-user role and usability in Web engineering (interviewees 1, 3 and 6);
2. The need for close collaboration and interaction between business and technology stakeholders (interviewees 1, 3, 4);
3. Poor suitability of the current in-house process to Web engineering and the difficulty in trying to apply its predictive approach to Web-based development. Also identified was the need for a more adaptive, iterative and incremental process (interviewees 4, 5, 6, 7, 8);
4. The difference in the skills required to develop Web-based applications when compared to traditional IT projects (interviewee 6 – the engineering room manager who resourced the project with internal and external staff);
5. The fact that the company did not distinguish between different types of project such as Web-based application development (interviewees 3, 7, 8).

The interviewees' answers to question 4 reinforce the conclusions to chapter 8 that the in-house company process is poorly suited to Web engineering. In addition, the last observation reinforces the point that the company desires and employs a one-size fits all process. Observation one reinforces criterion 5 and observations two and four reinforce criterion 3 for a Web engineering process, see chapters 3 and 4 for more detail.

10.2.3 Perceptions of the current in-house process

The interviewees' answers to question 5 'What do you think of the organisation's development process and its applicability to Web application development?' illustrate the suitability of the company's in-house process to Web engineering:

1. The in-house process is not suitable for Web engineering (interviewees 1-8);

2. The in-house process should support closer cooperation between business and technology to develop Web-based solutions (interviewees 1 & 2);
3. The in-house process has poor focus on end-users (interviewee 2);
4. The development process life-cycle needs to be more adaptable, and needs to support an iterative and incremental development approach (interviewees 2, 4, 5, 6, 7 & 8);
5. Cultural and organisational barriers present the biggest challenge in developing Web-based applications differently (interviewees 1 & 3);
6. The one-size fits all nature of the in-house process is a problem for Web engineering within the company (interviewee 2).

10.2.4 Perceptions of the AWE process

Answers to question 6 showed that three of the eight interviewees had read the AWE Process technical report (McDonald & Welland 2001c). The three interviewees' answers to question 7 'What do you perceive as the strengths of the AWE Process?' are listed below:

- Interviewee 1: "Iterative nature helps maximise technology and business knowledge to the benefit of the end-user";
- Interviewee 2: "Collocation. AWE allows the development team to do it. Gives the development team a mandate to go and make it happen";
- Interviewee 7: "It is a methodology that any medium financial service sector organisation should feel happy embracing. The mistake made in this organisation is that we treat it as a technology methodology rather than a business methodology, as it tries to improve productivity and provides the 'power of delivery'".

The answers from the three interviewees who answered question 8 'What do you perceive as the weaknesses of the AWE Process?' are listed below:

- Interviewee 1: "In this organisation there are serious organisational and cultural barriers. The finance model dictates the development approach and causes a culture of protectionism. The current approach hinders technology and business working together especially at the initiation stage";
- Interviewee 3: "Not fit for all. Some people need to be hand held or are to silo-focused in their work. AWE needs people who are willing to collaborate";
- Interviewee 7: "Only weakness is that we have not used it fully. No weaknesses, as I have not seen it used fully in anger. There are many cultural barriers that would need to be overcome in order to fully adopt AWE in this organisation".

Of the three interviewees who had read the AWE process technical report two of them mentioned strengths surrounding the focus on business and technology working together, and two interviewees mentioned AWE's enablement of the development team to deliver a solution as a strength. With respect to the weaknesses of AWE, two interviewees mentioned the cultural and organisational barriers within the company as being the major hurdles to AWE's successful adoption. Interviewee 2 indicated that AWE was not suitable for all individuals.

10.2.5 Collocating the Business and Technology Team.

All the interviewees' answers to question 9, 'What were your opinions regarding collocating the business and technology development team?', illustrated the positive and beneficial nature of collocating the business and technology developers during the Identity Management project. Interestingly the answers to question 10 'What problems did you experience/perceive with collocating the development team?' showed that the travel involved in collocating the business and technology teams (interviewees 1, 2, 3 & 8) proved by far the biggest issue, as both the business and technology teams were geographically based in different countries. Other observed problems included: the difficulty in scaling collocation to large teams and an individual's need and desire to work individually at certain times. Both these issues are known challenges in applying agile software processes and are discussed by Beck (2000). Interviewees also mentioned the need for strong leadership as previously discussed in chapter 6, and the need for CRACK (Collaborative, Representative, Authorised, Committed and Knowledgeable) developers, these are also challenges observed and discussed in the agile literature (Boehm & Turner 2003a).

The interviewees' answers to question 11 'What benefits did you experience/perceive with collocating the development team?' reinforced the positive responses received to question 9. Interestingly the technology sponsor's response (interviewee 7) to question 11 "IM did not really accrue the benefits, hindered by multi-location, multicultural project. The utopia would be to give one team direction and guide them to doing it." was different from the other interviewees who were regionally focused and spent a great deal of time collocated. The technology sponsor was aligned at a global level and spent a great deal of focus on the global IM team that was split over two continents and communicated by teleconference and email. The answers to question 11 highlighted the following benefits from collocating the development team:

- Improved team communication (interviewees 1, 2, 3, 4, 6 & 8);
- Increased speed and efficiency during development (interviewees 1, 4, 5, 6 & 8);
- Increased team spirit and morale (interviewees 1, 2 & 6).

10.2.6 Business Process Re-engineering

The interviewees' answers to question 12 'How appropriate was the emphasis on business process re-engineering from the beginning of the IM stream?' indicated that while the IM stream was very aware of the importance and benefit derived from re-engineering the business processes that the Intranet project as a whole placed poor focus on this aspect of the solution. Indeed a number of the interviewees acknowledged that although the IM stream was aware of the importance of business process re-engineering, and focused on this aspect of the solution before any of the other streams, that it was extremely difficult to carry out effectively or to maximum benefit without other streams participation and co-operation. The importance of business process re-engineering being assisted by a Web engineering process is reinforced further by the following quotes from the interviewees in answer to question 13 'Did the development process hinder or assist with business process re-engineering activities?':

- "The Intranet project business process hindered it..." (interviewee 1);
- "I felt that the Intranet process had no explicit focus on business process re-engineering. Too technology focused" (interviewee 2);
- "Hindered. The development process is project oriented, concerned only with meeting project deliverables. Lack an enterprise model" (interviewee 4);
- "It hindered it as it was too structured. It needed a more iterative process approach..." (interviewee 6);
- "The in-house process has not been suitably reviewed to introduce new technologies." (interviewee 7);

- "I don't think that there is a real business development process for projects. Clearly a gap!" (interviewee 8).

Interestingly the answers given by interviewees 3 and 5 to question 13 indicate that they did not feel that the development process affected business process re-engineering. This could be due to the fact that it did not acknowledge the need for business process re-engineering during Web-application development.

10.2.7The Identity Management Programme

The interviewees' answers to question 14 'What did you consider successful regarding the Identity Management project?' indicate that the major success was that the IM stream delivered a working solution on time that added value to the Intranet project. This is illustrated by the following quotes from the interviewees' answers to question 14:

- "New technology was implemented and works, because the business and technology worked together as one team. IM was delivery focused" (interviewee 1);
- "It has gone in and is being used successfully in the project time-scales..." (interviewee 2);
- "It worked. Brought together a new team that gelled and grew together. Need to do this more often in the group" (interviewee 3);
- "The fact that we managed to deliver it given its scope" (interviewee 4);
- "It brought in new technology very swiftly..." (interviewee 5);
- "It was delivered on time; it was sold and accepted by other Intranet technical streams, which was the hardest to achieve; it brought about a cultural change in technology, and it was fun" (interviewee 6);
- "Implemented on time and within budget; met business requirements, ..." (interviewee 8).

The interviewees' answers to question 15 'What did you consider unsuccessful regarding the Identity Management project?' resulted in the following observations:

- Conflict between being comprehensive in agile development when time-to-market pressure are primary (interviewees 1 and 8);
- Problems with cost and effort estimation in an adaptive development environment where there are many unknowns (interviewees 2 and 3);
- The need for strong cross-stream communication in large Web engineering projects, achieved in AWE through orthogonal uni-discipline communication and the co-ordination team (interviewees 3, 5, 6, 7, 8).

10.2.8AWE's Influence on the Identity Management Stream

The interviewees' answers to question 16 'What influence did you consider the AWE Process having had on Identity Management being successful or unsuccessful?' showed that the major benefit was collocation of multidisciplinary developers into one team at the beginning of the project. Interestingly, both business developers (interviewees 1 and 2) observed the collocation of the business and technology team to fall away as the in-house process was adopted. The interviewees' observed other beneficial influences of AWE on the Identity Management project including: improved focus on requirements; improved communication; and increased focus on the project objectives/tasks. The technology sponsor's response (interviewee 7) to question 16, "The fact that the author created the identity management culture. While we did not employ AWE verbatim we adopted many of its principles in Identity Management. We employed the author not the book" illustrates the strong influence the AWE process had on the Identity Management project but also indicates the strong influence of the author on the project as well.

The major factor observed by the interviewees in contributing to the success of the Identity Management project, question 17, was primarily the skill and strength of the development team.

Questions 18 and 19 pertained to the Intranet project's unwillingness to adopt the Evaluation phase into the Intranet project's development process. The interviewees' answers to question 19 (note only four interviewees answered the question) indicated that carrying out such a usability evaluation was perceived to have a negative effect on time-scales and costs.

10.3 Summary

While the AWE process was not employed fully 'in anger', the principles of AWE have had a major influence on a large Intranet project with exceptionally tight timescales and major technical and business challenges. The IM project demonstrated the value of: collocating the business and technical developers; empowering the team to self organise; and communication through the browser experience.

Despite these achievements and the success in winning the global company technology prize, there remain significant obstacles to the successful adoption of AWE within the company, as illustrated by interviewees' answers in section 10.2. The major hurdles identified were: organisational and cultural barriers; poor alignment with the funding model; and the desire for a one size fits all development process.

All of the interviewees identified benefits in collocating the business and technology developers. Clearly the benefits of business process re-engineering in Web application development are illustrated in the interviewees' answers in section 10.2. However, even the influence of a process such as AWE that encourages business process re-engineering in Web application development was not nearly enough to overcome the organisational and cultural barriers within the company.

The following chapter contains the conclusions and suggested further work within the field.

11 Conclusions

My hypothesis is that developing Web-based applications (Web Engineering) has specific characteristics that differ from those normally assumed for software development processes. Therefore, a different type of development process is required for Web engineering. The first four sections of this chapter summarises the answers to the four research questions concerning the above hypothesis. The fifth and final section describes the areas for further work.

11.1 Research Question 1

The answer to the first research question 'Is it possible to define a set of criteria that a Web engineering process must fulfil?' is yes. Chapters 3 and 4 respectively established the empirical evidence and discussed in detail the criteria for a Web engineering process. The empirical evidence is based on the author's experience of building two large Web applications and the first survey of Web engineering in practice which enabled the development of an initial set of criteria. Subsequent reviews of published literature of Web engineering in practice identified further empirical evidence in support of these criteria. Based on the empirical evidence presented in chapter 3, chapter 4 discussed in detail the seven criteria for a Web engineering process:

1. Short development life-cycle times
2. Different business models
3. Multidisciplinary development teams
4. Small development teams working in parallel on similar tasks
5. Analysis and Evaluation
6. Requirements and Testing
7. Maintenance

These criteria are used later on in the dissertation to assess the applicability of development processes to Web engineering. In order to structure the descriptions of the processes evaluated, a working definition of process and a working description of the elements that comprise a process were presented.

11.2 Research Question 2

The answer to the second research question 'Can a new development process be defined to meet the criteria for Web engineering process?' is yes. The Agile Web Engineering (AWE) Process is described in Chapter 6. The AWE Process was developed to tackle the particular characteristics associated with Web engineering. AWE is an iterative and incremental process that is designed to address time-to-market pressures inherent in Web engineering. The AWE process life-cycle is designed for Web-based application development, evolution and maintenance. AWE emphasises the importance of business process re-engineering in maximising the return on investment and ultimately the success of Web-based endeavours. The development of Web applications requires multidisciplinary development teams as explicitly supported by AWE. AWE defines an orthogonal uni-discipline developer communication approach in order to assist with scaling this agile process to many teams of developers. AWE advocates the explicit identification of requirements, based on clear business needs and rigorous testing against these requirements. Developing Web-based systems without close involvement with end-users is flawed. AWE focuses the development team on the end-user experience of the proposed system and encourages evaluation of the

Web-based deliverables with a representative sample of end-users. Ultimately in AWE, end-users are the litmus test for project success.

11.3 Research Question 3

The answer to the third research question 'Can it be shown that such a new process is better suited to Web-based application development than existing plan driven, agile and alternative Web engineering processes?' is yes. Chapter 5 evaluated the support provided within three traditional software engineering processes against the criteria for a Web engineering process. The Unified Software Development (USD) Process, Dynamic Systems Development Method (DSDM) and eXtreme Programming (XP) are used as illustrative baseline examples to evaluate the suitability of traditional software engineering processes for Web engineering across the plan-driven, RAD and agile software engineering process spectrum. Analysis of these three processes showed that no existing traditional software processes either agile or plan-driven effectively addressed the criteria. Although as one moves through the plan-driven, RAD towards the Agile process end of the spectrum time-to-market pressures are better addressed.

I then evaluated other commercial processes that are proposed for Web application development against the criteria for a Web engineering process. Chapter 7 contains the evaluation of four Web engineering processes against the criteria using the available literature. The four Web engineering processes evaluated were Collaborative Web Development, Crystal Orange Web, the extensions proposed by Rational and Context Integration to the Rational Unified Process, and the proposed extensions to OPEN (Web OPEN).

With the exception of Crystal Orange Web, the commercial Web-engineering processes evaluated showed poor support for assisting with the time-to-market pressures experienced in Web-based projects. There is clearly a need for stronger support for different business models and business process re-engineering in the commercial Web engineering processes evaluated. Crystal Orange Web is the only process to incorporate the wide range of development roles required in Web engineering including business and domain experts. None of the processes provide a mechanism to support scalability to a number of small teams working in parallel. The Web engineering processes evaluated will all benefit from greater focus on end-user participation throughout development and evaluation phases. With respect to requirements, testing and maintenance, the extensions to traditional software engineering processes provide stronger support because of their foundations.

11.4 Research Question 4

The answer to the fourth research question 'Is it possible to demonstrate that this new Web engineering process can be used successfully in practice?' is yes. However, a number of obstacles have been identified that need to be overcome in order for this to happen. Chapters 8, 9 and 10 described the trials of AWE within a Fortune 500 financial service sector company. Initially, the in-house company process was evaluated against the criteria for Web engineering and it proved to provide weaker support than any other process evaluated in this dissertation. In fact it only provided weak support for business analysis, strong support for requirements and testing which it inherited from the plan-driven nature of the Waterfall model. I carried out a pre-AWE pilot survey that confirmed that the in-house process was heavily bureaucratic and provided extremely limited support for addressing time-to-market pressures. In addition the process was poorly understood by critical stakeholders and had little evolution during the 1990s. The evidence presented in chapter 8 indicates strongly that the company is facing the Web-based process challenges discussed in chapters 3 and 4.

I carried out the first commercial trial of the AWE process during the evolution and maintenance life-cycle of the company's Retail Internet Banking Channel in Europe. The first pilot focused on the inclusion of AWE's evaluation phase with a representative sample of end-users. The pilot established that the Agile Web Engineering (AWE) Process focus on end-user involvement through its Evaluation phase enabled an increase in end-user task completion rate from 47% to 79% on a commercial project. Despite the success of this first commercial trial of AWE, the company did not incorporate any features of AWE into its in-house development process. To further investigate this problem I carried out a post-AWE pilot survey of company stakeholders who were involved in the promotion of AWE and its first commercial trial. The post-AWE pilot survey illustrated that the interviewees were enthusiastic and encouraging of AWE's approach for building Web-based systems. The AWE pilot and the post-AWE pilot survey suggested that an agile approach, such as AWE is better suited to addressing the Web engineering process challenges than the current plan-driven in-house company process approach. This was validated using Boehm and Turner's home grounds analysis, which strongly indicates that the company's Web-based projects are better suited to an agile process like AWE, than a plan-driven process, like the company's in-house process that is a one-size fits all process for all information technology projects.

The second commercial trial of AWE involved using some of the AWE principles to influence the development of an Identity Management (IM) project, which underpinned a large Intranet programme with exceptionally tight timescales and major technical and business challenges. The IM project demonstrated the value of: collocating the business and technical developers; empowering the team to self organise; and communication through the browser experience.

Despite these achievements and the success in winning the global company technology prize for the IM project, there remain significant obstacles to the successful adoption of AWE within the company. Therefore, I carried out a further survey of the IM project team to ascertain their views on the influence of AWE and their perceptions of AWE. All of the interviewees identified benefits in collocating the business and technology developers and clearly they appreciated the benefits of business process re-engineering in Web application development.

The post-AWE pilot survey and the survey of the IM project team, following the second commercial trial of AWE, indicated four major hurdles that arose when trying to get a process specifically for Web engineering, based on AWE, officially adopted within the company for Web-based projects:

1. *Inertia*. The author experienced great resistance to doing anything differently in Web engineering projects, even though many company employees acknowledged the logic of adopting an agile approach for Web-based projects. They also acknowledged the benefits realised by using AWE, such as: evaluation with end-users; collocation of business and technology developers; and the benefits of business process re-engineering.
2. *Culture*. The company's culture was perceived to be a poor fit with AWE and agile process approaches in general.
3. *Poor alignment with the company's funding model*. The funding model is designed for predictive waterfall based mainframe centric development projects. When a project has many unknowns and needs to approach development in an adaptive nature then the funding approach prevents progress as an exhaustive definition of the proposed solution is required to progress past the initial phase.

4. *One Size Fits All Process vs. Specific Processes*. There exists a conflict between the company aspiration for a one-size fits all process, as opposed to processes specific to different types of development activity such as Web-based projects. This conflict ultimately presents a significant research challenge to the Web engineering community who recommend different software engineering approaches including process approaches for Web engineering (Murugesan et al., 2001).

11.5 Further Work

In the short term there is need for further work to develop research results presented in this dissertation. The criteria presented and discussed in chapters 3 and 4 should be enhanced and refined using further evidence. The author would hope that such evidence will come from further surveys and experience reports of Web engineering in practice. Any new Web engineering processes proposed can be evaluated against the criteria and may influence the evolution of the criteria.

Ideally, independent commercial trials of the AWE Process and further testing of some of its principles in commercial projects will help refine and evolve the AWE Process itself. In addition, there is a clear need for experience reports, not marketing material, relating to the successful and non-successful trials of commercial Web engineering processes, including the AWE Process.

There is a need for more investigation into activities and tools to support the AWE Process. At present most of the commercial and academic Web engineering focus is on tools and techniques that aid and assist with the implementation of Web-based systems. Suitable tools and techniques should underpin any development process, however, it is important that the process is defined and understood first, followed by the tools to support the process itself. Although, AWE is tool and technique independent, adopters will clearly benefit from an understanding of how existing tools can support the AWE process. In addition, this work should also potentially identify gaps, where new tools and techniques will be required. For example, tools to support the new impacts, primarily from the software model into the business and domain models, could better assist Web-based development efforts.

The introduction of Web services is a new and exciting research topic that is being predicted to be a major driver in how technology solutions are implemented. However, there is a clear need for research to help understand the impact Web services will have on development process approaches, both within the software model and in the wider context of the business and domain models.

In the longer term, I believe there are some major challenges highlighted by my research. In particular, the commercial trials of AWE process indicate that a better understanding of the importance of business process reengineering when developing Web-based systems is required. There is also a need for better understanding of the challenges in gaining full adoption of new process approaches in large organisations where inertia, conservatism and cultural resistance present serious obstacles to doing so successfully. I believe that these challenges are found in most large organisations, commercial and non-commercial, and that overcoming these obstacles is not purely a software engineering research challenge. In order to successfully understand how to overcome these obstacles research contributions from other related disciplines are required, for example, from business studies and sociology.

Appendix 1: First Survey of Web Engineering in Practice: Contractors' Questions & Answers

The questions and answers presented in this appendix are those recorded as part of 'A Survey of Web Engineering in Practice', that was carried out from October to December 2000. The part of the survey recorded in this appendix was conducted with eleven Web developers from five companies who service Web engineering contracts that are put out to tender, by organisations classified as outsourcers. The interviewees will be referred to using numbers 1 to 11, with each organisation being referred to using the letters A, B, C, D and E. The interviewee sample comprises: three employees from company A, three from company B, one from company C, three from company D and one from company E. All the organisations and individuals that took part in this survey were granted anonymity in order to assist in gaining access to information potentially of a commercially sensitive nature.

The five contractor companies that participated in this survey have Web-based development operations in Scotland. Company A is one of the largest companies servicing Web engineering contracts in the North of the United Kingdom, with over one hundred employees. Company A specialises in thin client business-to-customer e-commerce solutions for organisations primarily in the United Kingdom. Company B specialises in developing front-end solutions, playing to their strengths in the creative design field. Company B employs nine people in their new media team, and has clients throughout the UK. Company B often brings in outside IT contractors to assist with the more technical challenges some large projects present. Company C is a recent start up company in Central Scotland. Company C specialises in e-business solutions to clients in the north of the United Kingdom, focusing on business-to-business and business-to-customer e-commerce solutions. Company D is an Internet Service Provider (ISP) with a small Web development team. Company E is a large IT Consultancy firm with over 300 employees based throughout the United Kingdom. Currently 25% of company E's business is in the development of Web applications. Company E's customers are primarily made up of government and financial service sector organisations.

Each contractor interview was conducted following question's 1 to 22, below. The questions in italics, 1-11 and 13-22, were common to the contractor, outsourcer and in-house questionnaires. Question 12, in bold, is specific to the contractor questionnaire. For outsourcer and in-house questionnaires and answers, see appendices 2 and 3 respectively.

Question 1: '*What is your current job title?*'

1. Programming Team Leader (software engineering background)
2. Technical Research & Development Manager (software engineering background)
3. Producer (creative design background)
4. Account Manager (management background)
5. Technical Programmer (software engineering background)
6. Senior New Media Designer (creative design background)
7. Head of Production (software engineering background)
8. Programmer (software engineering background)
9. Managing Director (medical background)
10. Web Designer (creative design background)
11. Principal Consultant (software engineering background)

Question 2: '*What are your responsibilities within your Web development team?*'

1. Assist in the development of each project's scoping document (requirements and design specification) with clients. Responsible for managing technical developers throughout the rest of the project life-cycle (implementation, testing). Not involved in the maintenance stage of any project.
2. Responsible for exploring, developing and providing middleware solutions.
3. Responsible for the day-to-day management of the creative design aspects of a Web engineering project.
4. Account management, client liaison, process management (preparation of documents between client and company).
5. Responsibilities include the development of technical solutions in each Web engineering project.
6. Responsibilities include management of creative design and direction of each Web engineering project, as well as client liaison and new business direction in the team.
7. Project Management.
8. Client liaison, requirements capture, design, implementation and testing. Note, most sites are designed in such a way that each client can maintain their own presence.
9. Sales and project management occasionally do some development.
10. Provide Web design artwork, and to an extent programming.
11. Chief technical developer on each Web engineering project undertaken by organisation.

Question 3: '*How many developers (inclusive of third party or external developers) make up your average Web development team?*'

1. 8 people
2. 8 people
3. 9 people
4. 2-5 people
5. 7 people
6. 7 people
7. 5 people
8. 5-6 people
9. 5 people
10. 4 people
11. 8 people

Question 4: '*What is the average age of a member of your Web development team?*'

1. 30 years of age
2. 27 years of age
3. 25 years of age
4. 24 years of age
5. 25 years of age
6. 26 years of age
7. 25 years of age
8. 24 years of age
9. 24 years of age
10. 25 years of age
11. 29 years of age

Question 5: '*What percentage of males and females make up your average Web development team?*'

1. Male: 66% Female: 34%

2. Male: 66% Female: 34%
3. Male: 66% Female: 34%
4. Male: 80% Female: 20%
5. Male: 100% Female: 0%
6. Male: 90% Female: 10%
7. Male: 100% Female: 0%
8. Male: 60% Female: 40%
9. Male: 66% Female: 34%
10. Male: 75% Female: 25%
11. Male: 50% Female: 50%

Question 6: *'Break down the number of developers (inclusive of third party or external developers) in an average Web development team into the following roles!'*

Company	Interviewee	Software Engineer	Creative Designer	Team Manager	Business Expert	Domain Expert	Other	Total	Difference in Total from Q3
A	1	2	2	3	1[11]	0	0	8	0
A	2	2	2	3	1	0	0	8	0
A	3	3	3	1	1	1	0	9	0
B	4	1[12]	3	1	0	1	0	6	1
B	5	1	4	2	0	0	0	7	0
B	6	2	2	1	2	1	0	8	1
C	7	2	1	1	1	0	0	5	0
D	8	1	3	2	1	1	0	8	3
D	9	1	3	1	0	0	0	5	0
D	10	1	2	1	0	0	0	4	0
E	11	10	1	1	1	4	0	17	9
Average[13]		**2**	**2**	**2**	**1**	**1**	**0**	**8**	**1.27**

Question 7: *'What is the average length of a Web development project, from inception to first delivered working system?'*

Interviewee	1-4 Weeks	1-3 Months	3-6 Months	6+ Months
1		X		
2			X	
3			X	
4		X[14]		
5			X	
6		X		
7			X	
8		X		
9		X		

[11] Often there can be two or three business experts in a Web project.
[12] A software engineer is required only in one third of projects undertaken.
[13] With the exception of the last column, the figures in this row are rounded to the nearest person.
[14] Normally it only takes 1-3 months to develop a project from requirements to deliverable of final Web application. However it often takes as long to successfully get a contract as it does to develop the project.

10	X			
11			X	

Question 8: *'How often do your Web development projects run over time?'*

1. Did not answer.
2. 75% of the time.
3. 50% of the time.
4. 25% of the time.
5. 0% of the time.
6. 0% of the time.
7. 100% of the time.
8. 50-74% of the time.
9. 50-74% of the time.
10. 50-74% of the time.
11. 0-24% of the time.

Question 9: *'What are the reasons for projects running over time?'*

1. Did not answer.
2. Changes made by clients to project scoping document during development of project.
3. Lack of preparation and understanding on behalf of client, major reason for projects running over is poor communication between client and internally between different members of the development team.
4. Changing scope of project by clients, due to lack of knowledge of effort required in order to build a Web system. Poor allocation of resources on client's behalf for provision of content. Also conflict arises early in project between educating client and providing free consultancy.
5. Not applicable.
6. Not applicable.
7. Getting content from clients.
8. Mostly due to clients not producing content, sometimes though due to poor project management.
9. Poor prediction of effort, heavy workload, and clients not producing content, also not testing properly.
10. Mainly lack of content from clients.
11. Poor co-operation from clients, often content is not delivered in time. Also, two contracts can conflict in terms of time.

Question 10: *'How often do your Web development projects run over budget?'*

1. Did not answer.
2. Did not answer.
3. 50% of the time.
4. 0% of the time.
5. 0% of the time.
6. 25% of the time.
7. 50% of the time.
8. Don't know.
9. 5% of the time.
10. 25% of the time.
11. 0-24% of the time.

Question 11: *'What are the reasons for Web engineering projects running over budget?'*

1. Did not answer.
2. Did not answer.
3. Client changes to scoping document, during the project development.
4. Not applicable.
5. Not applicable.
6. Client changes in scoping document. Often these take the form of additions rather than changes.
7. Poor allocation of resources on client's behalf. Most clients have a fixed budget!
8. Not applicable.
9. Clients not delivering.
10. Lack of discipline on behalf of clients and developers.
11. Projects rarely run over budget as every project is fixed price.

Question 12: **'Do you ever find yourself short of development expertise in a Web development project?'**

1. Yes. Getting domain expert to provide content in a suitable format and to agreed time scale.
2. Yes. Getting domain expert to provide content in a suitable format and to agreed time scale.
3. Yes. Technical expertise is often spread pretty thinly across a number of projects in development in parallel.
4. Yes. Technical experts who can work in team centred on creative design skills.
5. Yes. Short on technical expertise.
6. Yes. Often short on technical expertise.
7. Yes. Lack of understanding of business practices on behalf of management, technical and creative design developers.
8. No.
9. Yes. In all areas.
10. No.
11. Often there is not enough technical expertise to deal with all concurrent competing projects.

Question 13: *'How many Web projects proposed result in a delivered system?'*

1. Not Sure.
2. Not Sure.
3. Did not answer.
4. 51-75% of projects.
5. 76-100% of projects.
6. 51-75% of projects.
7. 76-100% of projects.
8. Don't know.
9. 51-75% of projects.
10. 76-100% of projects.
11. 26-50% of projects. (Includes projects that go to other vendors).

Question 14: *'What are the prime reasons for projects not resulting in a delivered Web system?'*

1. Clients not having the budget.
2. Not applicable.
3. Did not answer.

4. Lack of budget on client's behalf, changes on client's behalf, or loose on pitch situation.
5. Client's scope is not clearly defined.
6. Lack of budget on client's behalf, lack of understanding in terms of the cost of technology.
7. Did not answer.
8. Not applicable.
9. Lack of budget 99.9% of the time.
10. Not applicable.
11. Pre-selected supplier, price is too high and requirements are often unclear.

Question 15: *'Do you have a well-defined and documented development process for building Web engineering projects?'*

1. Not really. Still in development, although some procedures are in place, primarily a scoping document that includes requirements and design. Both parties sign off on scoping document before project can continue.
2. Not really. Primarily focusing on backend solutions after involvement in scoping document tends to remove oneself from specific projects. Uses UML to model design of middleware solutions.
3. Not really. Primarily uses site maps for front-end and back-end implementation guide from scoping document.
4. Yes. Phases include requirements document, creative design proposals, design development, creation and implementation. No explicit testing stage, testing is integrated into creation and implementation stage. Maintenance is not considered as part of a project's development life-cycle.
5. Not really involved in any stage except creation and implementation, often uses design documentation and templates.
6. Assists with definition of Brief, Costing Model, and Design Development.
7. Well-defined process, not well documented. The only documentation is the requirements and design information that is delivered in one document.
8. Not really, mostly requirements, technical and design specifications.
9. We have a 'Document Pack' which includes: quotation and project definition, contract, design and implementation.
10. Research -> Development -> Conclusion -> Finalise.
11. Yes. Bespoke process. Business Analysis stage (sometimes already done by client) followed by requirements specification and risk analysis that comes in the form of a Project Initiation Document, and finally Test Document.

Question 16 – *'What features of your Web development and design process do you consider successful?'*

1. The scoping document.
2. Breaking up the development teams into smaller groups.
3. Task management and site maps.
4. Use of templates.
5. Creative process works well.
6. Good communication with clients.
7. All works well.
8. Small teams, good communication with clients.
9. Client content summary sheet. Documenting and getting clients to agree to the project specifications.
10. Them all.
11. Independent Project Advisor (who is not part of the development team), and a separate Quality Analysis Team.

Question 17: *'What features of your Web development and design process do you consider problematic?'*

1. Need for more flexibility in the process. Although one of the problems is that if you allow clients to change one aspect of the scoping document, they think they can change everything.
2. Major problem communicating with clients.
3. Need for better communication with clients, most clients don't understand the process.
4. No clear definition of process for clients to view, lack of communication with clients.
5. Managing the development process, keeping focused on which points of the job are crucial.
6. Better technical expertise, and a need to be stricter with clients.
7. Biggest problem is client management.
8. Don't think we produce enough documentation.
9. Getting clients to stick to the process.
10. Too many ideas.
11. Creative Design Aspects of Web design.

Question 18: *'If you do not use a Web development process, why not?'*

1. Not relevant.
2. Not relevant.
3. Not relevant.
4. Not relevant.
5. Not relevant.
6. Not relevant.
7. Not relevant.
8. Have a project life-cycle plan, very loose, need a process that is well defined not well documented.
9. Not relevant.
10. Not relevant.
11. Not relevant.

Question 19: *'What software tools do you use when developing a Web engineering project, what tasks do you use them for, and at what stage in the development are they used?'*

1. PROJECT DEFINITION: Outlook Express, Microsoft Project, Excel and Word. DESIGN: Photoshop, Freehand (for site Maps). IMPLEMENTATION: Cyberstudio and Dreamweaver for HTML creation, BBEdit or Notepad for HTML and PHP Scripts, Photoshop, FTP client, Flash, Director if there is a game to be produced, Internet Explorer and Netscape (normally just the most recent versions). TESTING: All testing is outsourced to a company who resides outside the United Kingdom. Testing is outsourced due to the lack of suitable testing suite in-house. MAINTENANCE: Same tools as implementation and CVS. A separate team to development handles all maintenance.
2. PROJECT DEFINITION: DESIGN: Primarily using UML for modelling middleware designs. IMPLEMENTATION: Forte for Java (Java IDE) because of its support for CVS. Primarily JavaBeans development to support thin client architectures. TESTING: Internally developed testing suite. MAINTENANCE: Same as implementation.
3. PROJECT DEFINITION & DESIGN: Microsoft Word, Excel, Project, Illustrator, Photoshop, Visio (for site maps), Flash, Dreamweaver. IMPLEMENTATION: Same

as project definition and design, Internet Explorer and Communicator. TESTING: Internet Explorer and Communicator. MAINTENANCE: Same as implementation.

4. PROJECT DEFINITION: Internet Explorer, Netscape, Outlook Express, Office 2000, Microsoft Project. DESIGN: IMPLEMENTATION: TESTING: MAINTENANCE:
5. HTML Editor, Dreamweaver and Interdev, Photoshop, Outlook Express, Internet Explorer and Netscape, SQL Development client, Office 2000, Flash.
6. Photoshop 5.5, Illustrator 8.0, Freehand 9.0, Flash 4.0 & 5.0, Fireworks 3.0, Dreamweaver 3.0, Ultradev, BBEdit, Homesite, FTP Clients, Outlook Express, Office 98, Acrobat Distiller and Viewer, and Director.
7. PROJECT DEFINITION: Microsoft Project, Office 2000. DESIGN: Photoshop, Illustrator, Zara, Quark Express. IMPLEMENTATION: ColdFusion, Homesite, SQL Server, Microsoft Office 2000, Flash 4.0 & other small freeware and shareware tools. TESTING: Word, Internet Explorer versions 4 and 5, Netscape versions 3, 4, and 5. MAINTENANCE: VNC.
8. PROJECT DEFINITION: Pine, Netscape, OFFICE 2000. DESIGN: Photoshop, BBEdit, Flash, Freehand, and pine, Netscape. IMPLEMENTATION: Sitepad Pro for VRML, Perl, Java, PHP, HTML.TESTING: Netscape and Internet Explorer. MAINTENANCE:
9. PROJECT DEFINITION: Pine, OFFICE 2000, Acrobat. DESIGN: Illustrator, Word and PDF. IMPLEMENTATION: Photoshop, Freehand, Illustrator, Fireworks, BBEdit, Sun's JDK, PHP 4.0, Perl, JVM, Postgress, and Access. TESTING: Netscape, Internet Explorer on Mac OS, Win 32 and Linux. MAINTENANCE:
10. DESIGN & IMPLEMENTATION: Photoshop, illustrator, freehand, BBEdit and Flash.
11. Front Page or ColdFusion. Serano for volume Web testing.

Question 20: *'What types of conflict arise between different developers in a Web development project?'*

1. We often find it hard to communicate technical issues within team. Also find difficulty in communicating project constraints to the client.
2. Conflicts in coding and development style. Happens more often when contractors are brought in.
3. Always a trade-off between function and form.
4. Biggest problem is people from a design background not understanding business objectives. Also many technical developers who do not rate the importance of design.
5. No problems system works really well.
6. Conflicts between technical and creative developers and content providers. Also function vs. form arguments.
7. Not answered.
8. Don't really have any!
9. Very few, good working relationship.
10. Arguments over tools.
11. Couldn't think of any. Main disagreements come in terms of time and effort prediction.

Question 21: *'How are conflicting views resolved between different developers in a project?'*

1. Resolve issues between ourselves.
2. No answer.
3. Trace conflict back to scoping document and resolve according to which solution best meets requirements.

4. Educate team by going back to scoping document, and resolve conflict according to which decision best meets project goals.
5. Not applicable.
6. Meet internally and find a compromise that works.
7. Resolve at management level, educating team members at the same time.
8. Not applicable.
9. Not applicable.
10. Winner usually proves his point.
11. Team Manager and Client resolve all issues.

Question 22: *'What mechanisms do you employ to measure the success of a project? At what stage in the development process do you employ these mechanisms?'*

1. Success is measured by client satisfaction with site, and is linked to client payment.
2. Two weeks before project launches the technical specification is tested and the client goes through site.
3. Achieving proposed time scale and budget for project in question. Note workload is measured in hours.
4. No formal project effectiveness mechanism in place. Use client feedback, profit margin, repeat business and referrals to measure success.
5. No set mechanisms, primarily consider a project successful if client is happy.
6. Projects success is measured by client's satisfaction with project, also a project is considered successful if it gains good PR for firm.
7. Repeat business, often customer goals are so vague that it is difficult to measure whether or not their needs have been addressed. There is quite a big conflict between educating the client as to what they need over what they want.
8. If site works quickly on low bandwidth connections, if it's delivered on time and if the client's happy.
9. Client feedback, longevity of deliverable and relationship with client.
10. If the end product works and is consistent across all platforms and browsers.
11. Make sure project is well-defined (personal commitment made by all project managers). Establish change control mechanisms. Use separate Quality and Testing teams.

Appendix 2: First Survey of Web Engineering in Practice: Outsourcers' Questions & Answers

The questions and answers presented in this appendix are those recorded as part of 'A Survey of Web Engineering in Practice', that was carried out from October to December 2000. The part of the survey recorded in this appendix was conducted with two Web developers from one organisation who outsource the majority of their Web engineering contracts. The numbers 12 and 13 refer to the interviewees in the order they were questioned. All the organisations and individuals that took part in this survey were granted anonymity in order to assist in gaining access to information potentially of a commercially sensitive nature.

The organisation represented by interviewees 12 and 13 is based in Scotland, with over 400 employees in their main office, and a further 1200 employees in other offices throughout the region. Each contractor interview was conducted following the question's 1 to 26, below. The questions in italics, 1-11 and 17-26, were common to the contractor, outsourcer and in-house questionnaires. Questions 12-16, in bold, are specific to the outsourcer questionnaire. For contractor and in-house questionnaires and answers, see appendices 1 and 3 respectively.

Question 1: *'What is your current job title?'*

12. IS Manager (software engineering background)
13. Project Manager (business management background)

Question 2: *'What are your responsibilities within your organisation?'*

12. Manage the software application development for the organisation, including both in-house and external projects.
13. Lead business transformation within organisation. E-enable entire organisation by 2003.

Question 3: *'How many people within your organisation are involved in a typical Web engineering project?'*

12. 20 full time developers, plus an additional 20 out with the IS department.
13. 120 in total, which will be split into smaller teams as e-enabling develops.

Question 4: *'What is the average age of people, within your organisation, who are involved in a typical Web engineering project?'*

12. 25–30 years of age.
13. 30 years of age.

Question 5: *'What percentage of males and females make up the people who are involved in a typical Web engineering project?'*

12. Male: 80% Female: 20%
13. Male: 40% Female: 60%

Question 6: *'Classify the types of people involved in a typical Web engineering project in your organisation!'*

12. Software Engineer: 10, Creative Designer: 0, Team Manager: 4, Business Expert: 2, Domain Expert (content provider): 0, Other: 4 (1 database analyst, 1 technical author, 2 report writing experts).
13. Not applicable at this stage, too early in project to classify.

Question 7: *'What is the average length of a Web development project, from inception to first delivered working system?'*

Interviewee	1-4 Weeks	1-3 Months	3-6 Months	6+ Months
12		X		
13		X		

Question 8: *'How often do your Web development projects run over time?'*

12. 0-24% of the time
13. 25-49% of the time.

Question 9: *'What are the reasons for projects running over in terms of time?'*

12. Not being able to properly define requirements.
13. Not Answered.

Question 10: *'How often do Web development projects run over budget?'*

12. 0% of the time.
13. 25% of the time.

Question 11: *'What are the reasons for projects running over budget?'*

12. Not applicable.
13. Not properly managed, not tied into business process.

Question 12: **'What are the primary reasons for outsourcing a Web engineering project?'**

12. One, don't have the necessary technical skills in-house. Two, time-scale is too short for in-house teams to cope with.
13. Skills not found in-house, not organisation's core competence, when time is a critical factor (employ organisation with ten developers for 6 weeks) as opposed to 2 people for 7 and ½ months.

Question 13: **'What criteria do you look for in vendors tendering for a Web engineering project?'**

12. Evidence of a quality approach to development (BSI, Certification, and Quality Manual), Similar technical framework, established track record in the field.
13. Evidence of necessary skills, proven track record, use of similar development technologies to in-house teams. Also good proposed project cost, and evidence of innovative approach in terms of solutions delivered.

Question 14: **'Describe the process of putting a Web engineering project out to tender?'**

12. Due to the short time scales in Web engineering it takes too long to assess each vendor on a project by project basis. As a result every two years, four vendors are selected to become preferred suppliers. Approved suppliers are given a requirements specification with a fixed budget. If they agree to take the project on, a testing and evaluation document is produced by the IS Team, while the project is in the design and implementation stage, this is given to the vendors only if requested! The organisation then use the testing and evaluation document as the metric the deliverable must pass in order for the project to be accepted.
13. Identify business problem, do business analysis, invite parties to tender, evaluate tenders, through presentations and written proposal which are evaluated by panel, refine list of vendors, interview and then commission project.

Question 15: **'Describe the process of selecting a vendor for a Web engineering project!'**

12. See question 14.
13. Gain executive approval, create panel to evaluate vendors, short list, mark and score each short-listed vendor's proposal, then select supplier.

Question 16: **'Describe the process of accepting the final deliverable of an outsourced Web engineering project!'**

12. See question 14.
13. Assess what is actually delivered compared to detailed brief.

Question 17: *'How many proposed projects result in a delivered system?'*

12. 100%
13. Not enough!

Question 18: *'What are the prime reasons for a project not resulting in a delivered Web system?'*

12. Not applicable.
13. Not a rigorous enough appraisal process, typically not enough business analysis done.

Question 19: *'Do you insist on a well-defined and documented development process or methodology for building Web sites?'*

12. Yes. See question 14.
13. Yes and No. Our department would, but organisation does not always do so.

Question 20: *'What features of your Web development and design process or methodology do you consider successful?'*

12. Documentation. Note can't always get the requirements and proposal well defined.
13. Understanding that it is about business processes, not technology, need to be very specific about functionality and business needs. Technology seduces too many people.

Question 21: *'What features of your Web development and design process or methodology do you consider problematic?'*

12. It is not possible to alter the requirements after they have been signed off.

13. Business buy-in, ownership of project deliverable, lack of clear leadership, and poor ability to manage change in both business and IT.

Question 22: *'If you do not use a Web development process or methodology, why not?'*

12. Not Applicable.
13. Not Applicable.

Question 23: *'What software tools do you use when you are involved in a Web engineering project, what tasks do you use them for, and at what stage in the development are they used?'*

12. Outlook Express, IE5, Netscape, NT 4.0, OFFICE '97, Rational Rose, Visual Interdev, Visual Studio, Visual Basic, JavaScript, Visio, XML, ASP, Site Server, Ingress, SQL Server, Oracle.
13. OFFICE, TEAM PHONE, Project, GIS System.

Question 24: *'What types of conflict arise in your organisation and with the vendor in a typical Web development project?'*

12. Getting internal clients to sign off on projects. 'Getting the requirements right is the hardest part of the process'
13. Often technological solutions do not address business goals, need better defined business goals. Also conflicts arise when people want to change scope of project after requirements have been signed off.

Question 25: *'How are these conflicts resolved?'*

12. Sign off projects without client approval, or educate them, finish this project and start another to deal with issues!
13. Use of Standards Procedure incorporating design and technical architecture, before project budget can be spent.

Question 26: *'What mechanisms do you employ to measure the success of a project? At what stage in the development process do you employ these mechanisms?'*

12. If a project is delivered on time then it is seen to be successful. Note: use expert opinion and analogy for project effort prediction and estimation.
13. Cost Model, use base-case to measure benefits of deliverable. Need to track return y on investment x. New role being created to monitor this.

Appendix 3: First Survey of Web Engineering in Practice: In-house Questions & Answers

The questions and answers presented in this appendix are those recorded as part of 'A Survey of Web Engineering in Practice', that was carried out during October to December 2000. The part of the survey recorded in this appendix was conducted using two Web developers from a financial services sector company with business interests worldwide. Both developers are based in Central Scotland with responsibilities on projects that target audiences throughout the European Union. The numbers 14 and 15 refer to the interviewees in the order they were questioned. All the organisations and individuals that took part in this survey were granted anonymity in order to assist in gaining access to information potentially of a commercially sensitive nature.

Each in-house interview was conducted following the question's 1 to 24, below. The questions in italics, 1-11 and 15-24, were common to the contractor, outsourcer and in-house questionnaires. Questions 12-14, in bold, are specific to the in-house questionnaire. For contractor and outsourcer questionnaires and answers, see appendices 1 and 2 respectively.

Question 1: *'What is your current job title?'*
14. Project Manager (software engineering background)
15. WebZone Manger (business management background)

Question 2: '*What are your responsibilities within your organisation?*'
14. Responsible for the delivery of solutions on time, to budget, that meet project requirements.
15. Business and Line manager for WebZone. Organise virtual teams for each project. Note we use a matrix structure for populating teams.

Question 3: *'How many developers (inclusive of third party or external developers) make up your average Web development team?'*
14. At the moment 20 people make up the WebZone team. Approximately 6 people per project (for example, the Intranet project).
15. 25 people in total make up the WebZone team. 2-5 developers on an average team although this figure could rise to 30 on large projects.

Question 4: *'What is the average age of a member of your Web development team?'*
14. 27 years.
15. 24 years.

Question 5: *'What percentage of males and females make up the people who are involved in a typical Web engineering project?'*
14. Male: 50% Female: 50%
15. Male: 65% Female: 35%

Question 6: *'Classify the types of people involved in a typical Web engineering project in your organisation!'*

14. Software Engineer: 4, Creative Designer: 2, Team Manager: 4, Business Expert: 0, Domain Expert (content provider): 0, Other: 2 people testing, 4 Research and Development technologies, 4 coders.
15. Software Engineer: 0.5, Creative Designer: 1, Team Manager: 0.2 Business Expert: 1, Domain Expert (content provider): 0.3, Other: 0,

Question 7: *'What is the average length of a Web development project, from inception to first delivered working system?'*

Interviewee	1-4 Weeks	1-3 Months	3-6 Months	6+ Months
12		X		
13		X		

Question 8: *'How often do your Web development projects run over time?'*

14. 0-24% of the time.
15. 0-24% of the time.

Question 9: *'What are the reasons for projects running over in terms of time?'*

14. Scope creep and business indecision.
15. It takes too long for approval and implementation.

Question 10: *'How often do Web development projects run over budget?'*

14. 1-25% of the time.
15. 1-25% of the time.

Question 11: *'What are the reasons for projects running over budget?'*

14. Unforeseen costs (e.g. having to buy additional encryption modules). Also existing infrastructure proves unsuitable.
15. Generally, business requirements are not solid.

Question 12: **'Do you use any outside organisations to assist in the development of your Web projects? If so who are they and why do you use them?'**

14. YES. TATA Consultancy Services for extra developers. Buchanan International for security audits and 'ethical hacking'.
15. YES. TATA Consultancy Services for extra developers, prefer to manage developers in-house as opposed to outsourcing

Question 13: **'What internal departments play a role in the development of your Web engineering projects? What do these departments contribute to projects?'**

14. Sales & Services win the business and manage the customer relationship. Program Office record metrics for the projects. Testing & Integration verify the project deliverables and ensure that they operate smoothly with existing IT infrastructure. Service Delivery manages the Web-based systems. Planning & Consulting check architectures.
15. Electronics & Telephonics. Products & Processes. Marketing Services. Service Delivery. E-enablement (1 person). Payments.

Question 14: **'Do you ever find yourself short of development expertise in a Web development project?'**

14. YES. Lack of people keeping their eye on new technologies, e.g. Java, XML and WAP.
15. YES. Technical skills generally pull people in from TATA.

Question 15: *'How many projects proposed result in a delivered system?'*

14. 51-75%
15. 51-75%

Question 16: *'What are the prime reasons for projects not resulting in a delivered Web system?'*

14. Lack of budget, unable to justify development costs!
15. Lack of Funding. Annual budget system restricts development.

Question 17: *'Do you have a well-defined and documented development process for building Web sites? How does this process begin and end?'*

14. YES. We use Dynamic Systems Development Method DSDM™ (http://www.dsdm.org) for all Web-based development projects. DSDM™ is a Rapid Application Development approach, unfortunately there are very few people within the organisation who are comfortable with it.
15. YES. Use DSDM™ after receiving operational vision. Stop when business clients are satisfied, and then do post implementation review. Difficult to do verification and validation as this takes 6 months on non Web-based projects, and we often just have a couple of weeks.

Question 18: *'What features of your Web development and design process do you consider successful?'*

14. DSDM™ allows close working with business people, bringing business on board. The iterative and incremental approach allows for market pressures enabling soft launches of Web-based systems.
15. Project Definition Workshop that takes the form of a structured meeting.

Question 19: *'What features of your Web development and design process do you consider problematic?'*

14. Managing scope is still difficult under DSDM™, also business people rebel against existing techniques, such as exhaustive testing and integration procedures as this takes a great deal of time.
15. Lack of understanding of traditional software infrastructure, and poor end-to-end testing.

Question 20: *'If you do not use a Web development process, why not?'*

14. Not applicable.
15. Not applicable.

Question 21: *'What software tools do you use when developing a Web project, what tasks do you use them for, and at what stage in the development are they used?'*

14. PROJECT DEFINITION: Microsoft PowerPoint, as it forces business people to bullet point! DESIGN: Not aware of any design tools. IMPLEMENTATION: Domino, Dreamweaver, Flash, Apache, CORBA, SOAP, IIS, ASP and MQ as middleware solution to legacy infrastructures on IBM and Compaq Mainframes. TESTING: Endeavour, Test Director. MAINTENANCE:
15. Not answered.

Question 22: *'What types of conflict arise between different developers in a Web development project?'*

14. Undercurrent of people who denigrate graphic design. Also major conflicts between traditional IT developers and Web developers.
15. Often traditional IT developers over engineer, do not know when to get off the development life-cycle. In addition, one of the developers is blind and complains regularly about poor support for disabled users.

Question 23: *'How are conflicting views resolved between different developers in a project?'*

14. Get people working closer together. Educate people that Web development is different from tradition IT development. Also a Web team breakfast.
15. Conflicts are resolved by line managers.

Question 24: *'What mechanisms to you employ to measure the success of a project? At what stage in the development process do you employ these mechanisms?'*

14. Program Office metrics, function point analysis. Measure efficiency of developers. Questionnaire for receiving customers, we did this with a sample of six customers for our new WAP site.
15. Post implementation testing.

Appendix 4: The Agile Manifesto

"Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." - Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas.

Appendix 5: Principles behind the Agile Manifesto

"Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer
through early and continuous delivery
of valuable software.

Welcome changing requirements, even late in
development. Agile processes harness change for
the customer's competitive advantage.

Deliver working software frequently, from a
couple of weeks to a couple of months, with a
preference to the shorter timescale.

Business people and developers must work
together daily throughout the project.

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

The most efficient and effective method of
conveying information to and within a development
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly."

Appendix 6: Pre-AWE Pilot: Questions

The questions presented in this appendix are those used in the Pre-AWE Pilot survey carried out in February 2002. The interviewees were asked the following eleven questions:

Question 1 – 'Are you familiar with the organisation's process? If so please describe the organisation's process?'

Question 2 – 'How do you perceive your involvement with the organisation's process?'

Question 3 – 'Are you aware of the rapid application development life-cycle within the organisation's process?'

Question 4 – 'How would you like to see the organisation's process evolving in the near future?'

Question 5 – 'What features of the organisation's process do you consider successful?'

Question 6 – 'In what areas of the organisation's process would you like to see improvement?'

Question 7 – 'How well does the organisation's process integrate with your standard processes?'

Question 8 – 'Identify the stakeholders/entities typically involved when using the organisation's process?'

Question 9 – 'What, in your opinion, are the major advantages of using the organisation's process?'

Question 10 – 'What, in your opinion, are the major disadvantages of using the organisation's process?'

Question 11 – 'Are there any known limitations to the organisation's process?'

Appendix 7: Post-AWE Pilot Survey: Questions & Answers

The questions and answers presented in this appendix are those used in the Post-AWE Pilot survey carried out in September 2002. The questions and recorded answers are listed below for the eight interviewees, referred to as hereafter as interviewees 1-8:

Question 1 – 'What is your current job title?'

1. Head of Internet Channel
2. Technology Business Partner
3. Information Security Analyst
4. Technology Consultant
5. Lead Technology Consultant
6. Lead Technology Consultant
7. Lead Technology Consultant
8. Technology Business Consultant

Question 2 – 'How long have you been involved in Web Development?'

1. 3 years
2. 0 years
3. 3 years
4. 4 years
5. 2 years
6. 8 years
7. 2 years
8. 3 years

Question 3 – 'What type of IT development background do you have?'

1. None. Most of the role does not involve IT.
2. My job currently involves liaising between the business and technology units within the organisation, acting as a catalyst for instigating new projects and assisting with strategic business direction. My background is from business transformation that has often required working with technology, particularly on business process re-engineering projects.
3. Nineteen years primarily in IT security. Traditional programming experience with COBOL/CICS on mainframe and menu driven systems on client.
4. Started carrying out PC systems installation (boxes, cables and operating systems), moved onto training. Then Systems Analysis involving requirements capture and design. Moved onto being involved in Web Development, that included Webmaster role, creating Guidelines and Standards for Web-based development, Web-development for a number of local government and commercial Web sites. Recently have been in an architecture role for Web-based and traditional IT systems.
5. Eleven years experience. Application development, Infrastructure planning, more recently the role of an IT Architect.
6. Eighteen years IT experience. In chronological order: started off as an assembler programmer; then moved to network designer/implementer/trouble-shooter; then communications application designer and implementer; Web development; current role is in architectural strategy.
7. Background mainly in systems analysis, primarily large scale mainframe-based systems. Recently working as a technical consultant/architect.

8. Five years experience in IT projects. Two years on a project to design and implement a call centre and back office administration system, that involved a lot of business process re-engineering. One year consulting looking at how to change business process to cope with Web-based and call-centre based financial service applications, in a number of organisations. One and a half years with a SME developing XML and PKI standards to facilitate e-commerce trading for Life Insurance companies. Spent the past twelve months on Identity Management for this organisation.

Question 4 – 'In your experience do you notice any major differences between Web-based development and Tradition IT Projects?'

1. I see no difference in practice but see the need for a different approach. Web-based development requires greater speed-to-market. Need change in attitude from Technology within the organisation, from can't do, to can do. Technology and business lack skills and understanding in Web-based application development. Lack of leadership in embracing new technologies/opportunities/challenges. Often I can get something developed via a third party for a third of the price, and still find better understanding skills in Web-based development than currently exist in-house.
2. Not in this organisation.
3. Differences exist in speed-to-market, customer interface and security issues. With respect to security, if an internal application is breached, then the cracker is more tied in the damage that can be done to company reputation. In Web-based development there is more scope for damage which can destroy trust, if integrity has been breached.
4. Yes. Speed-to-market is more important. Often it is expected that the project will start and finish within weeks as opposed to months. Requirements are more volatile. Prices are expected to be lower. Team make up is different 'need to get your hands dirty on every aspect of the system design'. Variation in team role is much greater. Process is more flexible because it is new. On past Web projects, never used any formal process 'cherry picked what we needed to achieve what we wanted'.
5. Whole of this organisation is geared up for mainframe-centric development only. Any other type of development is not well supported.
6. Yes. Main difference is the importance of the end-user. Less flexibility with traditional IT. The ubiquitous nature of the Web has a huge impact. Web requires a shift to more standards-based development, less proprietary. Web interface is the customer touch point. Complexity is greater in terms of path length, i.e. number of tiers, number of components, level of abstraction, potential for re-use etc. Web-based development impacts a wider percentage of the general population in terms of usage and development, therefore the global development peer group is greater.
7. Within this organisation there is less Web development expertise in-house. Better in-house experience for traditional IT projects. Traditional IT is very pattern intensive. Web-based standards and patterns were not perceived to be as mature as they are for traditional IT projects/technologies. Web-based architectures are more important and there is more room for error.
8. Huge differences between previous Web-based experience, for example, the time-to-market pressures were greater in previous organisations I worked for and this was achievable as we had no legacy dependencies. Previous observations from working in other organisations showed that the processes were lighter than in this organisation and the teams were smaller.

Question 5 – 'What are your responsibilities within your Web development team?'

1. My responsibilities include day-to-day operations surrounding our online Web applications in Europe, including on going development, in addition to sales and marketing through Internet Channels.
2. Not Applicable.
3. To ensure that the security policies and standards are enforced. In addition, my job also involves assessing the viability of a proposal overall from a risk perspective.
4. Degree of consultancy, advising on particular areas of concern. I'm expected to provide my experience form previous projects.
5. I consult on the design and also ensure the adherence to standards.
6. I'm responsible for Architectural Governance ensuring: consistent architecture; maximum possible re-use; that there is no vendor lock-in; performance and scalability; simplicity and security.
7. Responsible for architectural governance making sure that standards and procedures are being adhered to. This is more difficult in Web-based projects as this is a new area with many volatile parts where the standards, process and experience are not as mature as other areas of IT development.
8. Strategy, guidance, convincing people of the merits of the strategic direction. Often there is a need to sell Web-based technologies.

Question 6 – 'What do you think of the organisations current development process and its applicability to Web-based development?'

1. We often assume as an organization that we can do it all ourselves, however, I believe this is flawed. Often we need to bring skills in-house. Often we waste time and therefore money. I can often spend the same amount of money with a third party as it costs me to get the requirements and design phases completed in-house because of our development process.
2. I think our current process is cumbersome and overly bureaucratic. It does not lend itself well to change. There is too much of a divide between those who will use and/or be impacted by the deliverables and the development team.
3. Cumbersome. Outsourcing conflicts with the achievement of speed-to-market on projects.
4. This is primarily a COBOL shop, I would be amazed if we could kick a Web-development project off and follow our process without serious pain.
5. Too rigid. It is essentially a waterfall based process. It is not understood by the business within the organization. Time-to-market pressures are not addressed due to the inherent red tape in the current process.
6. Does not support experimentation. We need to involve our customers more to ensure that solutions are effective, wanted and needed. We are far too removed from customers, and therefore have to guess too much.
7. Not clearly articulated and understood. Does not lend itself to iterative and incremental development. Too focused on documentation. Not time-to-market friendly. Not convinced that it is appropriate for all areas of IT development.
8. Heavy and bureaucratic, unwieldy with very little flexibility. If we had a good process we could easily shrink staff by 10%. No willingness to change.

Question 7 – 'Have you read the AWE Process Technical Report? YES / NO'

1. Yes
2. Yes
3. Yes
4. Yes
5. Yes
6. Yes

7. Yes
8. Yes

Question 8 – 'What do you perceive as the strengths of the AWE Process?'

1. The development approach should help us streamline our current practices and help us get to market quicker. It will allow us to build usability in as mandatory and thus make people accountable for usability issues. In my experience it costs us more to avoid usability.
2. Iterative and Incremental approach. Involvement of all parties at all stages. Introduction of an evaluation phase. I would consider the AWE process approach applicable to more than just Web-based projects.
3. Iterative and Incremental approach. Flexibility, allowing, for example, to start from Business Analysis or Evaluation is one of its strengths. Emphasises involvement of the user community which will help ensure that it is functional and friendly.
4. I see the strengths of the AWE process as follows: an iterative and incremental agile process approach that allows developers to address the challenges presented by Web development; the fact that the process is not a one size fits all attempt to solve process problems but focuses on Web development; and that it learns the lessons from previous Web-based projects.
5. Collaboration. End-user involvement. Every developer involved from the start. Focus on the highest risks first. Communication through the browser experience. Sets out good high-level principles.
6. Simplicity. Common sense approach. Generic, therefore no dependency on tools, techniques or patterns that might date. Its involvement of end-users and iterative and incremental approach should help find problems early.
7. One strength is that it gets across a lot of information in a short document. It does not impose techniques, enabling one to fit with current techniques. Puts down in clear writing things that need to happen during the development of Web-applications and appear obvious but most projects miss. Strong focus on the end-user and the system being the key thing that is being delivered.
8. It's flexible. Addresses time-to-market pressures. It will enable the necessary empowerment to those who are tasked with the job. Should improve morale by making projects more enjoyable and allowing projects to move forward. Should cut costs.

Question 9 – 'What do you perceive as the weaknesses of the AWE Process?'

1. We would need to ensure that all the risks were covered, and proper controls were put in place to support AWE, particularly with respect to testing to ensure that we do not cut corners. Currently, the organisation's development process has no measure of quality.
2. Needs organisational buy-in before it can be adopted. It is probably seen as a threat by traditional developers. Potentially it could allow areas of development to be open to interpretation, which could potentially allow some people to abuse the flexibility provided by AWE.
3. Don't see weaknesses with the process itself. Project management and ultimately the quality of the developers will determine the success of the project. This will require good people who are not blinkered by the predictive approach. Good leadership is essential to successful adoption.

4. One weakness is that AWE has not been used in a real project. This is perceived as a risk in an organisation like this one. Another weakness is that AWE requires a cultural shift, which is easier to write about in paper than to happen in reality.
5. Needs cultural and organisational change in order for an organisation like this to adopt it, need to shift culture from a predictive to adaptive. AWE has not been proven in a real world project as yet.
6. 'Possibly not prescriptive enough, potentially it allows for too much latitude. I find the business model and domain model similar.
7. I think that to make it work you would need an AWE Process evangelist in every team. Like all processes, AWE needs proper training and expertise across the organisation. Currently perceived to be a problem with our process. There might be issues on a large project crossing geographical boundaries.
8. Need to have a company that is forward thinking and is willing to change. Need to have an employee culture where employees will want to change. Not suitable within the financial service sector. Relies on staff that are good. Poorer skilled developers might struggle more and managers might feel that they are losing control.

Question 10 – 'What obstacles/environmental conditions do you see to adopting the AWE Process within this organisation?'

1. Not answered.
2. Resistance to change. Ignorance. Needs senior management buy-in. Organisational changes at the highest level have hindered AWE's adoption.
3. Rigid existing approach. Blinkered approach to development. A lack of willingness to try something new, as this is seen as a risk 'the uncertainty principle'. There is a comfort within the organisation that the current approach will deliver something although more often than not it is not optimal.
4. Biggest is cultural, getting people to read, understand and agree how to use AWE. New processes are seen to introduce risk, as there will be a fear that deadlines will be impacted. 'IMHO I think there is value in this organisation trying the AWE Process'.
5. Organisational resistance to change. People and culture of this organisation may not lend itself well to this challenge. The business funding model is a problem, the business want to be predictive before they spend, even if the prediction is not accurate. Agile approaches such as AWE question the Buy over Build model. The organisation is driven by business generated deadlines.
6. Lack of vision. Inertia, we suffer from a why change attitude. There may not be a wealth of projects could exploit AWE.
7. People are too set in their ways. Command and Control management style. Needs high-level visionary buy-in. The organisation is too risk averse. Also there is a serious lack of investment in pilots.
8. Organisational structure is bureaucratic. There is a lack of empowerment for senior Technology leadership. There is a lack of senior Technology leadership vision. Organisational culture (banking), why should we learn something new? No real external demand for change. Technology does not listen. The organisation is not in a leading edge position. Lack of financial drivers. The business are not masters of technology and the business are not customers.

Question 11 – 'What did you think of the *AWE Pilot*[15] carried out by Andrew McDonald and Tracey Connor?'

1. Useful, highlights issues that are not addressed through the current development process.
2. Excellent. Clearly demonstrates what should be done to involve end-users to help the project team understand what the end-users' needs are and what is required to be tackled to solve problems.
3. Useful piece of work. The only way to expose important user feedback to the success of the project. Key factor is including people who are not familiar with project/IT in general.
4. Have not seen the results in detail, but a good thing to do. Ten out of ten for trying it out. Usability is an important issue.
5. A good piece of work. It was quick, cheap and derived a lot of benefit.
6. I don't know an awful lot about it!
7. Valuable piece of work for a small amount of resources. Highlighted a number of issues that used to be addressed and could easily be addressed during development. Highlighted and confirmed reasons for using the AWE Process.
8. Don't know a lot about it.

Question 12 – 'Why do you think this was not carried forward and formalised within our Development Process?'

1. The business would have been happy to adopt the AWE process on Web-based projects. Technology is re-active rather than pro-active when it comes to tacking new challenges.
2. Due to organisational leadership change, that has created a lot of uncertainty as to our direction going forward, thus embracing change would be difficult. It is recognised that there is value in the AWE process, but getting opportunity to try it out has been difficult.
3. Because it was out with the norm. Not given proper consideration by those in power.
4. Rightly or wrongly the AWE process is new and it would be a hassle to do things differently, people want an easy life so ignore change. Lack of money.
5. Our current development process is so large that to add an Evaluation Phase to the existing process would be seen to make it clumsy. In addition, no one in the organisation is focused on how to improve the process, no one is tasked with improving the process.
6. Not applicable, see previous answer.
7. Lack of clear ownership of existing processes. Lack of vast amount of Web-based projects in our current plan. Need a high-level evangelist in the projects. Organisation was not prepared to fully support projects like this one.
8. Was not management's idea. Both these people are just students.

Question 13 – 'What changes, if any, would you like to see to our development process?'

1. Clearly the agile route has its merits, I would like to see our process adopting the agile route. Allowing the organisation to focus on time-to-market pressures and relieving the bureaucratic nature of our current process. I find it shocking that we do not get value for money, it is better to go external. As an organisation we need to focus on particular channels, currently however there is no strategy for how we are going to service customers in the future.
2. Would like to adopt the AWE Process.

[15] AWE Pilot replaces the in-house company name for this project to ensure the anonymity agreement between the company and the authors.

3. Use the evaluation phase, involve end-user for testing. Customer surveys, post-implementation reviews are all that currently happens to get feedback on how users are performing with the solutions produced by this shop. We need to adopt a speedier approach. We should look across the board at all agile approaches. Our current development method is cumbersome and slow-to-market.
4. Our current process is too cumbersome. I find it confusing even after a year. It is poorly understood, I still need more training to understand our process.
5. I would like to see different processes for different types of development. We are currently using a one size fits all approach to project development, and this is not well suited. Essentially we are trying to apply a process, and techniques for mainframe/structured design IT projects, to object-based and Web-based projects.
6. I would like to see different approaches for different types of project challenge rather than a one size fits all. I think the agile approach has a lot of merits and would like to see the organisation adopting this route.
7. Introduction of more flexibility, different types of process approach for different types of IT challenge. Ownership with measurable outcomes for our process approaches.
8. Streamlining. More empowerment. Get rid of cowboys.

Question 14 – 'What do you perceive as the strengths of the organisation's process?'

1. Our current process forces a very risk averse approach (strength and weakness).
2. Clear documentation. Rigid, therefore novice and idiot proof in theory. Clearly understood by developers.
3. Formal methodology, well documented, does at least provide a framework. SPRAT is good for low cost, low resource type projects.
4. That they have a process at all. The organisation has invested time and money in their process. That it's still alive and living. It does deliver something.
5. Lends itself towards a financial viewpoint. Very structured, helps manage progression and does deliver something. It also calls out what documentation should be delivered.
6. Good at ensuring we have a robust solution. Good for well understood systems. Good where end-users are not heavily involved. Good for long-term document centric projects.
7. Not prescriptive in the techniques to be adopted. The downside is that the techniques need to be defined elsewhere. The way it is applied is poor.
8. Keeps people in line. Does provide a framework. Provides a standard, and an audit trail. It has sign off from everyone although probably too many stakeholders.

Question 15 – 'What do you perceive as the weaknesses of the organisation's process?'

1. Cost versus the benefit equation is not well balanced. Also the process is too focused on short term issues.
2. Cumbersome, repetitive, duplicates process. Encourages project development by book rather than by need. Very inflexible.
3. Poor project management. People do not follow process. In the past we often see deviation from process. Now we are starting to see more compliance, IMHO because we are forcing people to have document approval meetings.
4. It is poorly understood. It is cumbersome. Needs to be refined. Born from old technologies and paradigms. It is a one size fits all approach that is not designed for new technology challenges.
5. A lot of duplication of documentation. One size fits all approach is flawed.
6. Process is not suitable for all projects: particularly small projects; projects that require heavy end-user involvement; require iteration, or speed to market.

7. Too much emphasis on documentation. Too often the wrong documentation is produced. Testing is too long and the testing focus is not sound. Our process is not clearly understood by most people. There is no clear accountability and responsibility. IMHO the current in-house skills are not strong enough to adopt an Agile Approach. You would also need a mentor within the teams and in co-ordination team for AWE to be adopted successfully.
8. Everyone does not understand it. Expensive to maintain. It puts the company into silo solution mode all the time. Too bureaucratic, far too many meetings. Difficult to get consensus at meetings, often a lack of coherent agendas.

Question 16 – 'Describe the major phases involved in the organisation's development process.'[16]

[16] The answers to question 16 are not presented as they were considered too specific in terms of organisational terminology to present without compromising the anonymity agreement between the author and the company.

Appendix 8: Survey of an Intranet Development Team: Questions & Answers

The questions and answers presented in this appendix are those used in the survey of the Identity Management Intranet stream, carried out in May 2004. The questions and recorded answers are listed below for the eight interviewees, referred to as hereafter as interviewees 1-8:

Question 1 – 'What was your role on the Identity Management team?'

1. Business Project Manager.
2. Business Analyst.
3. Technology Consultant.
4. Technical Lead.
5. Project Stream Coordination Role.
6. Engineering Room Manager.
7. Technology Sponsor.
8. Project Release Manager.

Question 2 – 'What were your responsibilities within the Identity Management team?'

1. Responsible for Identity Management Intranet project requirements and deliverables. To make sure that the business requirements were signed off and agreed with the various business areas impacted by the Intranet project. To ensure that the solution delivered met the business requirements. To ensure that the Intranet requirements were fulfilled.
2. Responsible for enterprise user provisioning product selection and role based access and control from a business perspective. Involved in the development of the requirements right through to implementation and the creation of the user administration processes.
3. No formal responsibilities defined. Operated primarily in a design role. It is very difficult to talk about roles on new technology projects in this group. Have to define your own role.
4. Firstly to understand IM concepts, develop and deploy solution. Train new technical members. Involved in communicating the IM solution to other streams.
5. Contact point for other Intranet Streams that relied on Identity Management.
6. Accountable for the delivery of the technical solution.
7. Direction setting. Global engagement. Establishment of the strategy.
8. To ensure that the business objectives were achieved within the project timescale.

Question 3 – 'How long have you been involved in the development of Web applications?'

1. Three and a half years.
2. Four and a half years.
3. Eighteen months within this group. Eight years in total.
4. Eight years.
5. 18 months.
6. Four years.
7. Five years.
8. Three years.

Question 4 – 'In your experience do you notice any major differences between Web-based development and Tradition IT Projects?'

1. Usability is of greater importance in Web-based projects. Also there is a need for greater collaboration between business and technology stakeholders. Technology need to educate the business during the creation of the requirements and throughout other phases of the design process.
2. Not really. The business elements are similar for my role.
3. Not as much as I thought. In this group a technology project is a technology project. Increased visibility of end-user importance. Greater interaction and a bigger stakeholder management issue, complexity is increased.
4. There is a difference in the way they are run. Often though this is down to people's mind sets. In Web-based projects, project managers seem to be keener to get business involvement throughout and to use a more iterative and incremental process approach.
5. The development path on traditional projects can be more precisely defined. One is less likely to encounter technical issues at advanced stages of the project in traditional projects. In Web-based development it is harder to predict what, with so many different streams what problems you are going to encounter, and one needs to try things first.
6. Yes. Web-based development requires a completely different skill set. The development process is less structured and more iterative. There are a lot more security implications to address. The end-user population is larger and more diverse.
7. Not in this organisation. As a company we don't treat Web-based development as different. Web-based development is constrained in this organisation by methodologies best suited to mainframe development. It is not enough to use labels e.g. J2EE, Java, etc. Need an engrained cultural shift.
8. The methodologies for project management are the same. However the major differences were estimating the duration of tasks as the technology was new to staff and vendors alike.

Question 5 – 'What do you think of the organisation's development process and its applicability to Web application development?'

1. Lacking. The CEO vision is easy to understand in a power point presentation, however, it is extremely difficult to achieve in practice. For example, achieving one helpdesk for the whole of the Intranet project proved impossible. Too many barriers presented by culture and organisational structure. The executive role needs to operate differently.
2. Not that familiar with the in-house process. The organisation has tried to maintain a common approach to all projects. Not aware of any changes from a business perspective, there is not enough thought about how the technology impacts the business and end-users. It needs to be modified to be more adaptable for Web-based technology. Too much of a one-size fits all nature.
3. As an organisation we are naive. We do not invest in the infrastructure required to maximise the potential from Web-based development.
4. The in-house process is not a good fit for Web development. The in-house process is based on the Waterfall model and depends on being able to predict a lot upfront. It is also fairly inflexible. Also it does not cope well with many different streams working in parallel like the Intranet project, where there are many overlapping issues between different streams.
5. I think that the in-house process is lacking behind the technology we are using to develop solutions. Testing is a particular problem as it still takes a traditional project

development view. This is problematic because legacy procedures are being imposed in a rapid development environment.
6. The in-house process is not applicable. The current version is not applicable to Web engineering as it is waterfall based. Although the organisation is trying to change this.
7. I don't believe we have as yet created a suitable environment for Web engineering. This is evidenced in our inability to deliver time-to-market solutions. Other Web focused companies can change in weeks rather than years. If we were Web agile we would be more dominant in the market place.
8. The in-house process does not allow for prototyping, which is essential when developing applications with 'newness'.

Question 6 – 'Have you read the AWE Process Technical Report? YES / NO'
1. Yes about two years ago.
2. No.
3. I have read most of it.
4. No.
5. No.
6. No.
7. Yes.
8. No.

Question 7 – 'What do you perceive as the strengths of the AWE Process?'
1. Iterative nature, helps maximise technology and business knowledge to the benefit of the end-user.
2. Not applicable, see answer to question 6.
3. Collocation. AWE allows the development team to do it. Gives the development team a mandate to go and make it happen.
4. Not applicable, see answer to question 6.
5. Not applicable, see answer to question 6.
6. Not applicable, see answer to question 6.
7. It is a methodology that any medium financial service sector organisation should feel happy embracing. The mistake made in this organisation is that we treat it as a technology methodology rather than a business methodology, as it tries to improve productivity and provides the 'power of delivery'.
8. Not applicable, see answer to question 6.

Question 8 – 'What do you perceive as the weaknesses of the AWE Process?'
1. In this organisation there are serious organisational and cultural barriers. The finance model dictates the development approach and causes a cultural of protectionism. The current approach hinders technology and business working together especially at the initiation stage.
2. Not applicable, see answer to question 6.
3. Not fit for all. Some people need to be hand held or are too silo-focused in their work. AWE needs people who are willing to collaborate.
4. Not applicable, see answer to question 6.
5. Not applicable, see answer to question 6.
6. Not applicable, see answer to question 6.
7. Only weakness is that we have not used it fully. No weaknesses as I have not seen it used fully in anger. There are many cultural barriers that would need to be overcome in order to fully adopt AWE in this organisation.
8. Not applicable, see answer to question 6.

Question 9 – 'What were your opinions regarding collocating the business and technology development team?'

1. It makes a massive improvement. Much easier to communicate face-to-face than through the telephone or email. Telephone and email encourage ambiguity and a defensive nature in stakeholders.
2. Definitely a good thing provides for easier communication.
3. I don't think we could have done what we did in such a short period of time without collocating the team.
4. It was good for communication and feedback. It guards against mistakes being made in the early stages, such as requirements and high-level design.
5. I encountered problems due to the fact that I was not 100% devoted to the Identity Management stream at the beginning. But if I had been on nothing but Identity Management 100% of the time at the beginning I would say that it is quite suitable, in fact very necessary.
6. It is crucial. We used to do it 15 years ago. However, you need smart business people who have authority to make decisions and will make decisions.
7. Essential. Technology is an enabler as part of a solution.
8. Essential.

Question 10 – 'What problems did you experience/perceive with collocating the development team?'

1. There are a number of practicalities that present themselves if it is to be carried out over an extended period of time e.g. travel, personal circumstances (peoples personal lives not making it easy for them to travel i.e. young children) etc. Difficult to scale to large teams.
2. I had to travel to collocate with the team.
3. Logistics. It is hard to travel if you have a young family. Also there are times when one wants to work separately as well.
4. Can't think of any.
5. On-going work on other projects.
6. Need to have one team leader. Didn't have smart enough business customers.
7. The culture is not there. Stakeholders worry more about personal issues rather than deliverables. Camps emerge. However, these can be overcome by strong cohesive leadership.
8. Travelling. People have to stay away from home.

Question 11 – 'What benefits did you experience/perceive with collocating the development team?'

1. Cuts down ambiguity, creates team spirit, feel that you work more efficiently.
2. Face-to-face contact with all team members helped with my understanding. Encouraged a greater sense of team spirit.
3. Decision making process is quicker. Easier to communicate compared with telephone and email. Prevents misinterpretation often found when communicating through email and telephone.
4. Communication was easier. It was easier to check quality and quicker to achieve it.
5. With rapid development, decisions need to be made faster, having people collocated achieved this. Having the business there was very important because we needed to focus heavily on the end-user experience and the business could answer for them immediately.
6. Speed. The business can see and get what they want. Communication, collocation breaks down the 'them and us' culture.

7. IM did not really accrue the benefits, hindered by multi-location, multicultural project. The utopia would be to give one team direction and guide them to doing it.
8. Improved speed in decision making; better co-ordination of activities; both business and technology understood each other's issues and difficulties.

Question 12 – 'How appropriate was the emphasis on business process re-engineering from the beginning of the IM stream?'

1. It was appropriate, yes, however, the IM stream picked it up too late, even though we were the first stream to address it in the Intranet project. Communicating it was difficult.
2. Some of the user provisioning processes were new rather than re-engineering existing user provisioning processes. Although a lot were evolved.
3. It was recognised by all parties, although there was not enough effort put into re-engineering, this was primarily due to a lack of executive level sponsorship in the business.
4. Paramount, because the scope of IM in terms of funding management and promotion needs to be beyond one projects boundaries and enterprise wide in focus.
5. Business process re-engineering is critical. There was not enough emphasis on this.
6. The emphasis from the business was not strong enough as they did not understand what it meant
7. Difference between importance and appropriateness. Reason for failure is that we did not use business process re-engineering methodology. However, I have subsequently addressed this on other projects. Technology alone cannot solve problems that have not been defined.
8. Very poor. Took quite a while for the business to understand that the business process re-engineering activities were just as important as the technology used.

Question 13 – 'Did the development process hinder or assist with business process re-engineering activities?'

1. The Intranet project business process hindered it, the technology stakeholders were saying the right things, however the business process was too predictive and not adaptive enough. Lack of senior stakeholder buy-in in the business. The business stakeholder got too caught up in the technology.
2. I felt that the Intranet process had no explicit focus on business process re-engineering. Too technology focused.
3. Did not hinder, not sure that it assisted either. Lack of trust, the organisation was not prepared to change the status quo.
4. Hindered. The development process is project oriented, concerned only with meeting project deliverables. Lacks an enterprise model.
5. The development process did not affect it either way.
6. It hindered it as it was too structured. It needed a more iterative process approach. Not enough prototyping 'try and change'.
7. The in-house process has not been suitably reviewed to introduce new technologies.
8. I don't think that there is a real business development process for projects. Clearly a gap!

Question 14 – 'What did you consider successful regarding the Identity Management project?'

1. New technology was implemented and works, because the business and technology worked together as one team. IM was delivery focused.
2. It has gone in and is being used successfully in the project time-scales. It has forced greater thought surrounding Identity Management in general.

3. It worked. Brought together a new team that gelled and grew together. Need to do this more often in the group.
4. The fact that we managed to deliver it given its scope.
5. It brought in new technology very swiftly. I thought that it was a good foundation for enterprise-wide business process re-engineering. If Identity Management is followed through it will be of great benefit to the organisation.
6. It was delivered on time; it was sold and accepted by other Intranet technical streams, which was the hardest to achieve; it brought about a cultural change in technology, and it was fun.
7. The global team.
8. Implemented on time and within budget; met business requirements; introduced new technologies and processes into the organisation; improved global relationships within the company; Winners of the global innovation award.

Question 15 – 'What did you consider unsuccessful regarding the Identity Management project?'

1. Time-to-market pressures conflict with being comprehensive.
2. Unsuccessful in that User provisioning has not gone in quicker. Cost and communication were a problem throughout Intranet project.
3. Cost and effort estimation is difficult. Vendor support time was underestimated, however, this forced the group's developers to pick up more of the issues. Marketing of the IM concept is difficult.
4. Not getting user provisioning automated early enough. Also we could have spent more time focusing on ongoing operational management.
5. Need to be sold more to the business. We are still too closely associated with the Intranet project.
6. The lack of perception and interest by senior management in what was achieved. An ill-formed delaying decision in the most important part of the solution.
7. Despite the evident quality of the deliverable it did not receive sufficient business buy-in. Now being addressed by the European business sponsor.
8. Failure to implement an automated user provisioning solution during phase one. Failure of vendors to respond quickly, they did not have enough strength and depth.

Question 16 – 'What influence did you consider the AWE Process having had on Identity Management being successful or unsuccessful?'

1. Collaboration between the business and technology stakeholders diminished after the selection of the automated user provisioning product when the Intranet process was adopted. This was to the detriment of the project as the end-users were not the primary focus. Business and technology began to run in different directions.
2. Close working of business and technology was significant. As the project has matured, this close working has fallen away. Areas in the technology and business have matured and the development is coming to an end, now business as usual.
3. Collocating the team was beneficial. Added focus to the task.
4. Improved communication fostered team spirit between business and technology and helped move things along quicker.
5. Can't think of any one particular feature because I have not read the technical report, but think it had a positive effect on success.
6. Helped build the team at the start. Helped define requirements as we understood them.
7. The fact that the author created the identity management culture. While we did not employ AWE verbatim we adopted many of its principles in Identity Management. We employed the author not the book.
8. Not applicable.

Question 17 – 'What other factors contributed to Identity Management being successful or unsuccessful?'

1. Strong technology leadership. Strong individuals and gifted technical team.
2. Enthusiasm regarding the new technology helped. Desire from the technology stakeholders to be the global leaders in the Identity Management arena. Desire to give the end-user the best possible experience.
3. Senior technology sponsor.
4. I think the enthusiasm and dedication of the team members. Also bringing in the vendors as part of the team.
5. Availability of skilled staff. Good Web technology experience in the team. Directing and assigning the right staff to the stream.
6. High quality individuals. Some of the best I have worked with.
7. Determination of the technology team and the technology stakeholders' wider business experience.
8. Quality and focus of the team including the individuals supplied by the vendors.

Question 18 – 'Were you aware of the AWE Pilot run on the groups Internet Banking channels?'

1. No not really.
2. No.
3. Not really.
4. No.
5. No.
6. Yes.
7. No.
8. No.

Question 19 – 'Why do you think that the Intranet project, of which Identity Management is a stream, would not adopt the successful end-user approach piloted for Internet Banking?'

1. Three reasons: Never bought-into by senior stakeholders; seen to conflict with time-to-market pressures; and seen to have an impact on cost and time.
2. Not applicable, see response to question 18.
3. A lot to do with not being able to challenge the status quo. The business team was under technology. Lack of business leadership. A view that we could just replicate the 3270 legacy system. Lack of faith in end-users.
4. Not applicable, see response to question 18.
5. Surprised that we didn't. Clearly we did not pay enough attention to these issues. I suppose it is like a car manufacturer offering the customer the ability to custom design the dashboard of their car. Can't understand why they did not adopt it. There needs to be an ergonomics testing phase in the development process.
6. Leadership resistance to many iterations due to tight timescales. Restricted by company policies.
7. Not applicable, see response to question 18.
8. Not applicable, see response to question 18.

Glossary

.NET	.NET is an enterprise framework based around the Microsoft technology suite.
Agile Process	A term applied to a number of lightweight methodologies that have emerged since the mid 1990s. These methodologies value: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation, and responding to change over following a plan. There are a number of Agile Processes, including: Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development and AWE.
Browser Experience	A technique that AWE developers should use to collaborate and focus their development efforts around the Web interface, using collaborative sessions to discuss and review the end-user browser experience.
Business Expert	A stakeholder responsible for business issues in a development team using the AWE Process.
Business Model	The Business Model reflects issues associated with business objectives during Web or software engineering endeavours.
Client	The Client is a stakeholder that represents the body or bodies who are paying for the project development.
Co-ordination Team	A co-ordination team is responsible for guiding the inter-team communication process and for leading the inter-team initiatives that will enhance the objectives of the larger Web presence.
Contractor Organisation	An organisation that tenders and services contracts for Web application development, which have been put out to tender by Outsourcing Organisations.
Creative Design Model	The Creative Design Model reflects issues associated with the aesthetic aspects of the user interface on a Web or software engineering project.
Creative Designer	A stakeholder responsible for aesthetic issues in a development team using the AWE Process.
Customer Community View	Those individuals who interact directly with the live software system (the end-users) or are affected by the live operation of the software system.

Domain Expert	A stakeholder responsible for issues associated with the application domain to which Web applications are being applied. For example bank branches in the Web-based Teller System.
Domain Model	The Domain Model reflects issues associated with the domain to which the objectives reflected in the business models and the software model are to be applied during Web or software engineering projects.
End-Users	Stakeholders representing the users of a proposed system in a development project using the AWE Process.
Enterprise Framework	An Enterprise Framework is a series of technologies, standards and processes, with supporting software infrastructure to help organizations develop solutions for the enterprise.
Extranet	An extension of an organisation's Intranet out onto the Internet, enabling selected authorised individuals both within and out with the organisation to access the company's private data and applications using the World-Wide Web.
Formal Communication	Communication conducted through paper or electronic documents.
Heavyweight Process	A term used to describe Monumental Processes. Derived from the perceived heavy volume of documented deliverables produced by projects using Monumental Processes and their rather heavyweight management style.
Informal Communication	Communication conducted through face-to-face interaction.
In-house Organisation	An organisation that develops Web applications using developers from within its own organisation.
In-Sourcing	A technique used by Outsourcing Organisations to ensure In-house development expertise on projects outsourced to Contracting Organisations. In-Sourcing works by steadily replacing contractors with outsourcing organization employees during the project development life-cycle.
Inter-Team Communication	Communication, whether formal or informal, between different development teams on the same Web presence or Web development project.
Internet	An interconnected system of networks that links computers across the globe using the TCP/IP protocol. The World Wide Web runs on top of the Internet.

Intra-Team Communication	Communication, whether formal or informal, between developers in the same development team.
Intranet	A privately maintained computer network based on the Internet, with access to restricted to authorised individuals. Many companies use World Wide Web technologies to run a number of Web-based applications within their Intranet.
Java™ 2 Enterprise Edition (J2EE)	J2EE is an enterprise framework based around the Java programming language.
Lightweight Process	A term often used to describe Agile Processes. Derived from the lightweight nature of the documented project deliverables produced from projects using Agile Processes.
Orthogonal Uni-discipline Developer Communication	Communication, whether formal or informal, between similar types of developer in different development teams on the same Web presence or Web development project.
Outsourcer	An organisation that uses Contracting Organisations to help develop their Web applications.
Plan-driven Process	Traditional software engineering processes that are heavily influenced by the engineering disciplines. Monumental Processes include the Rational Unified Process and Structured Systems Analysis and Design Method.
Predictive Process	Predictive Processes are deterministic in nature, identifying very early in the development life-cycle the problem and proposed solution.
Software Engineer	A stakeholder responsible for technical issues in a development team using the AWE Process.
Software Model	The software model reflects the technical issues associated with the development of a software or Web application.
Team Leader	A stakeholder responsible for team guidance and project management issues in a development team using the AWE Process.
The Agile Web Engineering (AWE) Process	An Agile Process for the development of Web-based applications.
World Wide Web (WWW)	The collective information residing on all Internet servers that use the HTTP protocol, accessible to users via a simple point-and-click Web browser system.

References

Abrahamsson et al. 2002	Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002), 'Agile Software Development Methods: Review and Analysis', *VTT Publications 478* <http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf>
Abrahamsson et al. 2003	Abrahamsson, P., Warsta, J., Siponen M. T. & Ronkainen, J. (3-10 May 2003), 'New Directions on Agile Methods: A Comparative Analysis', *25th International Conference on Software Engineering, Portland, Oregon* pp. 244-254, ISBN: 076951877X.
Adobe 2004	Adobe Systems Incorporated (07 April 2004), 'Adobe Photoshop', *Adobe Systems Incorporated*, <http://www.adobe.com/products/photoshop/main.html>
AGENCY.COM 2004	AGENCY.COM Ltd. (16 February 2004), 'AGENCY.COM: People - Web Page', *AGENCY.COM Ltd.*, <http://www.agency.com/about_us/people.asp>
Apple 2004a	Apple Computer, Inc. (02 April 2004) 'Apple – Quicktime', *Apple Computer, Inc.*, <http://www.apple.com/quicktime/>
Apple 2004b	Apple Computer, Inc. (02 April 2004) 'Apple – Quicktime VR', *Apple Computer, Inc.*, <http://www.apple.com/quicktime/products/qt/overview/qtvr.html>
Baetjer 1997	Baetjer, H. (Oct 1997), 'Software as Capital: Economic Perspective on Software Engineering', *IEEE Computer Society Press*, ISBN: 0818677791.
Barry & Lang 2001	Barry, C. & Lang, M. (April-June 2001) 'A Survey of Multimedia and Web Development Techniques and Methodology Usage', *IEEE MultiMedia, vol. 8, no. 2*, pp. 52-61.
Baskerville et al. 2003	Baskerville, R., Balasubramaniam, R., Levine, L., Pries-Heje, J. & Slaughter, S. (November-December 2003), 'Is Internet-Speed Software Development Different?', *IEEE Software*, vol: 20 no.6, pp. 70-77.
Beck 2000	Beck, K. (December 2000), 'Extreme Programming Explained: Embrace Change', *Addison- Wesley*, ISBN: 0201616416.
Beck et al. 2001	Beck, K. et al. (11-13 February 2001), 'Manifesto for Agile Software Development', *The Agile Manifesto Web Site*, <http://www.agilemanifesto.org/>
Benington 1956	Benington, H.D. (28-29 June 1956), 'Production of Large Computer Programs', *Proceedings Symposium on Advanced Programming Methods for Digital Computers*, Republished in Annals of the History of

	Computing, Oct. 1983, pp. 350-361.
Berry & Naumann 2000	Berry, V. & Naumann, A. (09 August 2000), 'RAD – A Whatis Definition', *TechTarget*, <http://whatis.techtarget.com/definition/0,,sid9_gci214246,00.html>
Boehm 1981	Boehm, B.W. (1981), 'Software Engineering Economics', *Prentice Hall, Inc.* ISBN: 0138221227.
Boehm 1988	Boehm, B.W. (1988), 'A Spiral Model of Software Development and Enhancement', *IEEE Computer*, vol. 21, no. 5, pp. 61-72.
Boehm & Turner 2003a	Boehm, B. & Turner, R. (2003), 'Balancing Agility and Discipline: A Guide for the Perplexed', *Addison-Wesley*, ISBN: 03211861265.
Boehm & Turner 2003b	Boehm, B. & Turner, R. (2003), 'Rebalancing Your Organisation's Agility and Discipline', XP/Agile Universe 2003, LNCS 2753, pp. 1-8, ISBN: 354040662X.
Burdman 1999	Burdman, J. (1999), 'Collaborative Web Development: Strategies and Best Practices for Web Teams', *Addison-Wesley*, ISBN: 0201433311.
Calvert, Smith & Natis 2001	Calvert, M., Smith, D. & Natis, Y. (29 August 2001), 'Management Update: Don't Waste Money on Application Server Overcapacity', *Gartner, Inc.* Report Code: IGG-08292001-04.
Cockburn 2000	Cockburn, A. (June 2000), 'Characterizing People as Non-Linear, First-Order Components in Software Development', *4th International Multi-Conference on Systems, Cybernetics and Informatics*, Orlando, Florida.
Cockburn 2002	Cockburn, A. (Feb 2002), 'Agile Software Development', *Pearson Education, Inc.* ISBN: 0201699699.
Coda et al. 1998	Coda, F., Ghezzi, C., Vigna, G. & Garzotto, F. (April 1998), 'Towards a Software Engineering Approach to Web Site Development', in Proceedings of the 9th International Workshop on Software Specification and Design, 20th International Conference on Software Engineering, Kyoto (Japan), pp. 8-17.
Conallen 1999	Conallen, J. (1999), 'Building Web Applications with UML', *Addison-Wesley*, ISBN: 0201615770.
Conradi, Fernström & Fuggetta 1994	Conradi, R., Fernström, C. & Fuggetta, A. (1994), 'Concepts for Evolving Software Process', Software Process Modelling and Technology, ed. Finkelstein, A., Kramer, J. & Nuseibeh, B. *Research Studies Press Ltd.* pp. 9-31, ISBN: 0863801692.
Constantine	Constantine, L. L. (19 April 2001), 'Lightweights, Heavyweights and Usable Processes for Usable Software', *Keynote at Software*

2001	*Development 2001, San Jose.*
Constantine & Lockwood 1999	Constantine, L. & Lockwood, L. (1999), 'Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design', *Addison-Wesley*, ISBN: 0201924781.
Dart 2000	Dart S., (2000), 'Configuration Management: The Missing Link in Web Engineering', Artech House Publishers, ISBN: 1580530982
Datamonitor 2002	Datamonitor (April 2002), 'The Business Case for User Provisioning: ROI Model for eTrust Admin', *Datamonitor*, <http://www.cyberklix.com/mailrimgs/datamonitor.pdf>
Deshpande & Hansen 2001	Deshpande, Y. & Hansen, S. (April-June 2001), 'Web Engineering: Creating a Discipline among Disciplines', *IEEE Multimedia*, pp. 82-87.
Deshpande, Murugesan & Hansen 2001	Deshpande Y., Murugesan S. and Hansen S. (2001), 'Web Engineering: Beyond CS, IS and SE Evolutionary and Non-engineering Perspectives', *Web Engineering, LNCS 2016*, pp. 14-23.
Devine & Welland 2000	Devine, J. & Welland, R. (2000), 'Cultural Computing: Exploiting Interactive Digital Media', *Invited paper in Museums International (UNESCO, Paris)*, No. 205, vol. 52, no. 1, pp. 32-35, ISSN: 1350-0775
Devine et al. 2004	Devine, et al. (02 April 2004), 'Welcome to the Hunterian Museum and Art Gallery', University of Glasgow, <http://www.hunterian.gla.ac.uk/index.html>
DSDM 2004a	Dynamic Systems Development Method (DSDM) Limited (23 March 2004), 'DSDM Homepage', *Dynamic Systems Development Method Ltd,* <http://www.dsdm.org/>
DSDM 2004b	Dynamic Systems Development Method (DSDM) Limited (23 March 2004), 'DSDM Consortium – Full Members UK/International Consortium', *Dynamic Systems Development Method Ltd.*, <http://www.dsdm.org/en/membership/full_members.asp>
DSDM 2004c	Dynamic Systems Development Method (DSDM) Limited (11 April 2004), 'Agile Development: What is IT?', *Dynamic Systems Development Method Ltd*, <http://www.dsdm.org/kss/details.asp?fileid=344>
DSDM 2004d	Dynamic Systems Development Method Ltd. (23 March 2004), 'DSDM Tour: Roles', *Dynamic Systems Development Method Ltd.*, <http://www.dsdm.org/tour/roles.asp>
eBucks.com 2004	eBucks.com (26 Feb 2004), 'eBucks.com Homepage', *eBucks.com Ltd.* <https://www.ebucks.com/>

Ericsson & Simon 1984	Ericsson, K. A. & Simon, H. A., (1984), 'Protocol Analysis: Verbal Reports as Data', Cambridge, MA: MIT Press.
Fichman & Kemerer 1997	Fichman, R. G. & Kemerer C. F., (1997), 'Object Technology and Reuse: Lessons from Early Adopters', *IEEE Computer*, vol. 30, no. 10, pp. 47-59.
Fortune 2002	Time Inc. (July 2002), 'Fortune.com – Home', *Time Inc.*<http://www.fortune.com/fortune/>
Fortune 2003	Time Inc. (July 2003), 'Fortune.com – Home', *Time Inc.*<http://www.fortune.com/fortune/>
Fowler 2001	Fowler M. (March 2001), 'The New Methodology', *MartinFowler.com*, <http://www.martinfowler.com/articles/newMethodology.html>
Gaedke et al. 1999	Gaedke, M., Gellersen, H-W., Schmidt, A., Stegemüller, U. & Kurr, W. (1999), 'Object-oriented Web Engineering for Large-scale Web Service Management', *Proceedings of the 32nd Hawaii International Conference on System Sciences*, pp. 1-9.
Gaedke & Gräf 2000	Gaedke, M. & Gräf, G. (2000), 'Development and Evolution of Web Applications Using the WebComposition Process Model', *WebEngineering 2000, ed. Murugesan S. and Deshpande Y., Springer-Verlag, LNCS 2016*, pp. 58-76, 2001.
Glass 2002	Glass, R. L. (2002), 'Facts and Fallacies of Software Engineering', Addison Wesley Professional, ISBN: 0321117425.
Haire, Henderson-Sellers & Lowe 2001	Haire, B., Henderson-Sellers, B. & Lowe, D. (2001) 'Supporting Web Development in the OPEN Process: Additional Tasks', *Proceedings COMPSAC'2001: International Computer Software and Applications Conference, Chicago, Illinois, USA.*
Hansen, Deshpande & Murugesan 1999a	Hansen, S., Deshpande, S.Y. & Murugesan, S. (16-22 May 1999), 'A Skills Hierarchy for Web Information System Development', *Proceedings of the First ICSE Workshop on Web Engineering.*
Hansen, Deshpande & Murugesan 1999b	Hansen, S., Deshpande, S.Y. & Murugesan, S. (16-22 May 1999), 'Web Engineering: Beyond CS, IS and SE-An Evolutionary and Non-Engineering View', *Proceedings of the First ICSE Workshop on Web Engineering.*
Hammer & Champy 2001	Hammer, M. & Champy, J. (23 August 2001), 'Reengineering the Corporation', *Nicholas Brealey Publishing Ltd, Revised Ed.*, ISBN: 1857880978.
Hardy,	Hardy, C. J., Thompson, J. B. & Edwards, H. M., (1995), 'The use,

Thompson & Edwards 1995	limitations and customisation of structured systems development methods in the United Kingdom', Journal of Information and Software Technology, Volume 37, Issue 9, pp. 467-477.
Henderson 2002	Henderson, P. (Ed.), (March 2002), 'Systems Engineering for Business Process Change: New Directions, Collected Papers from the EPSRC Research Programme', Springer-Verlag UK, ISBN: 1852333995.
Henderson-Sellers 2004	Henderson-Sellers, B. (Feb 2004), 'The OPEN Website', *OPEN Consortium*, <http://www.open.org.au/>
Humphrey 1999	Humphrey W. (September 1999), 'Introduction to the Team Software Process', *Addison Wesley*, ISBN: 020147719X.
IBM 2004	IBM Corporation (10 April 2004), 'IBM United Kingdom', *IBM Corporation* <http://www.ibm.com/uk/>
ICWE 2002	Olsina, L. & Oscar P., (9-13 September 2002), 'Ibero American Conference on Web Engineering', Santa Fe, Argentina, <http://www.ing.unlpam.edu.ar/icwe2002/>
ICWE 2003	Cueva Lovelle, J. M. et al. (Eds.), (July 2003), 'Web Engineering: Proceedings of the 3rd International Conference', Springer Verlag, LNCS 2722, ISBN: 3540405224.
ICWE 2004	Koch, N., Wirsing, M. & Fraternali, P., (Eds.), (July 2004) 'Web Engineering: Proceedings of the 4th International Conference', Springer Verlag, LNCS 3140, ISBN: 3540225110.
INTERSHOP Communications 2000	INTERSHOP Communications, 'Electronic Commerce Site Deployment, Based on INTERSHOP Enfinity™, Sample Project Plan and Costs', *INTERSHOP Communications*, 2000, <http://www.intershop.com/>
ISO 2004	International Organisation for Standards (12 March 2004), 'ISO 9000 index', International Organisation for Standards, <http://www.iso.ch/iso/en/iso9000-14000/iso9000/iso9000index.html>
Jacobson, Booch & Rumbaugh 1999	Jacobson, I., Booch, G. & Rumbaugh, J. (January 1999), 'The Unified Software Development Process', *Addison-Wesley*, ISBN: 0201571692.
Jacobson et al. 1992	Jacobson, I., Christenson, M., Jonsson, P. & Övergard, G. (1992), 'Object-Oriented Software Engineering: A Use Case Driven Approach', *Addison-Wesley*, ISBN: 0201544350.
Jalote 2000	Jalote P. (2000), 'CMM in Practice: Processes for Executing Software Projects at Infosys', Addison-Wesley, ISBN: 0201616262.

JUnit 2004	JUnit (22 March 2004), 'JUnit, Testing Resources for Extreme Programming', *Object Mentor, Inc.*, <http://www.junit.org/index.htm>
JWE 2004	Rinton Press, (31 August 2004), 'Journal of Web Engineering', Rinton Press, Princeton, New Jersey. ISSN: 15409589. <http://www.rintonpress.com/journals/jwe/>
Kirkpatrick 2004	Kirkpatrick, D. (26 January 2004), 'Why "Bottom Up" is on Its Way Up', *Fortune Europe Edition*, vol. 149, no. 1, p. 30.
Koch 1999	Koch, N. (November 1999), 'A Comparative Study of Methods for Hypermedia Development', *Technical Report 9905, Ludwig-Maximilians-Universität München.*
Kruchten 1995	Kruchten P. (Nov 1995), 'The 4+1 View of Software Architecture', *IEEE Software*, vol. 12 no. 6, pp. 45-50.
Kruchten 2003	Kruchten, P. (December 2003), 'The Rational Unified Process: An Introduction', *Addison-Wesley*, 3rd edition, ISBN: 0321197704.
Lewis & Rieman 1994	Lewis, C. & Rieman, J. (1994) 'Task-Centered User Interface Design', *Available via anonymous ftp from* <ftp.cs.colorado.edu>
Lindvall et al. 2002	Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Forrest, S., Tesoriero, R., Williams, L. & Zelkowitz, M. (September 2002), 'Empirical Findings in Agile Methods', *XP/Agile Universe 2002, LNCS 2418*, pp. 197-207. ISBN: 3540440240
Lowe & Eklund 2002	Lowe, D. & Eklund J. (7-11 May 2002), 'Client Needs and the Design Process in Web Projects', The Eleventh International World Wide Web Conference WWW2002, Honolulu, Hawaii, <http://www2002.org/CDROM/alternate/678/>
Macromedia 2004	Macromedia, Inc. (02 April 2004), 'Macromedia – Flash MX 2004', Macromedia, Inc. <http://www.macromedia.com/software/flash/>
MacLean et al. 1991	MacLean, A., Young, R., Bellotti, V. & Moran, T. (25-29 November 1991), 'Design Space Analysis: Bridging from Theory to Practice via Design Rationale', Proceedings of Esprit '91, Brussels, Belgium, pp. 720-730.
McDonald et al. 1996	McDonald, A., Ross, B., McDaid, S., McLean, A. & Cassidy, C. (10 June 1996), 'The University of Glasgow Hunterian Map', 3rd Year Computing Science Team Project, Department of Computing Science, University of Glasgow.
McDonald & Welland 2001a	McDonald, A. & Welland, R. (01 March 2001), *A Survey of Web Engineering in Practice*, Department of Computing Science Technical

	Report TR-2001-79, University of Glasgow, Scotland.
McDonald & Welland 2001b	McDonald, A. & Welland, R. (01 May 2001), 'Web Engineering in Practice', Proceedings of the Fourth WWW10 Workshop on Web Engineering, pp. 21-30.
McDonald & Welland 2001c	McDonald, A. & Welland, R. (2 December 2001), 'Agile Web Engineering (AWE) Process', Department of Computing Science Technical Report TR-2001-98, University of Glasgow, Scotland.
McDonald & Welland 2002	McDonald, A. & Welland, R. (26 July 2002), 'Supporting Evolution in a Web Engineering Process', Electronic Imaging and the Visual Arts (EVA) 2002, London.
McDonald & Welland 2003	McDonald, A. & Welland, R. (July 2003), 'Agile Web Engineering (AWE) Process: Multidisciplinary Stakeholders and Team Communication', Cueva Lovelle, J. M. et al. (Eds.): Third International Conference on Web Engineering, ICWE 2003, LNCS 2722, pp. 515-518. ISBN: 3-540-40522-4.
McDonald & Welland 2004a	McDonald, A. & Welland, R. (July 2004), 'Evaluation of Commercial Web Engineering Processes', *International Conference on Web Engineering (ICWE 2004)*, Munich, Germany.
McDonald & Welland 2004b	McDonald, A. & Welland, R. (25 Mar 2004), 'Agile Web Engineering (AWE) Process: Perceptions within a Fortune 500 Financial Services Company', accepted for The Journal of Web Engineering.
Mendes 2000	Mendes, E. (28-30 April 2000), 'Investigating Metrics for a Development Effort Prediction Model of Web Applications', *Proceedings of the 2000 Australian Software Engineering Conference*, pp. 31-41.
Mendes & Counsell 2000	Mendes, E. & Counsell, E. (28-30 April 2000), 'Web Development Effort Estimation using Analogy', *Proceedings of the 2000 Australian Software Engineering Conference*, pp. 203-212.
META Group 2001	META Group, Inc. (2001), 'META Group, Inc. – Return on Intelligence', META Group, Inc. <http://www.metagroup.com/>
Microsoft 2001a	Microsoft Corporation (24 September 2001), 'A Guide to Reviewing the Microsoft .NET Framework: A Platform for Rapidly Building and Deploying XML Web Services and Applications to Solve Today's Business Challenges', *Microsoft Corporation*, White Paper, <http://www.microsoft.com/net/>
Microsoft 2001b	Microsoft Corporation (23 September 2001), '.NET Homepage', *Microsoft Corporation*, <http://www.microsoft.com/net/>

Microsoft 2004a	Microsoft Corporation (02 Feb 2004), 'Windows XP Home Page', Microsoft Corporation, <http://www.microsoft.com/windowsxp/default.asp>
Microsoft 2004b	Microsoft Corporation (07 April 2004), 'Microsoft Office Assistance: Microsoft Office Outlook 2003', *Microsoft Corporation*, <http://office.microsoft.com/assistance/topcategory.aspx?TopLevelCat=CH79001807&CTT=6&Origin=ES790020011033>
Microsoft 2004c	Microsoft Corporation (16 February 2004), 'Microsoft Solutions Framework (MSF) Resource Library', *Microsoft Corporation*, <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/tandp/innsol/msfrl/default.asp>
Murphy 2001	Murphy T. (02 February 2001) 'Application Delivery Strategies: Achieving Usability', Meta Group, File: 947, 02 February.
Murugesan et al. 1998	Murugesan S. et al. (14 April 1998), 'First Workshop on Web Engineering', 7th World Wide Web Conference in Brisbane, Australia.
Murugesan et al. 1999	Murugesan, S., Deshpande, Y., Hansen, S. & Ginige, A. (16-22 May 1999), 'Web Engineering: A New Discipline for Development of Web-based Systems'.
Murugesan et al. 2001	Murugesan, S., Deshpande, Y., Hansen, S. & Ginige, A. (2001), 'Web Engineering: A New Discipline for Development of Web-based Systems', *Web Engineering*, LNCS 2016, pp. 3-13.
Nielsen 2000a	Nielsen, J. (2000), 'Designing Web Usability: The Practice of Simplicity', *New Riders Publishing*, Indianapolis, ISBN: 156205810X.
Nielsen 2000b	Nielsen J. (19 March 2000), 'Why You Only Need to Test With 5 Users', Jacob Nielsen's Alertbox, <http://www.useit.com/alertbox/20000319.html>
Nielsen 2001a	Nielsen, J. (5 August 2001), 'First Rule of Usability? Don't Listen to Users', *Jacob Nielsen's Alertbox*, <http://www.useit.com/alertbox/20010805.html>
Nielsen 2001b	Nielsen, J. (19 August 2001), 'Did Poor Usability Kill E-Commerce?', *Jacob Nielsen's Alertbox*, <http://www.useit.com/alertbox/20010819.html>
Nielsen 2002	Nielsen, J. (11 November 2002), 'Intranet Usability: The Trillion-Dollar Question', *Jacob Nielsen's Alertbox*, <http://www.useit.com/alertbox/20021111.html>
Nielsen &	Nielsen, J. & Norman, D. (September 2001), 'The Nielsen Norman Group Web site', *The Nielsen Norman Group*,

Norman 2001 <http://www.nngroup.com/>

Nielsen, Molich, Snyder & Farrell 2000 Nielsen, J., Molich, R., Snyder, C., & Farrell, S. (2000), 'E-Commerce User Experience'. *The Nielsen Norman Group*, ISBN: 0-9706072-0-2.

Notes Domino 2004 IBM Corporation, (31 August 2004), 'IBM Software - IBM Lotus Domino – Product Overview', IBM Corporation, <http://www.lotus.com/products/product4.nsf/wdocs/dominohomepage>

OML 2004 Outline Markup Language (14 April 2004), 'OML: Outline Markup Language', SourceForge.Net, <http://oml.sourceforge.net/>

Pressman et al. 1998 Pressman, R. S., Lewis, T., Adida, B., Ullman, E., DeMarco, T., Gilb, T., Gorda, B., Humphrey, W. & Johnson, R. (September-October 1998), 'Can Internet-Based Applications Be Engineered?', *IEEE Software*, vol. 15, no. 5, pp. 104-110.

Pressman 2000a Pressman, R. S. (January-February 2000), 'What a Tangled Web We Weave', IEEE Software, vol. 17, no. 1, pp. 18-21.

Pressman 2000b Pressman, R. S. (May 2000), 'Software Engineering: A Practitioner's Approach', *McGraw-Hill*, 5th edition, ISBN: 0077096770.

Pressman 2004 Pressman, R. S. (02 April 2004), 'Software Engineering: A Practitioner's Approach', McGraw-Hill Science, 6th edition, ISBN: 007301933X.

Pimavera 2004 Primavera Systems Inc. (25 May 2004), 'Primavera Teamplay, Suite Components, Stakeholders and Extended Project Team', Primavera Systems Inc., <http://www.primavera.com/products/team_projteam.html#TeamPlayer>

Ramsay & Nielsen 2000 Ramsay, M. & Nielsen, J. (December 2000), 'WAP Usability Déjà Vu: 1994 All Over Again, Report from a Field Study in London, Fall 2000', Nielsen Norman Group, <http://www.nngroup.com/reports/wap>

Rational 2003 Rational Software Corporation (2003), 'Rational Unified Process, Version 2003.06.01.04', *Rational Software Corporation*, <http://www.rational.com/>

RBS 2004 The Royal Bank of Scotland Group (10 April 2004), 'RBS: Welcome to The Royal Bank of Scotland', *The Royal Bank of Scotland Group*, <http://www.rbs.co.uk/>

Red Sky Interactive 2004 Red Sky Interactive (16 February 2004), 'Red Sky Interactive Homepage', *Red Sky Interactive*, <http://www.redsky.com/>

Reifer 2000	Reifer, D. J. (November-December 2000), 'Web Development: Estimating Quick-to-Market Software', IEEE Software, vol. 17, no. 6, pp. 57-63.
Rogers 2003	Rogers, E. M., (2003), 'Diffusion of Innovations', Simon & Schuster International. ISBN: 0743222091
Royce 1970	Royce, W.W. (August 1970), 'Managing the Development of Large Software Systems: Concepts and Techniques', *In WESCON Technical Papers*, v. 14, pages A/1-1-A/1-9, Los Angeles, WESCON. Reprinted in Proceedings of the Ninth International Conference on Software Engineering, 1987, pp. 328-338.
Rubin & Butler 2003	Rubin, R. & Butler, S. (May 2003), North America E-Commerce: B2B B2C, eMarketer, <http://www.emarketer.com/products/report.php?2000147>
Rumbaugh, Jacobson & Booch 1998	Rumbaugh, J., Jacobson, I. & Booch, G. (1998), 'The Unified Modeling Language Reference Manual', *Addison Wesley*, ISBN: 020130998X.
Schwaber & Beedle 2002	Schwaber, K. & Beedle M. (2002), 'Agile Software Development with Scrum', *Prentice Hall*, ISBN: 0130676349.
Sun 1999	Sun Microsystems, Inc. (23 November 1999), 'Java™ 2 Platform Enterprise Edition Specification, v1.2', <http://java.sun.com/j2ee/download.html#platformspec>
Sun 2001a	Sun Microsystems, Inc. (27 July 2001), 'Java™ 2 Platform Enterprise Edition Specification, v1.3', <http://java.sun.com/j2ee/download.html#platformspec>
Sun 2001b	Sun Microsystems, Inc. (September 2001), 'Java™ 2 Platform Enterprise Edition Web site', <http://java.sun.com/j2ee/>
Spool et al. 2002	Spool, J. et al. (Feb 2002), 'What Causes Customers to Buy on Impulse?', *User Interface Engineering E-Commerce White Paper*, <http://www.uie.com/>
Standish Group 1995	The Standish Group (1995), 'The Chaos Report 1994', *The Standish Group International, Inc.* <http://www.standishgroup.com/sample_research/chaos_1994_1.php>
Stapleton 1997	Stapleton, J. (1997), 'Dynamic Systems Development Method: The Method in Practice', *Addison-Wesley*, ISBN: 0201178893.
Taylor et al. 2002a	Taylor, M., McWilliam, J., Sheehan, J. & Mulhaney A. (March-April 2002), 'Maintenance Issues in the Web Site Development Process', Journal of Software Maintenance and Evolution: Research and Practice,

	vol. 14, no. 2, pp. 109-122.
Taylor et al. 2002b	Taylor, M., McWilliam, J., Forsyth, H., & Wade S. (April 2002), 'Methodologies and Website Development: a Survey of Practice', Journal of Information and Software Technology, vol. 44, no. 6, pp. 381-391.
Tilley 2000	Tilley, S. (2000), 'Web Site Evolution', Scott Tilley's Homepage, 2000, <http://mulford.cs.ucr.edu/stilley/research/wse/index.htm>
Visconti & Cook 2004	Visconti, M., & Cook, C. R. (5-8 April, 2004) 'An Ideal Process Model for Agile Methods' Product Focused Software Process Improvement 5th International Conference, PROFES 2004, Kansai Science City, Japan, Proceedings LNCS Vol. 3009, Bomarius, F.; Hajimu I. (Eds.), ISBN: 3-540-21421-6.
Wahl, Howes & Kille 1997	Wahl, M., Howes, T. & Kille, S. (December 1997), "Lightweight Directory Access Protocol (v3)", *RFC 2251*, <http://www.ietf.org/rfc/rfc2251.txt>
Ward & Kroll 1999	Ward, S. & Kroll, P. (July 1999), 'Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process', *Rational Software Corporation*. <http://www.rational.com/>
XP 2004	XP 2004 (6-10 June 2004), 'Fifth International Conference on Extreme Programming and Agile Processes in Software Engineering', Garmisch-Partenkirchen, Germany. <http://www.xp2004.org/>
XP Agile Universe 2004	XP Agile Universe (15-18 August 2004), 'XP Agile Universe 2004', Calgary, Alberta. <http://www.xpuniverse.com/>
Zelkowitz & Wallace 1998	Zelkowitz, M. V. & Wallace D. R., (May 1998), 'Experimental Models for Validating Technology', IEEE Computer, vol. 31, no. 5, pp. 23-31.
Zhou and Stålhane (2004)	Zhou, J. & Stålhane T. (5-8 April 2004), 'Web-Based System Development: Status in the Norwegian IT Organisations', *PROFES 2004*, LNCS 3009, pp. 363-377. ISBN: 3540214216.

Index