A PROCESS IMPROVEMENT MODEL FOR PERSONAL WEB PAGE DEVELOPMENT

by

Thomas J. Adams BSc, BInfTech

A thesis submitted in partial fulfilment of the
requirements for the degree of

Bachelor of Information Technology with
Honours

School of Computing and Information
Technology
Faculty of Communication and Information
Technology

Griffith University

1998

ABSTRACT

Interest in the World Wide Web (WWW or web) is growing at a rapid pace. As more organisations and individuals seek a presence on the WWW, the size and complexity of web pages is growing at a rapid pace. However web page development is still predominantly performed in an *ad hoc* manner, often resulting in cost overruns, quality degradation, and maintenance problems. Although some web page development methodologies exist, there is little research on the web development process itself, and no research on improving the web development process. In response to this, this thesis presents the Personal Web Process (PWP), a prototype process improvement model for personal web page development. Aimed at the individual developer and using a personal software process improvement model as a basis, the framework described incorporates existing web development methods and issues into a proven improvement model.

ACKNOWLEDGEMENTS

I would like to extend special thanks to my supervisors Dr. Jim Webb and Mr. Don Abel, whose help, advice, and support has been invaluable. Without such excellent guidance, I would not have made it through such a gruelling year. To everyone who has commented on my writing in its many forms, I thank you. I hope the improvement is evident. To my fellow Honours students, I extend my congratulations for your own personal efforts. Thankyou for your help and empathy.

I also thank my family, who have long been neglected with my educational pursuits, and without whom I would never have survived. Lastly, to my partner, Janelle Chevis, I cannot express my appreciation for your support, guidance, understanding, stress relief, and love.

STATEMENT OF ORIGINALITY

The material presented in this thesis has not been previously submitted for a degree or diploma in any university, and to the best of my knowledge contains no material previously published or written by any other person except where due acknowledgment is made in the thesis itself.

Thomas J. Adams BSc, BInfTech

13th November 1998

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Interest in the World Wide Web (WWW or web) is growing at a rapid pace. Often using the term "information superhighway," media organisations have characterised the web as a new and exciting one-stop information centre, marketing tool and research area. While some of the enthusiasm may be unjustified, the internet and the web in particular have experienced overwhelming interest from all sectors of society. Many organisations now have web sites displaying information about themselves, their products, and their services. Potential customers are able to browse through information and product lists, and even purchase products.

However as more organisations and individuals strive to gain a presence on the web, competitive and technological influences are forcing an increase in the size and complexity of web pages and sites (Lowe and Webby, 1998: 1; Bichler and Nusser, 1996a: 1). Furthermore, web page development is often performed in an informal, *ad hoc*, undocumented, and unrepeatable manner (Lowe and Webby, 1998: 3). This lack of a defined development process can lead to cost overruns, quality degradation, and maintenance problems (Lowe and Webby, 1998: 1; Bichler and Nusser, 1996a: 1).

There are currently several web site development methodologies in existence. Some are specific to web development, while others are drawn from closely related fields including hypertext, multimedia, and database research. Although these methods implicitly provide a process for performing web page development, currently no research explicitly addresses the vital task of improving the development process.

In contrast, process improvement is well addressed in the software development literature, where many frameworks exist for improving the organisational development process. These include the Capability Maturity Model (CMM), Bootstrap, TickIT, Trillium, and ISO/IEC 15504 based assessment models. On the personal level, the Personal Software Process (PSP) addresses an individuals development process. Generally motivated by technology, customer need, regulation, and competition (Humphrey, 1992: 1), these process-oriented methods have enjoyed considerable success in improving the software development process.

Therefore, acknowledging that improvement initiatives have been successful in the software industry, the similarities between web development and software development suggest that similar methods may also be successful in web development. Accordingly, this thesis presents the Personal Web Process (PWP), a prototype process improvement model for personal web page development.

The following chapter describes the research task undertaken, identifying the research problem, research objectives, the research strategy, and justification of

the strategy adopted. Chapter 3 presents a review of the relevant literature, including discussions on software process improvement, web page development issues, and web page development methodologies. The structure of the model itself is outlined in Chapter 4, including a description of the development phases and process levels used. Chapter 5 presents a summary and conclusions from the research undertaken, also identifying the limitations of this research and future work. The process forms and scripts that form part of the model are included in appendices A through D.

Chapter 2

IMPROVING THE PERSONAL WEB PAGE DEVELOPMENT PROCESS

2.1 Introduction

As identified in Chapter 1, the World Wide Web (WWW or web) is growing at a rapid pace. Recent surveys suggest that there are now more than 36 million internet hosts worldwide, compared to 26 million in 1997, and 8 million in 1995 (Network Wizards, 1998). As more organisations and individuals strive to gain a presence on the web, competitive and technological forces are increasing the size and complexity of web applications (Lowe and Webby, 1998: 1). However, while some developers may be following a defined and disciplined development process, web development is still predominantly performed as a craft, rather than an engineering discipline (Isakowitz *et al.*, 1995: 34).

This craft-like approach is well described, and addressed in the software development literature. Also, authors in multimedia products and systems development literature recognise that in order to produce quality products on time and within budget, structured development methodologies must be used to define the development process (Sherwood and Rout, 1998: 2). The literature on web page and site development also displays this recognition. Development methodologies drawn from related fields such as software, multimedia, and database design are being adapted for use in web page and site development.

However, even after acknowledging that defined and effective methodologies are needed, web development is still predominantly performed in an *ad hoc* manner. Furthermore, while web page development methodologies such as the World Wide Web Design Technique (W3DT) and the Web Site Design Method (WSDM) implicitly define a process, there is little research into this process, and no research on improving it.

In contrast, there is a significant amount of research into improving the software development process. This research has spawned organisational assessment models including the Capability Maturity Model (CMM) (Paulk *et al.*, 1995), an assessment standard: ISO/IEC 15504 (ISO/IEC JTC1/SC7, 1996), and a personal process improvement framework: the Personal Software Process (PSP) (Humphrey, 1995). This research has proven successful in assessing and improving the software development process leading to increased quality and productivity. Considering that web page development and software development both involve substantial technical skills, some form of programming, the use of abstract concepts, produce intangible products, and require intellectual effort, similar methods may also be successful for web application development.

Acknowledging these issues, this chapter presents the research task undertaken, and described by this thesis. It identifies the research question, research objectives, and the strategy adopted to complete the task.

2.2 Problem Identification

There are currently several web site development methodologies in existence. The majority drawn from closely related fields including hypertext, multimedia, and database research. As interest in the "information superhighway" increases, competitive and technological forces are influencing an increase in the scope and complexity of web pages and sites (Lowe and Webby, 1998: 1). To cope with this, proponents of these methodologies suggest their use, claiming that this will increase the quality of web pages. However, while these methods implicitly provide a process for performing the development, most do not apply to single web pages. Furthermore, none of these methods specifically address the vital task of improving this process.

The benefits of improving the software process are well known. These include productivity improvements, cost and schedule predictability, higher quality products, increased process understanding, greater employee satisfaction, healthier organisation image, coherent organisational culture, defect reduction, higher return on investment (Haley, 1996: 33; Fowler, 1997; Humphrey, Snyder and Willis, 1991: 11, 22; Herbsleb *et al.*, 1994: 11; Wohlwend and Rosenbaum, 1994: 838). Analogous to the benefits of improving the software development process, Lowe and Webby (1998: 2) also identify some potential benefits from understanding and improving the web page development process. These include improved application quality and reliability, productivity increases, improved development visibility, improved job satisfaction, increased user confidence and satisfaction, reduced development and maintenance costs, improved management

(and developer) control of the process, and more comprehensible and easier to maintain applications.

The web page (or web) development process is the sequence of steps or activities required to develop and maintain web pages (Humphrey, 1995: 4). In their definition, Lowe and Webby (1998: 3) also include the relationships between activities, the resources used, the artefacts created, the communication paths, and management of the project. The inclusion of issues that are not specific development techniques implies that existing process descriptions and development methods are not sufficient to place the process within the context of the entire development life-cycle (Lowe and Webby, 1998: 3). The consequence of this is that a complete, effective, and repeatable web development process is yet to be identified.

Lowe and Webby (1998: 4) offer a potential solution to this in process assessment mechanisms from fields such as software engineering. Furthermore, while such methods may prove useful, if the process itself is not completely understood, there is a risk of misinterpreting the results of the assessment. Therefore, if methods adapted from similar disciplines are to be used for assessing and improving of the web development process, care must be taken to ensure that an adequate process description exists. However, if used with an adequate process description, methods adapted from related fields could provide the answer to understanding the web page development process.

Summarising these issues, the research question posed by this thesis is:

How can the web page development process be improved?

This thesis therefore presents a solution to the problem of improving the web page development process.

2.3 Research Objectives

The objectives of the research task undertaken are:

1. Establish the need for an improvement model for personal web page development;

2. Develop a process improvement model for personal web page development based on the concepts and techniques outlined in the Personal Software Process;

3. Consider the implementation issues and future consequences of using the model to improve the web development process.

2.4 Research Approach

There are four well-known research strategies in software research. These are the scientific method, the engineering method, the empirical method, and the analytical method (Glass, 1994: 44).

Research using the scientific method involves observing the world, proposing a model or theory of behaviour, measuring and analysing, and validating the hypotheses of the model or theory (Glass, 1994: 44). Copi (1972: 436) expands on

these, identifying the stages in the scientific method as identifying the problem, proposing preliminary hypotheses, collecting additional facts, formulating the hypothesis, deducing further consequences, testing the hypothesis, and applying the theory developed.

Research using the engineering method involves observing existing solutions, proposing better solutions, building or developing solutions, measuring and analysing the results of these solutions, and repeating this process until no more improvements to the solutions can be made (Glass, 1994: 44). Research using the empirical method involves proposing a model, developing statistical or other methods, applying these to case studies, measuring and analysing the results, and validating the model (Glass, 1994: 44). Research using the analytical method involves proposing a theory or set of axioms, developing the theory, deriving results, and comparing these results with empirical observations (Glass, 1994: 44).

Galliers (1992: 149) further classifies these research approaches into two categories, arguing for the inclusion of empirical and engineering approaches under the umbrella of scientific research as "the former term more reasonably describes the range of approaches of this type, … and the latter term more reasonably describes the focus of the research (ie. the application area) rather than the approach to that research." Therefore he argues, software research approaches can be classified into the categories of scientific and interpretivist. Scientific approaches include experiments, surveys, case studies, theorem proof, forecasting, and simulation (Galliers, 1992: 149). Interpretivist approaches include reviews,

action research, subjective and descriptive studies, and role playing (Galliers, 1992: 149).

Within these approaches, Glass (1994: 44) cites a four-step process for conducting research. These steps are the information gathering stage, the propositional phase, the analytical phase, and the evaluative phase. The information gathering stage involves gathering information through literature review, reflection, or survey (Glass, 1994: 44). The propositional phase involves proposing or building a hypothesis, method, algorithm, theory, or solution (Glass, 1994: 44). The analytical phase involves analysing and exploring a proposal, leading to the demonstration and/or the formulation of a principle or theory (Glass, 1994: 44). The evaluative phase involves evaluating a proposal or finding by means of experimentation or observation, leading to a revised model, principle or theory (Glass, 1994: 44).

To aid in the identification and construction of a suitable research approach for the research task undertaken, an analysis of the overall research objectives can be performed. The results of using Basili *et al.*'s (1994) Goal/Question/Metric approach to organise this analysis can be seen in Table 1.

Goal 1

To identify the current state of the web page development process

Questions

- What is the current state of the web page development process?
- How can the current state of the web development process be determined?
- What measurements can be made to determine the state of the web page development process?
- sAre there any existing assessment methods that can be used for this?
- Are there any existing assessment methods that can be modified for this?

Metrics

- Subjective and anecdotal descriptions of the current state
- Suitable metrics for evaluation
- Descriptions of applicable or modifiable assessment methods

Goal 2

To develop a means of improving the web page development process

Questions

- What is the current state of the web page development process?
- How can the current state of the web development process be determined?
- What measurements can be made to determine the state of the web page development process?
- How can the web page development process be improved?
- Are there any existing improvement models that can be modified for this?

Metrics

- Subjective and anecdotal descriptions of the current state
- Suitable metrics for evaluation
- Descriptions of applicable or modifiable assessment methods
- A suitable improvement model
- Descriptions of suitable of modifiable improvement models
- Productivity and quality measurements

Goal 3

To identify factors that will increase the effectiveness of the improvement framework

Questions

- How can the web page development process be improved?
- What issues influence the effectiveness of the improvement effort?
- Can the effectiveness of the improvement program be measured?
- How can the effectiveness of the improvement program be measured?
- Has any research addressed the issues of measuring the effectiveness of improvement models?

Metrics

- Productivity and quality measurements
- Subjective and anecdotal descriptions of the current state
- Subjective descriptions of the improvement process
- Quantitative and qualitative measures of the influence of the improvement model

Goal 4

To consider the implementation issues and future consequences of using the framework to improve the web development process

Questions

- What does the framework look like?
- How does it work?
- What components make up the framework?
- How do the components of the framework work together?
- What are the implementation issues associated with the framework?
- What are the implementation issues associated with similar frameworks?
- How can the implementation issues be identified?
- Has any research been conducted on identifying these implementation issues?

Metrics

- A description of the framework
- A explanation of the workings of the framework
- A listing and description of the components of the framework and their interaction
- Implementation issues from similar models
- Descriptions of research undertaken to identify implementation issues for similar frameworks

Table 1. Research Goals, Questions, and Metrics

Table 1 shows the goals, questions, and metrics used to formulate the research approach. Goals identify issues of focus at a conceptual level (Basili *et al.*, 1994: 3). Questions are then used to place the goal within a particular viewpoint, characterising the way the goal will be achieved (Basili *et al.*, 1994: 3). Metrics can then be used to provide an objective or subjective quantitative measurement that can be used to answer the questions, and hence determine if the goal as been achieved (Basili *et al.*, 1994: 3).

Recognisable through the analysis described in Table 1, developing an answer to the research question involves two steps, namely evaluation and improvement. Before any improvement can be initiated, knowledge and understanding of the current process must be acquired. Humphrey (1995: 450) states this as: "to improve human-intensive processes, you need to understand how they work currently." While we can look to existing development methodologies for answers relating to the development steps and activities, the process they define does not provide adequate scope for improving the process.

In Humphrey (1995: v), Basili highlights the use of scientific and engineering methods in improving the practice of software development. Basili *et al.* (1994: 1) also describes the need for a "measurement mechanism for feedback and evaluation." Acknowledging that software process assessment and improvement can provide the mechanism for establishing feedback and evaluation loops, and the similarities between software development and web page development, similar assessment and improvement methods could be used for improving the web development process.

Considering the issues presented above, the research task undertaken uses an engineering approach. The first stage in using an engineering approach to research involves observing existing solutions. As solutions to the problem of improving the software process and web page development methodologies have been defined and in some cases validated, this stage was conducted by performing a review of the relevant software and web page development literature, and participating in a personal software process improvement initiative (see Chapter 3).

The second stage of the engineering approach involves proposing improved solutions to the problem. Solutions to the research problem could include adopting new development methodologies, increasing spending on development activities, hiring more development staff, and adopting new technologies. Although these strategies may be successful in solving the problem, the solution adopted in this thesis is based on a personal process improvement model for software development.

The third stage of the engineering approach involves developing new solutions. This stage involved construction of the Personal Web Process (PWP), a process improvement framework for personal web page development. This work is presented in Chapter 4. Due to the limited timeframe of this work the final stage of the research approach was not conducted. However, issues related to this are discussed in Chapter 5.

Chapter 3

THE STATE-OF-THE-PRACTICE

3.1 Introduction

Many different methods have been used to facilitate software process improvement (SPI). These include at the personal level the Personal Software Process (PSP), and at the organisational level ISO 9001, Business Process Orientation (BPO), Business Process Re-engineering/Re-design (BPR), Bootstrap, the Capability Maturity Model (CMM), TickIT, Trillium, ISO/IEC 15504 based assessment models, and the Software process Improvement Model (SwIM). The later six fall into the category of software process assessment (SPA) methods, while BPO and BPR rely on general business re-orientation to achieve improvement (Gruhn and Wolf, 1995: 49; Gasston, 1996: 171). The PSP is unique in that it provides an *individual* process improvement framework of forms, guidelines and processes, where participants improve their personal process through exposure to good software engineering practices (El Emam *et al.*, 1996: 119).

While software process improvement is well understood in the software development industry, there are no methods that address process improvement in the web development field. Currently, several development methodologies attempt to define a web or hypermedia project lifecycle. These include the Hypermedia Design Model (HDM), the Object-Oriented Hypermedia Design

Model (OOHDM), the Relationship Management Model (RMM), the Web Site Design Method (WSDM), and the Structured Methodology for Multimedia Development (SMMD). Therefore, if one of the first steps in process improvement is to understand the process being improved, these methods should provide the structure necessary to allow investigation into the process, possibly leading to improvement.

This chapter introduces software process improvement at the personal and organisational level. It outlines many of the well-known improvement models, and gives a detailed overview of the Personal Software Process. Web development issues and models are then presented, including models from related fields such as hypertext development.

3.2 Software Process Improvement

Software has become critical to the successful functioning of modern society. As the software community strives to come to grips with software development, the problems in development have become so prominent that the term ‘software crisis’ is often used to describe them (Glass, 1994: 43). With these problems increasing, one apparent solution is to adopt methods to characterise, define, and control the software development process. If we accept the premise that product quality is largely governed by the quality of the processes used to build it, then the process must itself be of high quality (Ceberio, 1995). However, a high quality development process by itself is not enough to ensure continued high quality. To consistently achieve this, the process must be continually improved.

Software process improvement can be viewed as the operation of putting in place measures to strengthen weaknesses identified in processes, and it can be addressed at several different levels (Ibrahim and Hirmanpour, 1995: 23). Several models exist for these different levels. At the organisational level, the CMM developed by the Software Engineering Institute (SEI) provides a framework for evolutionary improvement through five levels of organisational maturity (Paulk, 1995: 4). At an individual level, Humphrey's (1995: 1) PSP provides a method for software engineers to improve their personal process through the introduction of good software engineering practices.

3.2.1 An Historical Perspective

Software process improvement has its origins in the Total Quality Management (TQM) methodologies championed by Deming, Juran, and Crosby (Youssien, 1998: 15). TQM is a holistic approach to quality, and has at its heart a perpetual cycle of improvement in "every aspect of the process by focussing on reduction or elimination of the impact of special causes in our processes" (Youssien, 1998: 15). Employed extensively in Japan from the early 1950s, *kaizen*, as it came to be known, concentrates on continuous and incremental improvement (Fowler and Rifkin, 1990: 5).

Historically, many industries in other countries have concentrated on "quantum leaps" for quality improvement; placing heavy reliance on innovation and outstanding people (Youssien, 1998: 17; Arthur, 1997: 49). More recently in the software industry, the benefits of continuous improvement are increasingly being recognised as an essential aspect of quality management (Fowler and Rifkin: 1990: 5). This move towards incremental and continuous improvement, coupled

with the applicability of traditional engineering management methods to software development means that the software industry in general has been more receptive to improvement efforts (Humphrey, 1992: 2).

Therefore, if the goal of process improvement is viewed as establishing processes of higher quality, the capability of the organisation's software development processes must first be determined. A capable process is a process with the inherent ability to produce planned results, and is often characterised as mature (Humphrey, 1992: 1). It is therefore by means of assessment that the capability of a process can be gauged. In reference to ISO/IEC 15504 based methods, Rout (1998: iv) considers software process assessment as:

> *"The disciplined examination of the processes used by an organisation against a set of criteria to determine the capability of those processes to perform within quality, cost and schedule goals. The aim is to characterise current practice, identifying strengths and weaknesses and the ability of the process to control or avoid significant causes of poor quality, cost and schedule performance."*

While software process assessment is usually considered to be one of the first steps in software process improvement (Fowler and Rifkin, 1995: 5), there is no obvious link between the two. The basis for a relationship could be established by considering two often quoted proverbs; suggesting that before you can start to improve, you need to know your current position.

> *If you don't know where you are going, any road will do.*

> *If you don't know where you are, a map will not help.*

Indeed Rout (1998: iv) makes the distinction that process assessment is often seen to be a "strong and effective driver" for process improvement. The ISO/IEC 15504 standard (ISO/IEC JTC1/SC7, 1996: 4) for software process assessment concurs, stating that software process assessment leads to the identification and selection of key activities for improvement and the continuous application of improvements to match business needs. Therefore, by providing organisations with a view of where they are, and where they could be, software process assessment can often be used in developing a "road map" for improvement.

3.2.2 Motivation

Humphrey (1992: 1) states that technology, customer need, regulation, and competition generally motivates the widespread adoption of software process improvement initiatives. While it seems reasonable to expect that product quality has an affect on customer satisfaction, Paulk *et al.* (1995: 3) comments that this is often considered the "weak link" in software development. Historically software organisations have relied on quality insertion at the end of the development process, commonly achieved by comprehensive testing procedures. However, there now seems to be widespread agreement that quality must be added throughout the development process (Fowler and Rifkin, 1990: 5). Many authors agree with this, proposing that the benefits of using software process improvement methods far outweigh the disadvantages. These benefits and disadvantages are summarised in Table 2.

Benefits	Disadvantages
Productivity improvements	Greater staff turnover
Cost and schedule predictability	Increased training commitment
Higher quality products	Cost
Increased process understanding	
Greater employee satisfaction	
Healthier organisation image	
Coherent organisational culture	
Defect reduction	
Higher return on investment	

Table 2. Benefits and Disadvantages of Software Process Improvement[1]

3.2.3 Methods and Models

3.2.3.1 Personal Improvement

Personal ability in within the context of this thesis can be considered to be an individual's ability to develop programs, or to solve a particular set of software problems. By repeatedly solving problems, software engineers gain a skill-set specific to the domain in which they are working, learning to adapt these skills to other domains when and where necessary. It has been reported that the abilities of software engineers can have significant effects on productivity. As early as 1981, Boehm (1981: 642) reported that the differences between high and low performing personnel impacted significantly on productivity, while Curtis, Krasner, and Iscoe (1988: 7) have described the phenomenal impact exceptional people could have on projects.

[1] Derived from Haley, 1996: 33; Fowler, 1997; Humphrey, Snyder and Willis, 1991: 11, 22; Herbsleb *et al.*, 1994: 11; Wohlwend and Rosenbaum, 1994: 838.

However most competency requires demonstrated proficiency with established methods (Humphrey, 1995: 3). Put differently, the ability to improve on one's performance often only comes with a proven mastery of basic techniques. Humphrey (1995: 3) considers discipline to be the encapsulation of years of knowledge and experience, stating that "a skilled professional can outperform even the most brilliant but untrained laymen." Within this broad definition, the personal discipline needed to complete today's complex software development tasks can be considered to be the adoption by an individual of definitions, methods, and metrics provided by a personal improvement framework.

The Personal Software Process (PSP) developed by Watts Humphrey, provides such a discipline. The PSP attempts to improve the personal practices of software engineers by exposing them to good software engineering practices (El Emam *et al.*, 1996: 119). It provides a structured framework of forms, guidelines and processes for software development (Humphrey, 1995: 1). The students are guided through seven phases (see Figure 1), enabling them to develop their personal process "from simple concepts, such as project planning, to advanced levels of process maturity, such as defect prevention" (Khajenoori and Hirmanpour, 1995: 132). Each process level (or phase) builds on the previous level by adding several process steps. This allows a mastery of existing concepts, and minimises the impact of new ones.

The PSP presents a discipline for software engineering, capturing concepts from industrial software practices, and scaling them down to a personal level (Humphrey, 1994: 70). Humphrey has selected twelve of these processes that can

be utilised at a personal level. These are noted with an asterisk in Figure 2. This allows individual software engineers to practice established methods in their everyday work.

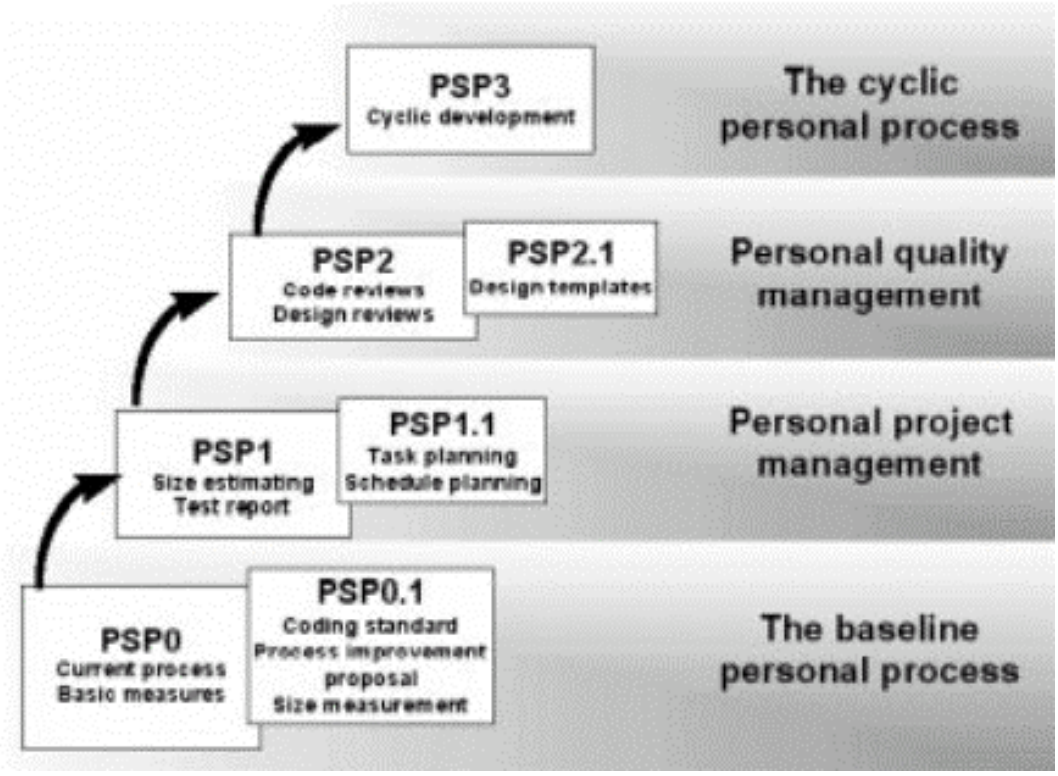

Figure 1. PSP Process Levels

Source: Hayes and Over, 1997: 7

Controversy exists on the exact nature of the PSP (see Schneider, 1998). Many participants feel that the process defined in the training course is the *only* process that can be used. Implying that if the development processes carried out by the individual or organisation involved in the improvement initiative are in conflict with the ones presented in the PSP course, the PSP is not relevant to the participant at all. However, this viewpoint is invalid for several reasons.

Firstly, the PSP is not just a technology, or the elusive "silver bullet" long-promised in the software development community. Viewing the PSP as a panacea will not solve the problems of individuals or organisations involved in software development. Secondly (and closely related to the first point), the PSP can be viewed from several levels. At one level, the PSP provides a defined method for improving the personal software development process. From another level, the PSP is a meta-process, in which specific process improvement models can be constructed (Schneider, 1998). Adopting the latter view, the PSP can be used to construct a development process that evolves to suit a specific user environment (Schneider, 1998). Therefore, if one user environment does not use Lines Of Code (LOC) as the counting standard, something more appropriate may be substituted. If an organisation uses graphical development environments, the PSP framework can be adopted for these tools. The PSP does not purport to provide a one-stop definitive solution to the problem of improving the software development process. As emphasised by Schneider (1998), "the key is to have a process, not a particular process." The forms and scripts provided by the PSP provide a framework for defining and measuring the process, the results of which can be used to modify the process. In essence, it is this modification of the process that drives the improvement.

3.2.3.2 Organisational Improvement

The "software crisis" has forced many organisations to search for solutions that attempt to manage the software process (Paulk *et al.*, 1995: 3). The Capability Maturity Model (CMM) developed by the SEI is one such solution. The CMM is a framework that describes key elements of an effective software development

process, providing a path for an organisation to follow, from an *ad hoc* immature process, to a mature, disciplined one (Paulk *et al.*, 1995: 3).

Five maturity levels are defined that characterise the activities of a software organisation at that level. At each maturity level, key processes are introduced, which provide an organisation with the means to evaluate their current practices, and position themselves for future improvement (see Figure 2).

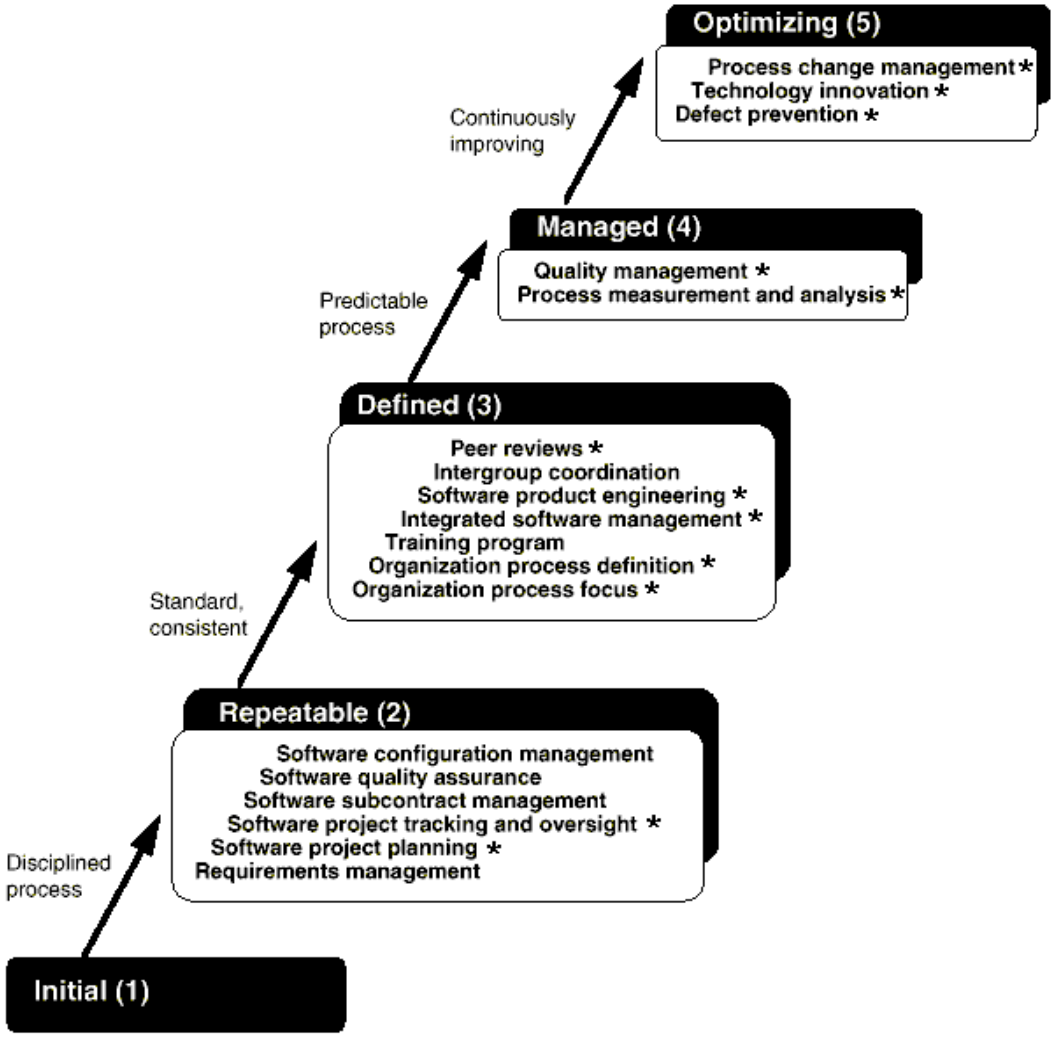

Figure 2. CMM Key Process Areas by Maturity Level (adapted from Thomson, 1998: 191)

3.3 Web Page Development

Use of the World Wide Web (WWW or web) has increased remarkably in the last few years. Originally a facility for information sharing and presentation in a networked environment, the web has grown enormously in usage and popularity. Now encompassing the world of business, many organisations are seeking a presence online, viewing the web as the ultimate marketing tool. However as more organisations struggle to gain a presence on the information-superhighway, competitive and technological forces are driving an increase in the size and complexity of web pages and sites. This increase leads to problems developing and maintaining web sites, leading to quality degradation and monetary losses.

However despite this increase in size and complexity, web page development is still largely performed in an *ad hoc* manner. This lack of a defined development process has driven the creation of methodologies for web development. Stemming from the closely related hypermedia, software, and multimedia fields, these methodologies may be considered directly applicable to web page development. Indeed, owing to these methodologies, similar web-specific methodologies have been constructed in an effort to describe, define, and regularise the web page development process. Method-independent initiatives are also underway to improve web page development. Issues such as metadata, hyperlink management, interface design, and configuration management have all been identified as problem areas.

This following section addresses these issues; starting with a historical perspective on web page development, issues and guidelines are discussed,

followed by a detailed summary of web page and hypermedia development methodologies.

3.3.1 An Historical Perspective

Historically, web page development has been performed using HyperText Markup Language (HTML) as the implementation language. A subset of the Standard Generalised Markup Language (SGML), HTML was originally developed by Tim Berners-Lee to allow access to structured documents in a networked environment (W3C, 1998a). Labeling this distributed document system as the World Wide Web, he based its architecture on the hypertext navigational paradigm described by Ted Nelson in the Xanadu system (Andrews, 1996: 1; Blustein, 1998). Hypertext can be defined as:

> *Text which does not form a single sequence and which may be read in various orders; specially text and graphics ... which are interconnected in such a way that a reader of the material (as displayed at a computer terminal, etc.) can discontinue reading one document at certain points in order to consult other related matter (Simpson and Weiner, 1993).*

Originally consideration was limited to only text-based information. The hypertext concept was then broadened to hypermedia, by including objects such as graphics, sound, and video (Blustein, 1998). However, as the relationship between hypertext and hypermedia is not well defined, they are often used interchangeably.

Consequently, the definition of a publishing language, coupled with a hypertext architecture witnessed the creation of a program that could represent this language in a presentational form. Starting with only a textual presentation interface, the Mosaic browser popularised the web, leading to the inclusion of non-textual

objects such as graphics, sound, and video into the web architecture. The addition of these objects and its current architecture means that the web now represents a distributed hypermedia system. Therefore, if the web is considered as a hypermedia system, then research in the hypermedia development field is directly applicable to the web development field.

3.3.2 Domain Development Differences

While we can use approaches developed in domains such as software development, web development remains notably different. These differences include a more incremental and iterative development lifecycle, finer grained maintenance, and cognitive management aspects (Lowe and Webby, 1998). Isakowitz *et al.* (1995: 34) make the observation that the use of prototyping and intensive testing with users is more prominent with hypermedia development than with software development because of a lower error tolerance. Furthermore, Nanard and Nanard (1995: 49) state that:

> *"Fundamental differences, however, make a pure transposition of techniques both difficult and inadequate. An important part of hypertext design concerns aesthetic and cognitive aspects that software engineering environments do not support."*

On a more general level, hypermedia development also includes the capturing and organisation of a complex information space (Isakowitz *et al.*, 1995: 34). The unstructured and dynamic nature of this information poses new questions and problems, including representing this information in a way that makes it accessible to users (Isakowitz *et al.*, 1995: 34), and developing suitable navigational structures and pathways (Schwabe *et al.*, 1996: 2).

Web page development often involves presenting a unified view of disparate resources such as multimedia objects, database systems, and textual information. The challenge of presenting this has promoted the development of new skill-sets in what has traditionally been a programming field. Bichler and Nusser (1996b: 3) comment that developing web pages often involves artistic skills. In addition Isakowitz *et al.* (1995: 34) state that web page development projects may draw people such as authors, librarians, content designers, artists, musicians, and programmers together. This combination of varied skill-sets and people with varying backgrounds indicates a vast difference from traditional established fields such as software engineering.

Although there are differences in general approach, skill-sets, and lifecycles, web page development is in essence quite similar to existing fields such as software development. These similarities include the use of abstract concepts, an intangible product, and the intellectual effort required. So while methods developed in other domains cannot be directly applied to web page development, they can be used if modified appropriately.

3.3.3 Issues and Guidelines

In a framework that attempts to improve the web page development process, problems and issues relating to current practice must be identified. Pam (1995: 1) identifies several problems with the current web architecture, including versioning, metadata, and link consistency (Pam, 1995: 1). Other important issues include web site management, analysis, and user interface design. As such, this section identifies the issues and guidelines related to web page development that are important to improving the process.

3.3.3.1 Web Site Management

A web site is a set of collected, interconnected documents under the same management (Schwabe *et al.*, 1996: 2). Managing a web site encompasses managing information, establishing guidelines and standards, developing and maintaining pages, ensuring link consistency, and applying version control. While managing a web site is an important function within the context of the complete web development lifecycle, as the scope of the research undertaken is limited to an individual developer, this issue is not considered.

3.3.3.2 Analysis

Analysis forms an important part of any development process. Analysis within a web development context includes project feasibility studies, purpose identification, requirements solicitation, visual layouts and guidelines, implementation guidelines and languages, audience identification and classification, and security. Within a web development organisation, analysis can often be performed independently of design, implementation, and testing. Indeed all these development stages can be performed independently by separate groups of people. However, as the process improvement model presented in this thesis relates to the improving the personal web page process, its scope is limited to issues that an individual developer would encounter. Accordingly, analysis issues are outside the scope of this research, and are hence not considered.

3.3.3.3 Aesthetics and User Interface Design Guidelines

As aesthetics and user interface design guidelines form part of the analysis phase, they are not considered within the context of this thesis.

3.3.3.4 Metadata

Metadata is information about documents (Newmarch, 1998: 1). Within the context of web page development, metadata inserts information about the web page within its source code. The Dublin Core (DC) metadata set describes an accepted way of setting this information within web pages (Newmarch, 1998: 2). The DC set of descriptors includes: Title; Creator; Subject; Description; Publisher; Contributors; Date; Type; Format; Identifier; Source; Language; Relation; Coverage; and Rights.

Figure 3 identifies the use of the DC descriptors within a web page element.

```
<META name="DC.author" content="Tom Adams"
<META name="DC.author" content="(TYPE=name) Tom Adams">
<META name="DC.author" content="(TYPE=phone) +61 7 3875 6526">
```

Figure 3. Dublin Core Meta data Descriptors

3.3.3.5 Configuration Management and Versioning

Due to the limited scope of this research, configuration management and version control issues were not considered.

3.3.4 Development Models, Methods, and Processes

Although research into the web development process is limited, several development models and methodologies do exist. Some are specific to web development (WSDM, W3DT), while others are drawn from closely related fields including hypertext (HDM, OOHDM), multimedia (SMMD), and database design (RMM) research. The Enhanced Object-Relationship Model (EORM) is also cited

by Lowe and Webby (1998: 3) as a methodology adapted from database design, however it is no presented within this thesis. This section presents a summary of these, within the context of their relationship to the improvement framework presented in Chapter 4.

3.3.4.1 Hypermedia Design Model (HDM)

Garzotto *et al.* (1993: 1) believe that hypermedia development would benefit from a systematic, structured development, "especially the case of large and complex applications." Such a structured approach the authors state, suggests the notion of *authoring-in-the-large* (Garzotto *et al.,* 1993: 1). Authoring-in-the-large refers to "the specification and design of global and structural aspects of the hypertext application" (Garzotto *et al.,* 1993: 2). This contrasts with *authoring-in-the-small*, which is concerned with "local" tasks, such as developing the content of particular screens and nodes (Garzotto *et al.,* 1993: 2). As authoring-in-the-large is applicable to many different applications developed within a single application domain, and is largely independent of the chosen implementation environment (Garzotto *et al.*, 1993: 2), any development model that takes into account issues such as authoring-in-the-large would provide a structured and systematic method that is applicable to a wide variety of applications and environments. The Hypertext Design Model (HDM) purports to be such a model.

HDM is a model to describe hypertext applications; applications modeled using HDM are labeled HDM models. Its intended use is in helping to conceptualise an application without regard for the "implementation details," and may also be used as a communication language between developers (Garzotto *et al.*, 1993: 3). The authors also state that HDM is a first step towards automated development tools,

analogous to CASE tools in software development. In a manner similar to database models, a HDM specification of a hypertext application makes a clear distinction between a conceptual *schema* and an *instance of the schema* (Garzotto *et al.*, 1993: 11). "A HDM schema is a collection of type definitions that describe an application at the global level; an instance of a schema is a collection of entities, components, units, and links" (Garzotto *et al.*, 1993: 7).

A HDM model consists of large structures of information representing physical or conceptual objects called *entities*, which are grouped by similarity in *entity types*. Entities are composed of *components*, which are in turn composed of *units*. Units are closely related to the standard notion of a hypertext "node." These information structures can be connected by three types of *links*. *Structural links* connect components of the same entity; *perspective links* connect units of the same component; and *application links* denote implementation environment specific relationships that connect entities and components in arbitrary patterns.

3.3.4.2 Object-Oriented Hypermedia Design Method (OOHDM)

Schwabe *et al.* (1996) propose that the new and complex problems posed by the dynamic nature of hypermedia information can only be solved in a systematic and modular fashion. To meet these challenges, they put forward the Object-Oriented Hypermedia Design Method (OOHDM). The authors of the model claim that OOHDM satisfies the criteria of a structured and systematic methodology, while still "maintaining the exploratory nature of the hypermedia" development process (Schwabe *et al.*, 1996). Based on the Hypertext Design Model (HDM), OOHDM is a model-based development methodology for developing hypermedia applications. The OOHDM development process consists of four phases

(conceptual design, navigational design, abstract interface design, and implementation), and supports an incremental process model.

The first development stage is conceptual design. This stage involves creating a model of the application using well-known object modeling techniques. Conceptual object classes are built using standard object relationships including generalisation, aggregation, and association (Schwabe *et al.*, 1996). The navigational design phase describes the navigational structure of the application in terms of navigational constructs. Navigational contexts are navigable paths derived from conceptual navigational classes such as nodes, links, indices, and guided tours.

The abstract interface design stage uses the navigational structure defined in the previous stage to develop an abstract interface model. This model, called an Abstract Data View (ADV), describes the application's user interface in an implementation independent way. ADVs are object models of interface objects and describe the static layout of the interface; the appearance of navigational objects, the static relationship between interface objects and navigational objects, and the reaction of interface and navigational objects to external events. The final development stage is implementation. In essence, implementation maps the navigational and abstract interface models to concrete objects in an implementation environment. The methodology's authors claim that using object-oriented techniques simplifies the implementation of these models in different environments.

3.3.4.3 Relationship Management Model (RMM)

Isakowitz *et al.* (1995: 34) view hypermedia applications as vehicles for managing relationships among information objects. They propose the Relationship Management Data Model (RMDM), a model similar to the Hypertext Design Model (HDM) for describing these relationships. RMDM defines modeling primitives including *entities*, *attributes*, *relationships*, *slices*, *links*, *indexes*, and *guided tours*. Although RMDM describes a representation scheme for hypermedia applications, is does not describe how to use the model as part of the development process (Isakowitz *et al.*, 1995: 36). The Relationship Management Methodology (RMM) does this. The authors of RMM claim that it differs from other methodologies in its "recommended sequence of steps, additional access structure formalisms, increased emphasis on graphic representations, and a more detailed, step-by-step procedure for hypermedia design and development" (Isakowitz *et al.*, 1995: 36).

The RMM development process consists of eight phases. These are entity-relationship design, entity design, navigational design, conversion protocol design, user-interface screen design, run-time behaviour design, construction, and testing and evaluation. These phases are shown within the context of the complete development lifecycle in Figure 4.

In the entity-relationship design phase, the information space of the application is described by the construction of an Entity-Relationship (ER) diagram. The entities and relationships identified in this stage form the basis of the application, often appearing in the final product as links and nodes (Isakowitz *et al.*, 1995: 36). The

entity design phase determines how the entities identified in the ER design phase will be presented to users of the application. Large entities may be split into smaller units of information called *slices*. This distribution of entities into slices is called the *slice design phase*, and results in the construction of a slice diagram, describing slices and navigational paths between them (Isakowitz *et al.*, 1995: 40). The navigational paths are designed in the navigational design phase. Each associative relationship from the ER diagram constructed in the ER design phase is analysed, and if accessible for navigation, it is relaced on the diagram by an RMDM access structure. At the end of this phase, the ER diagram will have been transformed into an RMDM diagram.

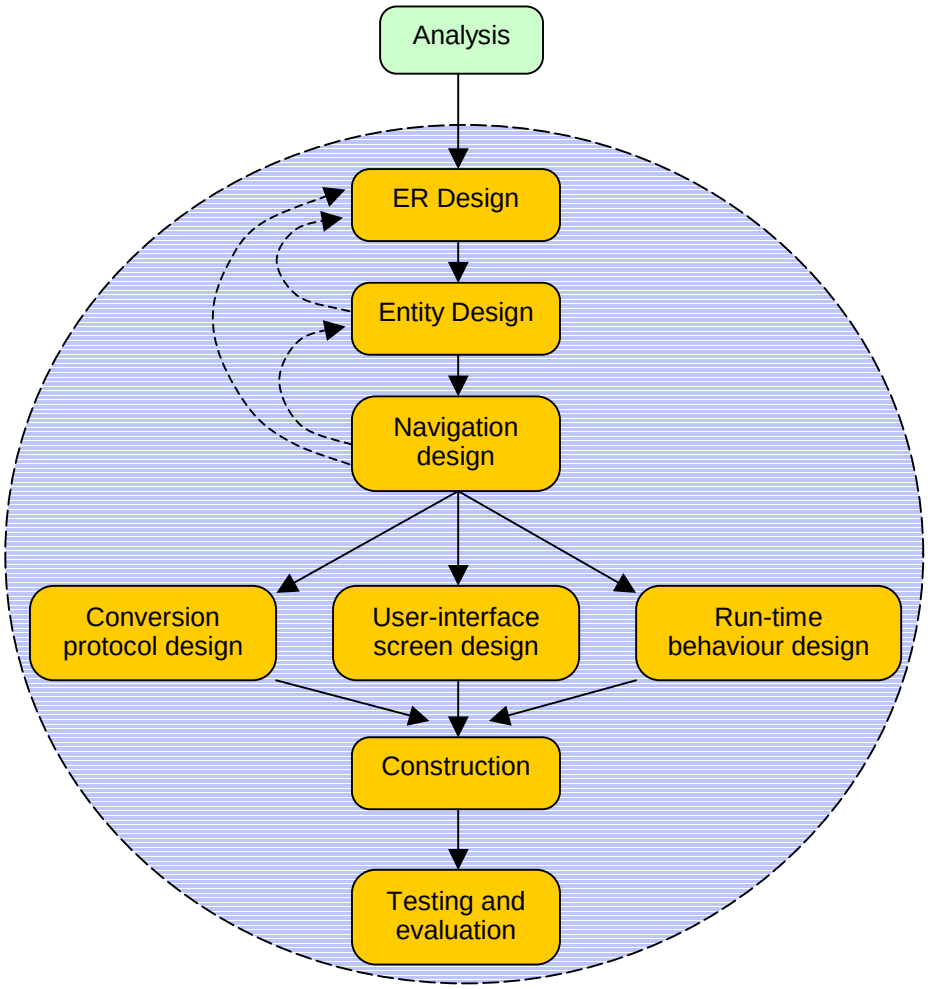

Figure 4. RMM Development Phases (adapted from Isakowitz *et al.*, 1995: 38)

The Conversion protocol design phase "uses a set of conversion rules to transform each element in the RMDM diagram into an object in the" implementation environment (Isakowitz *et al.*, 1995: 38). This process can be "performed manually," or with an automated tool (Isakowitz *et al.*, 1995: 38). The user-interface design phase involves the design of screen layouts for all objects in the RMDM diagram, including buttons layouts, general appearance, and location of navigational links. The run-time behaviour design phase designs the implementation (environment) specific functionality of the navigational mechanisms. The last two stages – construction and testing and evaluation – consist of constructing and testing the completed hypermedia application.

While this methodology is one of the more descriptive and complete hypermedia development methodologies, it does have some confusing aspects. The order in which these last three stages are to be performed is not clear. From the development phase diagram presented in Figure 4, it would appear that they are performed in parallel. However, in the description of the methodology, its authors make no mention of ordering. In addition, the conversion protocol design phase appears to be the implementation phase, rather than just a design phase as its name implies. The statement that this process can be performed manually or with an automated tool such as an RMDM to HTML compiler supports this. If this stage were in fact the implementation phase, then it would seem that it should be performed last, after the interface and run-time design. However, this would make the construction phase redundant. An improved solution would have the conversion protocol design phase constituting design activities only, possibly creating the set of conversion rules, rather than implementing them. The

implementation of the conversion rules could then be performed in the construction phase.

3.3.4.4 World Wide Web Design Technique (W3DT)

The World Wide Web Design Technique (W3DT) was constructed to aid in the development of large web sites. The authors of the methodology – Bichler and Nusser (1996: 2) – consider that existing hypermedia development methodologies derived from the database design field are not useful for an information space as unstructured as the web. They claim that while hypermedia application developers tend to create nodes and links in a straightforward and structured manner, the web supports both highly structured and highly unstructured information, and therefore development methodologies for web development should support both these (Bichler and Nusser, 1996a: 2). W3DT purports to support the development of web sites whose information domain contains structured information, such as database front-ends; as well as web sites whose information domain contains unstructured information.

The W3DT development process consists of four stages, identified within the context of the entire development lifecycle in Figure 5. These are information structuring, navigational design, organisational design, and interface design. The information structuring and navigational design stages organises and structures the information space, and creates navigational paths through these structures. The organisational design phase assigns information on maintenance and organisational integration to information objects (Bichler and Nusser, 1996a: 3). The final stage is interface design. This stage designs the web page interface, and includes issues such as attractiveness, and general aesthetic aspects.

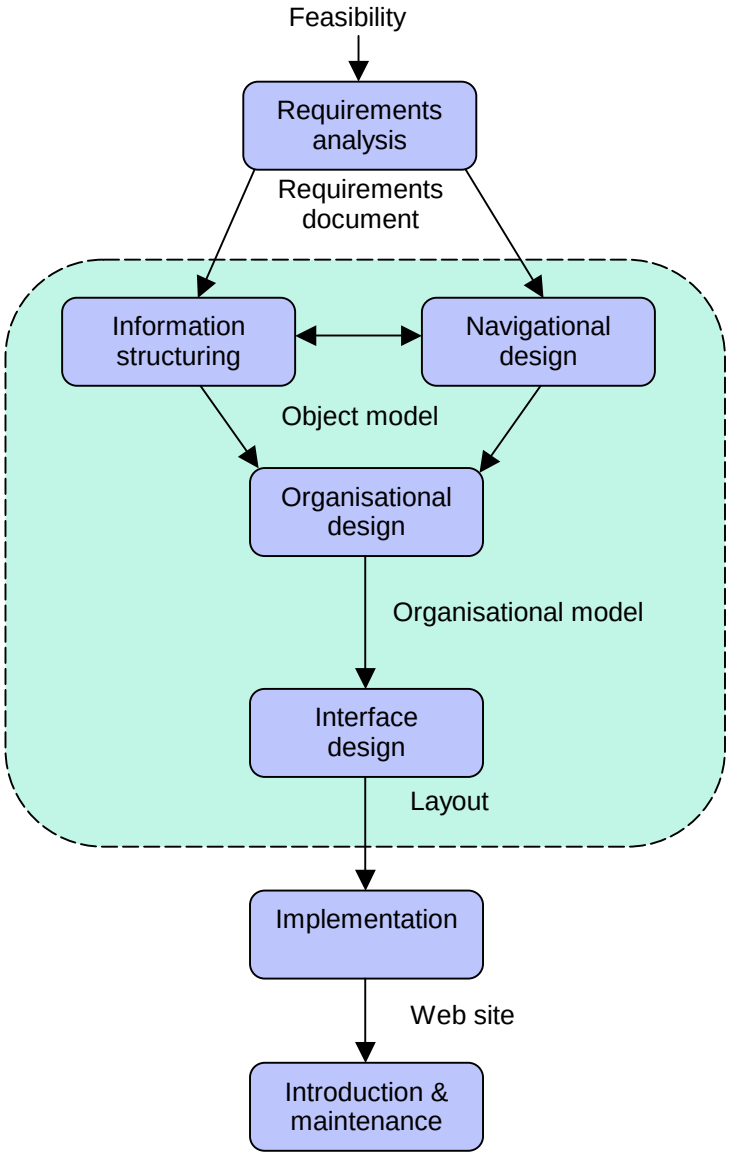

Figure 5. W3DT Development Phases (adapted from Bichler and Nusser, 1996a: 2)

While the W3DT methodology supports the development of web pages and sites, it does not describe the entire development process. Issues described in the interface design stage such as attractiveness and aesthetic concerns clearly form part of an analysis phase. Also, the main focus of the methodology is on design of the web page only, with little attention paid to the rest of the development lifecycle. This means that essential phases including analysis, implementation, and testing are neglected, or confused within other stages. Although slight

mention of these phases is made, attention is only drawn to their existence within the entire process, meaning that overall the model does not adequately describe the web page development process.

3.3.4.5 Web Site Design Method (WSDM)

De Troyer and Leune (1998) view web development from two perspectives, that of the visitor (faced with usability problems) and that of the manger (faced with maintenance problems). They argue that while methods exist for developing web pages, some (HDM, OOHDM, and RMM) are based on hypertext research, and hence do not deal with web-specific issues, while others are implementation or data driven (De Troyer and Leune, 1998). These, they say, are capable of solving the maintenance issues of managers, but are unable to solve the usability problems of visitors.

In response to this, they propose a model for developing kiosk[2] web sites that focuses on the structuring of large amounts of information, leading to improved usability of the web site. The Web Site Design Method (WSDM) is "user-centred[3]", and focuses on the set of potential visitors to the site (De Troyer and Leune, 1998). It models data from the perspective of different user classes, potentially making the final product more usable for members of these classes, and distinguishes between implementation-independent and implementation-dependent design.

[2] A kiosk web site supplies mostly information, and provides users with a navigational path through the information. As opposed to application web sites, which can be viewed as interactive information systems.
[3] User-centred in this context implies that the foremost priority of web site development is the potential set of users.

The WSDM methodology consists of four major stages (outlined in Figure 6). These are user modeling, conceptual design, implementation design, and implementation. The user modeling and conceptual design stages consist of several sub-phases, including user classification and class description, and object modeling and navigational design.

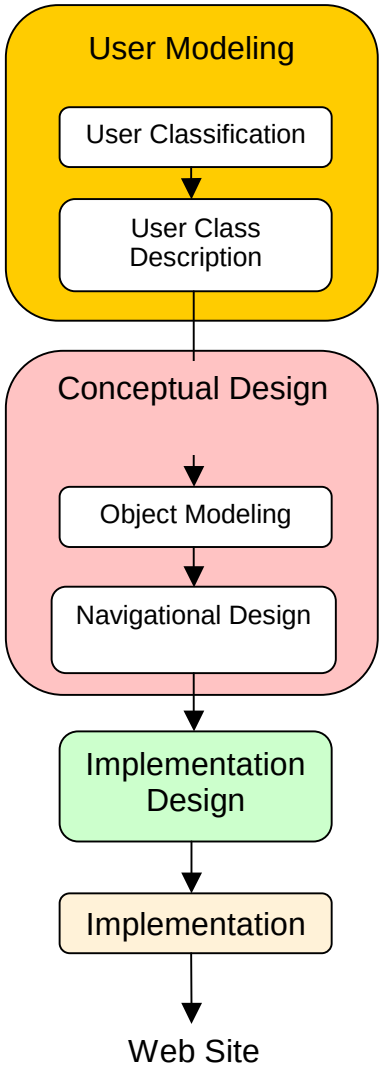

Figure 6. WSDM Development Phases (adapted from De Troyer and Leune, 1998)

The user classification stage identifies and classifies users of the proposed web site into groups. The user class description stage focuses on the information requirements of the classes identified, and attempts to describe the characteristics

of these classes. If users within one class have different characteristics, the class is split into *perspectives*, which describe the specific usability requirements of the users in that perspective.

The conceptual design phase details the implementation-independent aspects of the web site design, and consists of two sub-phases. The first sub-phase, object modeling, describes the information requirements expressed in the user class descriptions in conceptual object models. These *User Object Models* (UOM) describe the different types of objects, their relationships, and constraints on the relationships.

Navigational design is the second sub-phase of conceptual design. This phase involves the construction of a *conceptual navigational model*, which consists of a navigational path for each perspective identified. Navigational paths express how the users of a perspective can navigate through the information. During this stage components (information, navigation and external) and hyperlinks are constructed.

Implementation design fashions the actual "look and feel" of the web site, by focusing on implementation-dependent aspects of the web site design. Constraints including language limitations may influence design decisions taken during this stage. The result of this stage is an *implementation model*.

Implementation is the last stage of the WSDM development methodology, and involves the implementation of the site in the chosen environment. For example,

an HTML implementation may dictate that the implementation model be converted into a set of files containing HTML code (De Troyer and Leune, 1998). De Troyer and Leune (1998) state that automation could possibly be used in this stage, allowing information stored in a database to be automatically converted to the appropriate implementation environment.

While this method seeks to improve on existing development methods by providing a more focused approach specifically designed for web development, its overall construction is confusing. The authors include in their design phases activities which should be addressed in an analysis phase. These include the model's focus on the web site's set of potential users, which should be identified as an analysis issue, allowing subsequent phases to concentrate on design issues explicitly. The authors propose that the implementation stage fashions the actual "look and feel" of the web site, aiming "to create a consistent, pleasing and efficient" implementation design of the conceptual design. However, the method not only assumes that these are required characteristics of the web site, but also includes these design issues in the implementation phase. An improved solution would be to include such decisions in analysis and design phases, and only then determine whether the guidelines provided by the model are appropriate for a specific site.

WSDM's authors also fail to address a fundamental argument for its construction; namely that existing methods only address the maintenance issues of web site managers. In striving to correct this oversight, the model achieves its "user-centred" aim, however it fails to address the maintenance issues.

An evolutionary life-cycle approach is claimed to be supported, yet the method fails to provide a means for doing this. However as the model is in its infancy, such oversights can be expected to be corrected, and are indeed identified as areas for future research. By providing some of these suggestions, the method may be applicable to a variety of web sites, not just those user centred kiosk sites.

3.3.4.6 A Structured Methodology for Multimedia Product and Systems Development (SMMD)

Although developed for multimedia products, the Structured Methodology for Multimedia Products and Systems Development (SMMD) presented by Sherwood and Rout (1998) offers several important suggestions for web development. Using a product lifecycle based on rapid prototyping and iterative refinement, this methodology claims to draw on the "best practices" of software development to create a method that defines a multimedia process, providing a baseline for improved management of multimedia projects (Sherwood and Rout, 1998: 2).

The SMMD attempts to provide a structured framework for managing what has historically been defined as a 'cottage industry' (Sherwood and Rout, 1998: 2). The model proposes that multimedia development be viewed as a project. Using this approach, product development can be broken down into several stages, allowing increased management control of the individual phases and the overall process (Sherwood and Rout, 1998: 4).

Sherwood and Rout (1998: 6) suggest that multimedia products lend themselves to "development approaches based on collaborative analysis and design, iterative and rapid prototyping, small development teams comprised of specialists with

advanced tool sets, and project management based on prioritisation." Following this definition, the SMMD focuses on six stages of multimedia development (see Figure 7). These stages are divided into development, management, and support activities, each comprising several smaller activities.

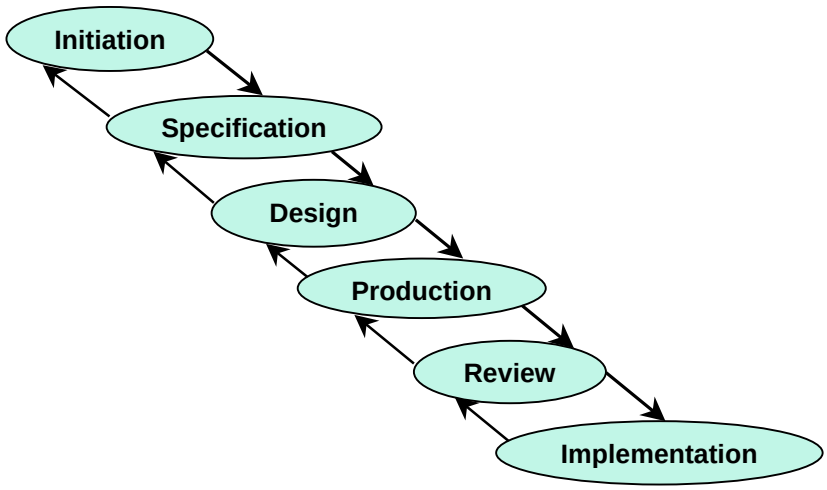

Figure 7. Overview of the SMMD Development Lifecycle (adapted from Sherwood and Rout, 1998: 5)

The initiation stage focuses on the planning required for the development of the product. Tasks including determining the overall development strategy, copyright costs and rights negotiations, and determination of the scope of risk management activities are performed during this stage. Configuration management, requirements solicitation, and a preliminary project plan are also outlined. The results from this stage are a feasibility study, scope and risk assessments, and a client acceptance criterion.

The specification stage details functional and performance requirements. Testing and usability criteria are established, and the feasibility of the project is reassessed. The results from this stage may include a quality agreement, test and evaluation plan, and a risk management plan. The major focus of the design stage

is development of the design document. This document outlines the human activity that the proposed system will support, identifies the users of the system, and selects the basic forms of the solution. The production stage deals with producing both the product's content, and the product itself. Any media acquisition negotiations should also be finalised.

Review and evaluation occur throughout the development process, being used to critically analyse the product before starting the next phase or iteration. Maintenance evaluation may also be conducted on online systems to examine their viability over time. The last stage of the methodology focuses on delivering the product to the client. Tasks involved in this stage include finalisation of client support and maintenance levels, signing of the acceptance criteria, and a review of the project performance.

3.3.4.7 Summary

The RMM methodology is by far the most detailed, and along with SMMD is the only method that includes a testing and evaluation phase. Indeed these methods are the only ones that consider the entire development process, concentrating on more than just design and implementation issues. However, in most other respects all methodologies are remarkably similar. They all include design and implementation phases, and all except SMMD include navigational and interface design as part of this.

In general the overall content of all the web design methodologies is sound. However, they all suffer from lack of an identified research approach in their construction. None of the (descriptions of the) methodologies attempt to define

the approach taken in their construction, apart from mentioning that they are based on other models from similar fields. While proof of their value may be determined by their use, proof of the soundness of their construction, and decisions taken during construction this is only evident through the strategy adopted in their construction.

3.3.5 Metrics

Metrics allow quantitative measurement. They are a way of giving objective clear and precise values to objects being studied (Botafogo *et al.*, 1992: 159). An often-used software size metric is the Line Of Code (LOC). The PSP framework uses LOC as a proxy for size, and being relatively simple to measure, can give a reasonably accurate and consistent representation of the amount of work required to develop the piece of software. However there is no equivalent simple web page size measurement.

Although many authors have addressed the issue of measuring hypertext products (Nielsen, 1992; Botafogo *et al.*, 1992; Bray, 1996; Garzotto *at al.*, 1995), most of this work concentrates on performance; structural issues including inter-connectedness, complexity, orderedness, and compactness; or navigational issues. In contrast, the quantitative measurements required for this framework are related to the actual size of the web page.

Botafogo *et al.* (1992: 159) consider this, identifying nodes and hyperlinks as the primary measures of size in a hypertext system. However, as the model presented in this thesis is limited in scope to web page development, node measures are not appropriate. Not considering nodes would suggest that measuring the number of

hyperlinks on a page would be a good measure of size. While hyperlink measurement is clearly one indication of size, the notion of size is really only a proxy for measuring intellectual effort, and the number of hyperlinks a page contains does not convey the entire picture.

Chapter 4

THE STRUCTURE OF THE MODEL

4.1 Defining a Framework

The Personal Web Process (PWP) improvement framework follows the same basic structure as the Personal Software Process (PSP). Starting from a defined and measured baseline process, developers learn to incorporate project and quality management techniques into their work through the use of process levels. Distinct and separate process levels have been incorporated into the framework to allow separation of quality management and project management concepts, maintain consistency with the PSP, and allow logical progression from an undisciplined, unrepeatable process, to a disciplined, repeatable process.

A process level is a stage within the improvement framework where a developer performs certain activities. For example, process level 0 identifies a baseline process. A developer using this process level for developing web pages would perform their usual development activities, combined with size and time measurement activities. A process level also includes several process elements, which describe the process to be performed, provide a convenient storage facility for process data, support planning activities, and help analysis. The process elements of the PWP include forms and scripts.

This chapter presents the Personal Web Process (PWP), a process improvement model for personal web page development. This section describes several issues pertinent to developing a model for web page development, including process elements, implementation languages, the web page development process, and omissions. Section 4.2 presents the complete process improvement model, detailing process levels, process elements, and activities. A complete record of the forms and scripts used in this framework can be found in Appendices A through D.

4.1.1 Process Elements

Humphrey (1995: 442) identifies four essential items that should be included in a personal process. These are scripts, forms, standards, and provisions for process improvement.

Process scripts describe the execution of the process, and refer the developer to relevant standards, forms, guidelines, and measures (Humphrey, 1995: 442). Scripts consist of a set of entry criteria, process phases, and a set of exit criteria. Entry criteria explicitly state requirements that must be satisfied before starting the process described by the process script. Similarly, exit criteria explicitly state requirements that will be satisfied after process described by the process script is complete. Process phases describe the essential steps to be performed. While steps should usually be followed in the order in which they are described, there may be some instances where steps may be need to be repeated or performed simultaneously.

Forms provide a convenient and consistent framework for the collection and collation of relevant process measurements. Examples of data collection forms used in the PWP framework include time and defect recording logs, and the project plan and summary. Figure 8 shows how process scripts and forms work together during the development process.

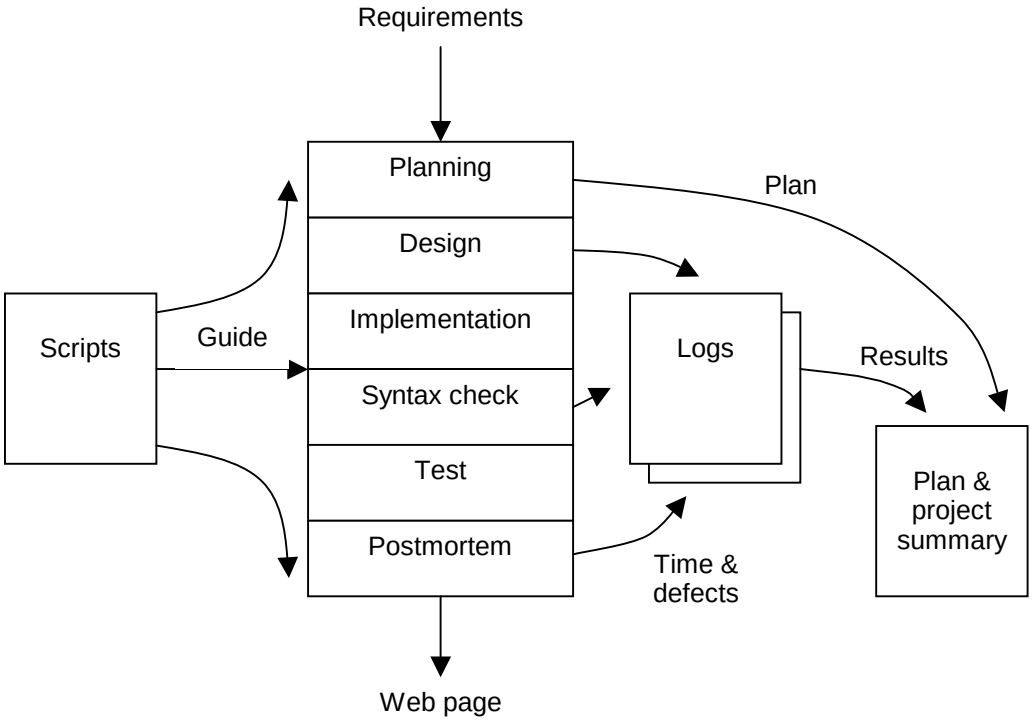

Figure 8. PWP Process Elements and Flows (adapted from Humphrey, 1995: 31)

Standards provide a basis for verification of product and process quality (Humphrey, 1995: 444). The PWP identifies a syntax standard and a defect type standard. The syntax standard guides the writing of web page source code. Use of the syntax standard allows writing source code that is easy to read, comprehend, modify, and review. The defect type standard defined in Table A11 provides an example catalogue of commonly occurring web page development defects, and is used in conjunction with the defect log to simplify the classification of defect types.

The process improvement provision supplied in the PWP is the Process Improvement Proposal (PIP). This form is introduced in PWP level 0.1, and is mainly used to report process comments, problems, and observations, which may be used to evolve the process.

4.1.2 Implementation Language

Historically, web page development has been performed using HyperText Markup Language (HTML) as the implementation language. A subset of the Standard Generalised Markup Language (SGML), HTML was originally developed by Tim Berners-Lee to allow access to structured documents in a networked environment (W3C, 1998a). With the recent surge of interest in the web, standardisation initiatives have resulted in the creation of several HTML standards, the latest being HTML 4.0 (W3C, 1998a). In response to recognised limitations with HTML architecture, the Extensible Markup Language (XML) was developed. XML – a larger subset of SGML than HTML – is a meta-language for describing document types, and custom markup languages (W3C, 1998b). Other languages commonly used in web page construction include DHTML, Cold Fusion, Lingo, Java (Gosling *et al.*, 1996), JavaScript (Netscape Communications Corporation, 1998), Perl, CGI, ActiveX, and other scripting languages.

As the PWP is a framework for improving the web development process, care has been taken to reduce its focus on specific technologies. Therefore, although the majority of web page development occurs using HTML, the PWP framework is generic enough to support any implementation language and technology. However if necessary, the concepts and techniques presented in the framework can be

adapted and modified to suit a particular development environment, such as web pages developed in HTML using a non-graphical editor.

4.1.3 The Personal Web Page Development Process

As web page development is a new and rapidly growing field, there are very few documented processes (Lowe and Webby, 1998: 3). Although the development methodologies defined provide some notion of a development process, the process is defined implicitly by the methodology itself (Lowe and Webby, 1998: 3). However, even if we consider this as a methodological flaw, combining the phases of the methodologies that do exist for web page development provides a usable set of development phases for use within the scope of this framework. Derived through an analysis of the hypermedia literature, Table 3 provides a summary of the development phases of several well-known hypermedia development methodologies described in Section 3.3.4.

Methodology	Development Phases
OOHDM	• Domain analysis • Conceptual design • Navigational design • Abstract interface design • Implementation
RMM	• Feasibility study • Information/navigation requirements analysis • Entity-relation design • Entity design • Navigational design • Conversion protocol design • User interface screen design • Run-time behaviour design • Construction • Testing and evaluation
WSDM	• User modeling • Conceptual design • Implementation design • Implementation
SMMD	• Initiation • Specification • Design • Production • Review and evaluation • Delivery and implementation

Table 3. The Development Phases of Several Hypermedia Development Methodologies[4]

Using these development phases as a basis, the development phases identified for use in the PWP framework are planning, navigational design, interface design,

[4] Derived from Schwabe *et al.*, 1996; Schwabe and Rossi, 1995; Isakowitz *et al.*, 1995; De Troyer and Leune, 1998; Sherwood and Rout, 1998.

design review, implementation, syntax review, syntax check, testing, and postmortem. These phases have been identified on the grounds of their importance to web page development and the framework, and use in majority of existing development methodologies. Table 4 describes the each of these development phases, the PWP process level in which it is introduced; and its major activities. For the purposes of the PWP, development activities relating to the construction or maintenance of one or more web pages are labeled as projects. The remainder of this section explains each development phase used in the framework in detail, outlining each step of the phase.

Phase name	**Level Introduced**	**Description**
Planning	PWP 0	Plan what you are going to do. • Requirements; • Time and resources estimation; • Task and schedule planning.
Navigational design	PWP 2	Design the navigational (hyperlink) structure of the page. • Design the navigational structure of the web page; • Document the design in the Navigational Design Template; • Ensure that the design meets the requirements; • Fix and log all defects found.
Interface design	PWP 2	Design the interface layout. • Design the screen layout of the web page; • Document the design in the Interface Design Template; • Fix and log all defects found.

Design review	PWP 2	Review the design documentation for defects. • Navigational design review; • Interface design review; • Fix and log all defects found.
Implementation	PWP 0	Implement the design in the required language. • Implement the design; • Fix and log all defects found.
Syntax Review	PWP 2	Review the web page source code for defects. • Syntax review; • Fix and log all defects found.
Syntax check	PWP 0	Validate the implementation language syntax. • Syntax validation; • Fix and log all defects found.
Testing	PWP 0	Load the page into a browser, test all hyperlinks, and ensure correct functionality. • Load page into browser; • Check visual display; • Test hyperlinks; • Test other functionality; • Fix and log all defects found.
Postmortem	PWP 0	Complete data collection and process improvement forms. • Complete project plan and summary form with project data; • Fill out process improvement proposal forms; • Complete miscellaneous data collection forms; • Fix and log all defects found.

Table 4. PWP Development Phases, Descriptions and Key Tasks

4.1.3.1 Planning

The Planning phase focuses on planning the development of the web page. Effective plans are vital for effectively managing development, meeting commitments, and tracking progress (Humphrey, 1995: 59). The planning scripts

defined in Tables A2, B2, C2, and D2, detail each step of the planning phase. For the highest process level, these steps are: document requirements, produce a conceptual design, estimate size, estimate time, estimate defects, and plan tasks and schedule. Data collection during the planning stage is made using a Project Plan and Summary form.

4.1.3.2 Navigational Design

The Navigation Design phase involves designing the navigational structure of the web page. This structure is described by navigational paths, which express how users can navigate within the web page, and between web pages. Existing development methodologies cite customised user navigational structures as one of the most important elements of navigational design (Schwabe *et al.*, 1996: 3; De Troyer and Leune, 1998: 8). However due to the PWP's focus on the individual web page developer, customised user navigation is an analysis issue, and is hence not relevant within the scope of this framework. While this issue is not relevant within the scope of the framework, encouragement is given to still include it as part of the web page requirements.

The navigational design script defined in Table D11, details each step of the navigational design phase. For PWP level 2, these steps are: construct a navigational design and ensure that the design meets the requirements. A navigational design is documented in the Navigational Design Template presented in Table D10, and discussed in Section 4.2.3.3

4.1.3.3 Interface Design

The Interface Design stage involves designing the interface of the web page. This stage takes the navigational design produced in the previous stage and makes it perceptible to the user of the web page through the interface (Schwabe *et al.*, 1996: 5). Separating the navigational design from the interface design allows different interfaces to be constructed using the same navigational model (Schwabe *et al.*, 1996: 5). This is of benefit if the web page is to be used by users with different abilities or backgrounds, and decreases reliance on specific user-interface technology (Schwabe *et al.*, 1996: 5).

An effective design should describe the screen layout for everything that is to appear on the screen. This includes images, formatting structures such as tables and frames, location of hyperlinks and navigational objects, embedded objects such as sounds, form fields, and scripting objects. The interface design script defined in Table D13, details each step of the interface design phase. For PWP level 2, these steps are: construct an interface design and ensure that the design meets the requirements. An interface design is documented in the Interface Design Template presented in Table D12, and discussed in Section 4.2.3.4

4.1.3.4 Design Review

Introduced in PWP process level 2, the design review phase involves reviewing the designs developed as part of the navigational and interface design phases for defects and conformance to requirements. This stage is a vital activity in ensuring that defects do not propagate further through the development process. The development script defined in Table D3 details each step of the design review phase.

4.1.3.5 Implementation

The Implementation stage maps the navigational and interface designs into the chosen implementation environment. That is, the designs produced in the preceding two stages are used to construct a web page in an appropriate or required language. For example, an HTML implementation requires that the designs be mapped to a file or set of files containing HTML source code. Web page files can be constructed using text editors, specialised text editors, and graphical editors. A discussion on issues relevant to this framework arising from using graphical web page editors is presented in Section 4.2.1.2. The development process scripts defined in Tables A3, B3, C3, and D3, detail each step of the implementation phase.

4.1.3.6 Syntax Review

Introduced in PWP process level 2, the syntax review phase involves manually reviewing the implementation language source code for defects. Defects that would usually be missed by the syntax validator, can often be found by performing a manual review of the web page file. The development script defined in Table D3 details each step of the syntax review phase.

4.1.3.7 Syntax Check

The Syntax and Graphical Check stage involves checking the web page source code syntax for errors. Syntax checking is performed automatically, with the use of language validation software. The development scripts defined in Tables A3, B3, C3, and D3, detail each step of the syntax check phase.

4.1.3.8 Testing

The Testing phase involves testing of the web page in a web page browser to check for defects. The components of testing are visual testing, hyperlink testing, and general functionality testing. Visual testing involves ensuring all graphical elements are in the correct position, are the correct size, are the correct colour, and that the display corresponds with the interface design. Hyperlink testing involves thoroughly testing all navigational paths, ensuring that they link to the correct place (Isakowitz, 1995; 43). Functionality testing involves testing the general functionality of the web page, ensuring that graphical objects behave correctly, and other objects such as scripts and form fields function correctly. The development scripts defined in Tables A3, B3, C3, and D3, detail each step of the testing phase. These steps are: load the web page into a browser, check the visual layout of the web page, test all hyperlinks, and ensure the correct functionality of other objects such as scripts and form fields.

To log the results of tests for later referral, PWP level 1 introduces test reports (defined in Table C8). These forms describe the test, its objectives, the planned results, and the actual results obtained.

4.1.3.9 Postmortem

The Postmortem phase focuses on reviewing the development, collecting data, collating data, and analysing data. This stage is particularly important as data collected throughout development can be used to improve the planning process (Humphrey, 1995: 35). The major task performed during this stage is completion of the project plan and summary form. This involves collating and recording defect data, measuring and recording size, and collating and recording

development time. The development scripts defined in Tables A3, B3, C3, and D3, detail each step of the postmortem phase. For the highest process level these steps are: document requirements, produce a conceptual design, estimate size, estimate time, estimate defects, and plan tasks and schedule.

4.1.4 Omissions

The review of the web page development literature presented in Section 3.3.3 identifies several issues that are important to web page development. These issues are web site management, project feasibility studies, user analysis, user interface design, design guidelines, hyperlink management, metadata, and configuration management. While all these issues are important when considering the entire web development process, as the scope of this framework is limited to individual developers and single web pages, some issues have been omitted from the framework.

Project feasibility studies and user analysis form part of an analysis phase. As it is anticipated that users of the PWP framework will not be in the position to perform any form of analysis relating to the web pages that they are developing, such information is assumed to be already known, forming part of the requirements of the web page. Web site management, hyperlink management, and configuration management are also not considered due to the limited scope of the framework. However, the use of defined design templates, design guidelines and standard metadata elements have been included in the framework, to provide a quality management process level.

4.2 The Framework

The PWP improvement framework follows the same basic structure as the PSP. Starting from a defined and measured baseline process, developers learn to incorporate project and quality management techniques into their work. The PWP framework consists of three process levels: the baseline personal process, personal project management, and personal quality management. Within each of these stages, the developer is required to perform various tasks. For example, level 0 requires the developer to record size and time measures, and develop an implementation language standard. Figure 9 shows the PWP process levels and their constituent activities. The remainder of this section discusses these.

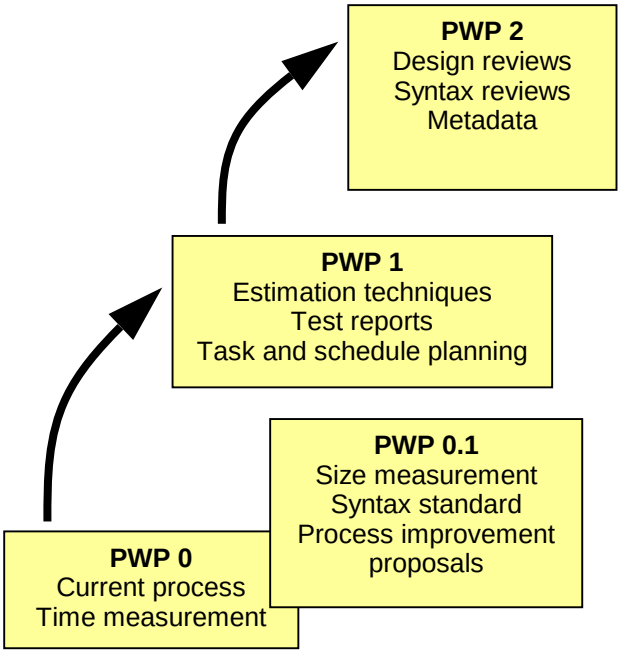

Figure 9. PWP Process Levels

4.2.1 Level 0 – The Baseline Personal Process

The first step in the PWP framework is to develop a baseline process, from which subsequent improvements can be measured. The principle objective of this stage is to provide the framework necessary to collect data on the initial development

process (Humphrey, 1995: 30). This involves taking measurements of project size and development time, to develop a picture of the current development process. The PWP 0 process level is divided into two sub-levels. PWP 0 identifies the current state of the development process within a defined framework, while PWP 0.1 introduces further methods involved in the establishment of a baseline process description. To initiate the development of a defined process, developers perform their usual development process, divided into six defined stages.

4.2.1.1 The PWP 0 and PWP 0.1 Process

The PWP level 0 and 0.1 processes (PWP 0 and PWP 0.1) consist of three major phases, shown in Figure 10. The Planning phase focuses on planning the development of the web page, documenting project requirements and time estimation. The Development phase is where the web page is actually constructed, and is broken down into three sub-phases. The design sub-phase involves designing a web page to meet the requirements. The implementation sub-phase involves implementing the web page in the required or appropriate implementation language. The syntax check sub-phase involves validating the syntax of the implementation language source code. The test sub-phase involves ensuring correct visual layout, and testing the functionality of hyperlinks and other objects. The last phase of the PWP 0 and PWP 0.1 development processes is the Postmortem phase. This phase involves completing the planning forms with size, time and defect data.

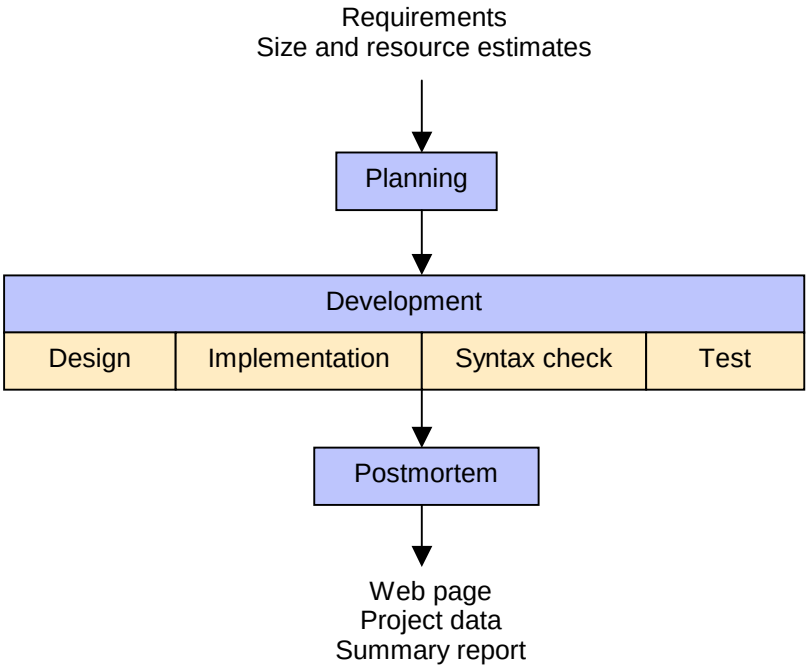

Figure 10. The PWP Level 0 and Level 1 Development Process
(adapted from Humphrey, 1995: 34)

4.2.1.2 PWP 0 and PWP 0.1 Measures

Resource Measurement

The amount of resources committed to a development project is often measured in monetary terms. The record of expenditure on development projects can be used in future planning activities including resource allocation, commitment making, task and schedule development, and general decision making. However, as this framework pertains to the individual web page developer, resource commitment measured directly in terms of monetary value is not appropriate.

The resource expenditure measure used in the PWP framework is time. Measuring development time gives the developer information on time distribution, allowing further analysis and decision making (Humphrey, 1995: 38). This information could be used to reassign time to problem areas, improve the understanding of the

actual time distribution of the developer, and allow further complex analysis of data. Although time measurements could be taken in arbitrary units, to allow future analysis, and to provide an appropriate trade-off between fine- and course-grained units, time in the PWP framework is measured in minutes.

Size Measurement

Visually, web pages consist of static (passive) entities such as hyperlinks, text, images, and formatting structures; and active (dynamic) entities such as form fields, script objects, and embedded objects (Garzotto, 1995: 74). Some of these elements are shown in Figure 11. Hyperlinks are navigational constructs that allow direct intra- and inter-document access (Andrews, 1996: 1). Information is usually presented in the form of text, whether in paragraphs, bulleted or numbered lists, or within formatting structures such as frames and tables. Images can be used to convey information, provide aesthetic appeal, and provide functionality. Form fields and script objects often provide interaction between the user and the web page, and can also be used for information storage. Several newer web browsers also include the capability to handle objects such as sound and video embedded in a web page.

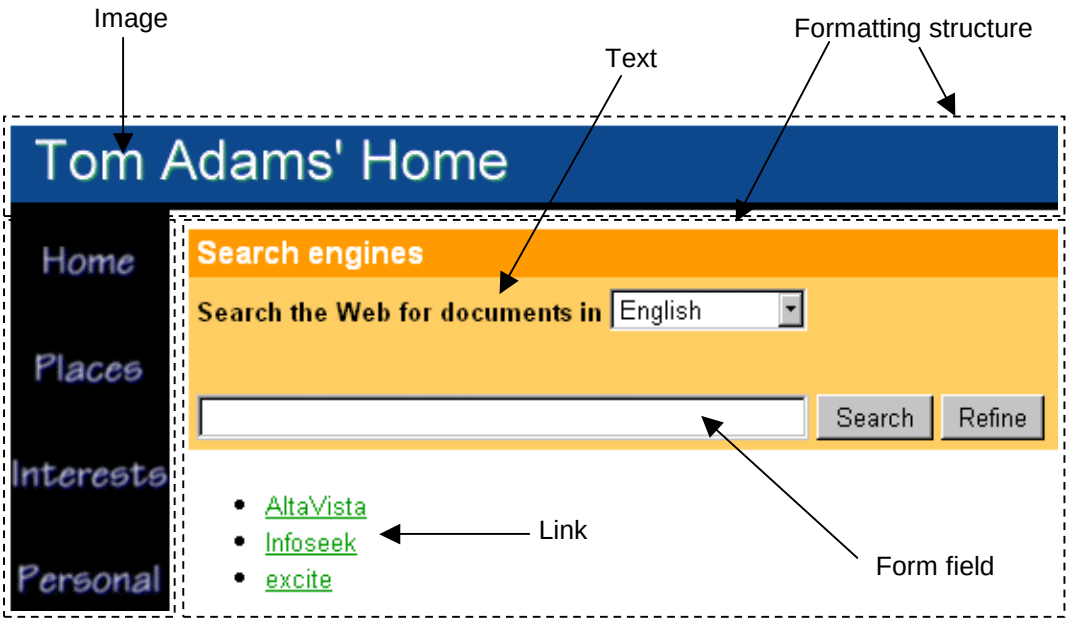

Figure 11. Visual Web Page Components

Although these objects appear visually when viewed using a web browser, the underlying source code contains element declarations only. Elements are descriptive formatting codes used in web pages to represent structure or desired behaviour (W3C, 1998a). Element declarations usually consist of three parts: a start tag, content, and an end tag (W3C, 1998a). Most element types allow the definition of property fields called attributes, having one or more values (W3C, 1998a). An example of these for the HTML element type <P> is shown in Figure 12.

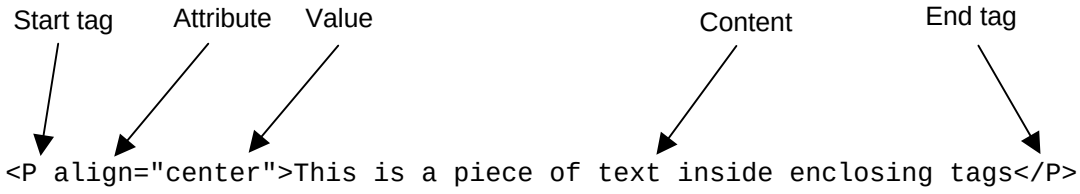

Figure 12. Element Components

As discussed in Section 3.3.5, capturing only one metric for web page size does not adequately reflect the amount of intellectual effort required in developing a web page. As web pages have both visual and source dimensions, considering

only the graphical attributes of a page may not reflect the amount of work invested in actually typing the source code, choosing appropriate elements, checking syntax, and managing hyperlinks. Likewise, measuring source code fails to take into account the amount of effort positioning, sizing, and structuring information and images. Therefore, any metric attempting to capture size should incorporate both visual and source dimensions.

However, counting both visual entities and source code elements may not provide orthogonal measures. Orthogonal measurement is essential to provide an independent and accurate representation of the entity being measured. Although the discussion presented above highlights the need to consider both dimensions in a size measure, actually counting both may not be appropriate. One way to solve these problems would be to include an portion of both metrics in the one size measure. Such a metric could be based on only one dimension of size, however include in any measurement a factor representing the other dimension. Therefore, counting only one dimension would simplify the counting procedure and possibly provide orthogonal measurement, and including the other dimension would satisfy the need to include both dimensions in a measurement of web page size.

Considering these issues, the PWP framework uses the Visual Page Component (VPC) as a size metric. A VPC represents a visual web page object. These objects include images, text paragraphs, hyperlinks, active entities, and formatting structures. The VPC satisfies the criteria of including both dimensions of size, and is also simple to measure. Another advantage of the VPC is that it supports the use of visual development tools. As each of these entities also contains a

corresponding source code element, the VPC count for each of these objects must include a specific contribution factor, accounting for the effort required to develop and manage the source code element. For example, developing a table may require more work with source code than developing an image. Therefore, the contributing factor of the source code element to the VPC count for a table would be proportionally larger than that for an image.

Although the VPC incorporates the source code dimension of web page size, the addition factor this dimension brings to the measurement is not defined within this thesis. Such a determination would be a task for future research, and possibly for individual analysis and contemplation. Web page developers using the framework are therefore encouraged to define an appropriate contributing factor that accounts for the amount of effort required for developing and managing source code. A contribution factor of zero means that developing and managing the source code element required no effort, and a contribution factor of one means that developing and managing the source code element required a large amount of effort. For example, if a visual web page editor were used in development, the contributing factor would be zero as such editors automatically generate source code. However, ignoring this factor altogether may help in obtaining an initial estimate.

Using these definitions, measuring size becomes a simple matter of counting. The PWP method for measuring size is described in Table 5.

Metric	Counting Method
VPC	Count one VPC for each: • Image; • Formatting structure, eg. Frame, table; • Hyperlink; • Form field; • Script object, embedded object; • Paragraph or part thereof. For each specific visual object: • Add a contributing factor between one and zero that accounts for the effort required to develop and manage the source code element. Zero meaning that developing and managing the source code element required no effort, and one meaning that developing and managing the source code element required a large amount of effort.

Table 5. PWP Size Measurement Method

For use within a framework such as this, Humphrey (1995: 69) considers a size metric to be useful only if it satisfies three criteria. These are usefulness for planning, precision, and automated counting. VPCs are satisfy this set of criteria as they are easily measurable, provide an accurate representation of the entity being measured, can be precisely defined, and can be counted automatically. Two other issues that are also important are repeatability and consistency. If a web page was counted one day, and counted again in three days time, the same size measurement should be obtained. Also, if two people were to count a web page, they should both get the same or very similar measurements. Counting visual page components is clearly produces a measure that is repeatable and consistent. Counting the number of visual components on a web page on different days would give the same measurement. Also, if an appropriate counting standard is used, different people counting the same web page will get exactly the same answers.

4.2.1.3 Project Plan and Summary

The project plan and summary is the principal planning form of the PWP level 0 and level 0.1 process. It records the planning information needed for analysis in the postmortem phase, and for future reference. An excerpt from the PWP 0 project plan and summary form is shown in Figure 13. The top section of the form identifies the developer of the web page, the date, the title of the web page, the name of the project, and the implementation language used. The remaining sections detail size (for PWP 0.1), time, and defect estimates and actual measures. To aid in planning future projects, a breakdown percentage per development phase is also given.

TABLE A5 PWP 0 PROJECT PLAN AND SUMMARY

Developer _____ Date _____
Page title _____ Project _____
Language _____

Time in Phase (min.)		**Plan**	**Actual**	**To Date**	**To Date %**
Planning	PLAN				
Design	DSGN				
Implementation	IMPL				
Syntax check	SNTX				
Test	TEST				
Postmortem	POST				
Total					

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Implementation			
Syntax check			
Test			
Total Development			

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Implementation			
Syntax check			
Test			
Total Development			
After Development			

Figure 13. PWP 0 Project Plan and Summary Excerpt

PWP 0.1 introduces size and time tracking to the plan and summary form. As identified in Section 4.2.1.2, size measurements are made using the visual page component (VPC) metric. The plan and summary form records planned size, actual size, and to date size. Planned size is the size estimated during the planning phase. As PWP level 0 does not contain an estimation procedure this figure should be calculated using an appropriate mechanism. Actual size is the size of the web page measured after development in the postmortem phase. This figure is calculated using the size counting technique described in Section 4.2.1.2. The to date size is the summation of all size measurements from previous projects from the particular size category. For example, the to date Total VPCs field would contain the summation of the total amount of VPCs developed from all projects. The full PWP level 0 and level 0.1 project plan and summary forms are presented in Tables A5 and B5 respectively.

4.2.1.4 Process Improvement Proposal (PIP)

The process improvement proposal (PIP) form is introduced in PWP level 0.1. The PIP form is used mainly to report problems, comments, and observations on the process. An excerpt from the PIP form is given in Figure 14. The top section of the PIP form identifies the developer of the web page, the date, the name of the project, the process level the developer is working at, and the specific element of the process that the problem is related to. The remaining sections document a description of the problem, a proposal for overcoming the problem encountered, and any comments related to the problem or solution. The PIP form is presented in Table B7.

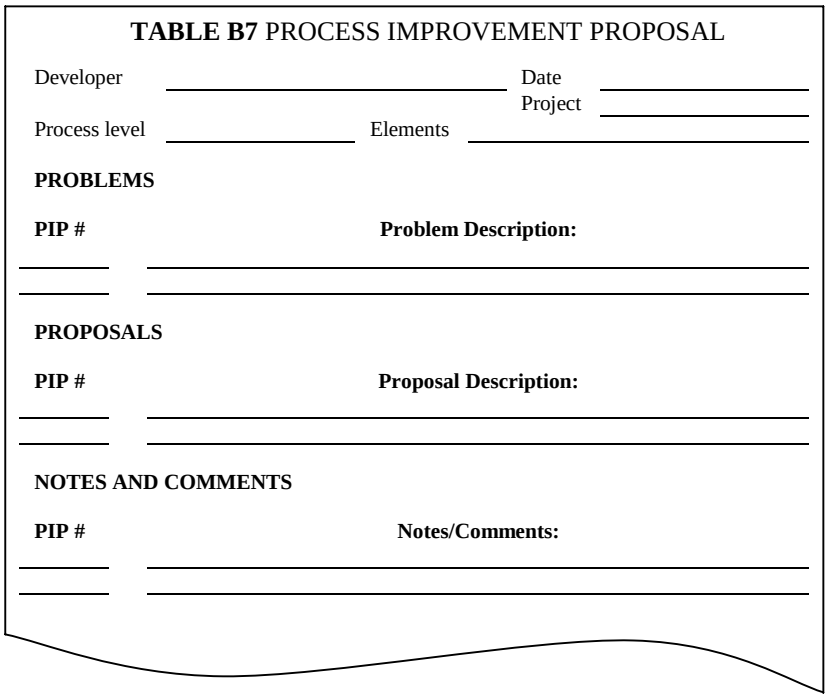

TABLE B7 PROCESS IMPROVEMENT PROPOSAL

Developer _____ Date _____

Project _____

Process level _____ Elements _____

PROBLEMS

PIP # **Problem Description:**

PROPOSALS

PIP # **Proposal Description:**

NOTES AND COMMENTS

PIP # **Notes/Comments:**

Figure 14. Process Improvement Proposal (PIP) Excerpt

4.2.1.5 Time Recording Log

The time recording log is a convenient form for keeping track of time spent in each development phase. During the postmortem phase, this information is collated and used to complete the project plan and summary form. A time recording log excerpt is given in Figure 15. The heading section of the form identifies the developer of the web page, the date, the title of the web page, the name of the project. The main section outlines the date the development activity took place, its start and finish times, the amount of time spent handling interruptions, the total time taken to complete this task, the phase of development in which the task took place, and any comments that may be pertinent to the entry.

TABLE A7 TIME RECORDING LOG

Developer _____ Date _____
Page title _____ Project _____

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments

Figure 15. Time Recording Log Excerpt

4.2.1.6 Defect Recording Log

The defect recording log is a convenient form for recording defects as they occur. Defects are a result of an error or mistake made by the developer. Defects should be recorded as they occur, rather than left until the postmortem phase. An excerpt from the defect recording log presented in Table A9 is given in Figure 16.

TABLE A9 DEFECT RECORDING LOG

Developer _____ Date _____
Page title _____ Project _____

Date Number Type Inject Remove Fix Time Fix Defect

Description: _____

Date Number Type Inject Remove Fix Time Fix Defect

Description: _____

Figure 16. Defect Recording Log Excerpt

The heading section of the form identifies the developer of the web page, the date, the title of the web page, the name of the project. The remaining sections pertain to the actual defect itself. The date the defect occurred is placed in the date field, along with a number that uniquely identifies the defect. The defect type is

recorded as a numerical code obtained from the defect type standard (see Table 6). An educated guess should be made to determine which category applies. While the actual category chosen is important, it is more important to be consistent in categorisation. The development phase in which the defect was injected and removed is also recorded. The time taken to fix the defect is recorded in the Fix Time section. Fix times should be recorded to the nearest minute. If the defect occurred while trying to fix another defect, the number of the original defect should be noted in the Fix Defect section. The description of the defect should be clear, and sufficient to allow later analysis.

4.2.1.7 Defect Type Standard

The defect type standard presents nine commonly occurring web page development defects. These are Documentation, Syntax, Link, Window, Graphic, Tag, Form, Build and Package, and Environment. The defect type standard is presented in Table 6 and Table A11. While these categories should sufficient to cover the entire scope of web page development defects, post-analysis customisation of categories may need to occur.

Type Number	Type Name	Description
10	Documentation	Comments, messages.
20	Syntax	Spelling, punctuation, grammar, typos, text alignment, table alignment, text formatting.
30	Link	Incorrect link, dead link.
40	Window	Frame display, new window.
50	Graphic	Alignment, size, file format, image colours.
60	Tag	General tag format or problems.
70	Form	General form problems.
80	Build, Package	Change management, version control.
90	Environment	Design, test, or other support system problems.

Table 6. Defect Type Standard

4.2.1.8 Syntax Standard

The syntax standard details the criteria against which the syntax and format of the source code can be evaluated. A syntax standard can be used to write legible source code, improve readability and maintainability, and provide correct structure (Humphrey, 1995: 614). The PWP framework does not define an explicit syntax standard, preferring instead to let individual developers define their own. This approach provides maximum customisation, allowing developer- and implementation language-specific issues to be better addressed than they would with a generic standard. However, all syntax standards should outline comment syntax, header formats, reuse instructions, indentation format, capitalisation usage, and white-space format.

4.2.2 Level 1 – Personal Project Management

Building on PWP level 0, PWP level 1 (PWP 1) introduces project management and planning steps to the framework. The principle objective of this stage is to

introduce project management activities, which provide an improved planning and overall management capacity. The activities introduced in PWP 1 are estimation techniques, task and schedule planning, and test reports.

4.2.2.1 The PWP 1 Process

The PWP 1 process is essentially the same as the PWP 0 process (see Figure 10). Although the development phases do not change, two new activities are introduced into the planning phase, and one new activity into the test phase.

4.2.2.2 Project Plan and Summary

The PWP 1 project plan and summary form introduces a summary section to the standard planning form. This section provides an easily accessible record of productivity and planning information. The data recorded in the summary section includes the planned, actual, and to date amount of VPCs produced per hour, the planned and actual total development time, the cost-performance index, the percentage of VPCs reused, and the percentage of VPCs planned for reuse. An excerpt from the PWP 1 plan and summary form is shown in Figure 17.

TABLE C5 PWP 1 PROJECT PLAN AND SUMMARY

Developer _____ Date _____
Page title _____ Project _____
Language _____

Summary	***Plan***	***Actual***	***To Date***
VPCs/Hour			
Actual Time			
Planned Time			
CPI(Cost-Performance Index)			(Planned/Actual)
% Reused			
% Planned for Reuse			

Figure 17. PWP 1 Project Plan and Summary Excerpt

4.2.2.3 Estimation Techniques

The estimation technique used within the PWP framework is based on historic data collected as part of the planning and postmortem development phases. This approach is similar to the Proxy Based Estimation Method (PROBE) defined in the PSP (see Humphrey, 1995: 109). Using the size estimating template defined in Figure 18 and Table C10, an initial estimate is adjusted using multiple regression analysis of historical data. This method is described in the Size Estimating Template Instructions, and the Estimating Script defined in Tables C7 and C11 respectively.

TABLE C10 SIZE ESTIMATING TEMPLATE

Developer _____ Date _____
Project _____

BASE WEB PAGE	VPCs
BASE SIZE (B) => => => => => => => => => =>	
VPCs DELETED (D) => => => => => => => => =>	
VPCs MODIFIED (M) => => => => => => => => =>	
PROJECTED VPCs (P)	
BASE ADDITIONS:	VPCs
TOTAL BASE ADDITIONS (BA) → → → → →	
NEW COMPONENTS:	VPCs (Planned for Reuse*)
TOTAL NEW VPCs (NV) → → → → → → → →	
REUSED WEB PAGES	VPCs
REUSED TOTAL (R) → → → → → → → →	

Figure 18. Size Estimating Template Excerpt

4.2.2.4 Task and Schedule Planning

A schedule details how time is allocated to task completion. To aid in this activity, PWP 1 introduces task and schedule planning templates, shown in Figure 19 and Figure 20 respectively. Task and schedule planning is just as important as accurate size and resource estimation (Humphrey, 1995: 168). Even if the estimate of total time or resources is close to perfect, incorrect assumptions about how this time is to be spent could result in schedule overruns and failure to meet commitments (Humphrey, 1995: 168).

The Task Planning Template details development tasks requiring completion. These tasks could be as simple as assigning a task for each development phase, or as complex as identifying each step of a particular stage of development. Tasks are identified with a name, planned completion time, and the value of this task in terms of the total development time. After tasks are completed, the completion date, and the actual value of this task are recorded. The exact usage of the Task Planning Template is described n Table C13.

TABLE C12 TASK PLANNING TEMPLATE

Developer _____ Date _____
Project _____

Task		Plan					Actual		
#	Name	Time	Planned Value	Cumulative Time	Cumulative Planned Value	Planned Date	Date Done	Earned Value	Cumulative Earned Value

Figure 19. Task Planning Template Excerpt

The schedule planning template details the amount of time that can be committed to development tasks. Ordered sequential by date, this form records the day-by-

day time planned and actually spent on development tasks. The exact usage of the Schedule Planning Template is described n Table C15.

TABLE C14 SCHEDULE PLANNING TEMPLATE

Developer _____ Date _____
Project _____

	Plan			Actual			
Date	**Direct Time**	**Cumulative Time**	**Cumulative Planned Value**	**Direct Time**	**Cumulative Time**	**Cumulative Earned Value**	**Adjusted Earned Value**

Figure 20. Schedule Planning Template Excerpt

4.2.2.5 Test Reports

TABLE C8 TEST REPORT TEMPLATE

Developer _____ Date _____
Project _____

Test Name/Number	
Test Objective	
Test Description	
Test Conditions	
Expected Results	
Actual Results	

Figure 21. Test Report Template Excerpt

Test reports provide a way of recording the planned and actual results of tests carried out on the web page. Keeping a record of tests performed can often help future modifications, by ensuring that changes do not cause problems with previously working sections (Humphrey, 1995: 620). Test reports can often be

completed in advance as part of the design process. Doing this also helps ensure that test reports are completed when tests are actually performed. An excerpt from the Test Report Template of Table C8 is shown in Figure 21.

4.2.3 Level 2 – Personal Quality Management

The PWP 2 process builds on level 1, incorporating quality management issues into the development process. The principle objective of this stage is to introduce quality management activities into the development process, attempting to improve the overall quality of web pages. The activities, forms, and concepts introduced in PWP 2 are navigational and interface design phases and templates, design and syntax reviews, and metadata.

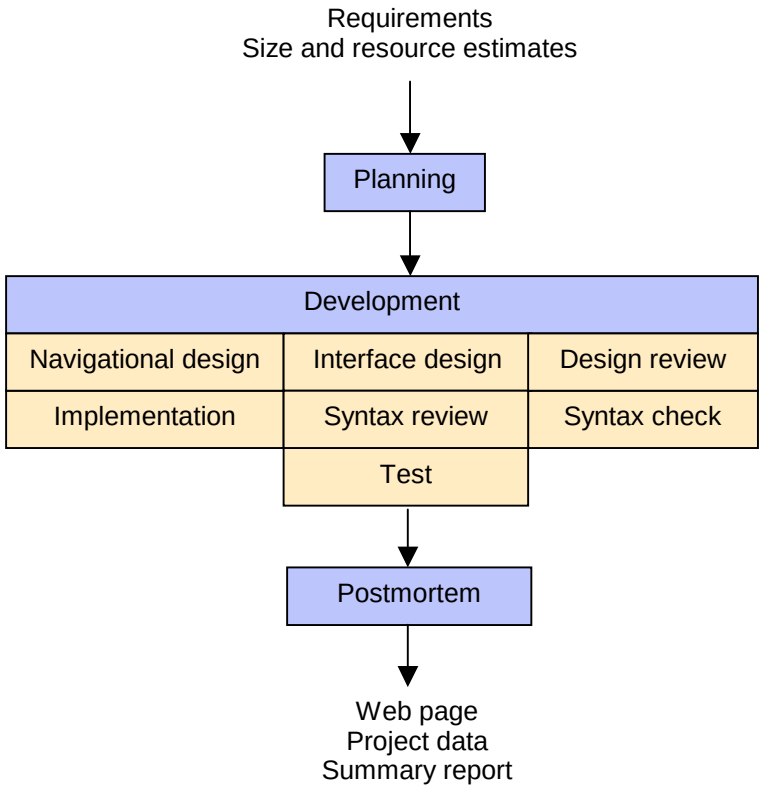

Figure 22. The PWP Level 2 Development Process (adapted from Humphrey, 1995: 34)

4.2.3.1 The PWP 2 Process

PWP 2 introduces new phases into the development process. These phases are navigational design, interface design, design review, and syntax review (see Section 4.1.3 for a discussion of these phases). The PWP 2 development process is shown in Figure 22.

4.2.3.2 Project Plan and Summary

The PWP 2 Project Plan and Summary form introduces several derived quality metrics. These include the number of defects per thousand visual page components (KVPCs), defect removal yield, and cost of quality. The number of defects per KVPC can be used to plan the amount of defects expected to be injected during development. Defect removal yield can be used to assess the effectiveness of design and syntax reviews, while appraisal and failure cost of quality can be used to determine identify and analyse the cost of defects injected in different phases. The PWP 2 Project Plan and Summary form is presented in Table D5.

4.2.3.3 Navigational Design Template

To aid in the construction of a navigational design, a Navigational Design Template is defined and described in Tables D10 and D11. This template should be used to describe the navigational structure of the web page. A navigational design should completely describe hyperlinks, navigation components, information components, and external components. The diagrammatic representation of these components is shown in Figure 23.

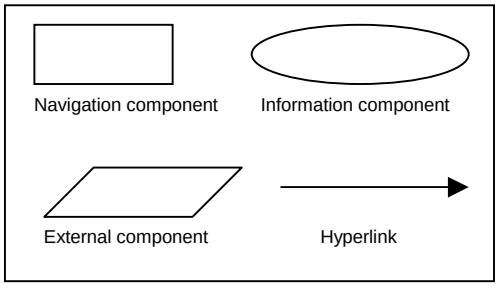

Figure 23. Graphical Representation of Navigational Components
(adapted from De Troyer and Leune, 1998: 8)

4.2.3.4 Interface Design Template

To aid in the construction of an interface design, an Interface Design Template is defined and described in Tables D12 and D13 (in Appendix D). This template should be used to design the interface of the web page. An interface design should include all hyperlinks, frames, tables, lists, text paragraphs, graphics, and other graphical components. Although an explicit format for representing these entities is not defined, any graphical notation representing these components should be clear, look like the actual component, and be distinct.

4.2.3.5 Metadata

As discussed in Section 3.3.3.4, metadata is a way of including descriptive information about a web page in its source code. The PWP 2 process identifies the use of metadata in the implementation phase. In PWP 2, the Syntax Standard should also be updated, to acknowledge the use of metadata elements.

Chapter 5

SUMMARY AND CONCLUSIONS

5.1 Summary

With the World Wide Web growing at a rapid pace, technological and competitive forces are influencing an increase in the size and complexity of web pages and sites. Although development methodologies exist which guide the web page development process, there is little research into the process itself, and no research into improving the process. Considering this lack of research, this thesis asks the question: *How can the web page development process be improved?* Acknowledging that process improvement has been successful in software development, a solution to this question was presented in the form of a process improvement framework based on a personal software process improvement model. While not providing a prescriptive solution to the research question, the Personal Web Process (PWP) framework lays the groundwork for web developers to improve their own development process.

The personal web process consists of three process levels. Level 0 describes the baseline personal process. Developers using this process perform their normal development activities combined with defined data collection activities. Level 1 defines the personal project management process. This process level defines the skills, techniques, and methods required to effectively manage personal web page development projects. Level 2 outlines a personal quality management process.

Similar to PWP process level 1, PWP process level 2 defines the skills, techniques, and methods required to effectively manage and improve the quality of web pages.

5.2 Applying the Model

Analogous to the strategy of the Personal Software Process (PSP), the process improvement model for personal web page development presented within this thesis could actually be applied to web page development by using a training course. The PSP training course is usually run using a series of exercises that gradually introduce the participant to the concepts and processes of the PSP. Starting by defining a baseline process, the scope of the exercises and the exposure to PSP process elements is increased, allowing the participant to improve their own personal process.

Therefore, practical use of the PWP could be initiated in a similar manner to this. By developing a series of web page development exercises, participants in the course could be guided through the process levels, gaining project and quality management skills, and improving their own personal process. While such research is beyond the scope of the research presented in this thesis, it is a prime candidate for future work in this area.

5.3 Limitations

Although the research task presented in this thesis is valid and justifiable in its construction, the solution to the research problem does have some limitations. These limitations include a reliance on evidence from the literature, lack of a feasibility study to test and refine the model, and the narrow scope of the model.

As an engineering research approach was adopted, the first stage in developing a solution was to observe existing solutions through a review of the appropriate literature. While software process assessment and improvement models are well documented and accepted within the literature, the applicability of similar methods to web page development is unproven. However, this reliance on the literature and the similarities between software development and web page development is justified by the limited scope and prototypal nature of the model. One way to empirically assess the feasibility of the model and hence provide areas for improvement would be to conduct a study into use of the model in practice. However, due to the limited timeframe available for this research, such a study was not undertaken.

The limited scope of this research also limits its wider applicability. As the model constructed only pertains to the individual web page developer, it cannot easily be generalised to a larger scope, such as web site development and web development teams.

5.4 Future Research

Due to the limited scope of the research conducted, several issues present themselves as areas for future research. As the timeframe of the research only allowed construction of the model and not validation, one area needing further research is in actually implementing the model. As identified in Section 5.2, the PWP framework could be implemented using a training course, analogous to that which introduces the PSP. This work could concentrate on developing the course

by identifying suitable exercises, revising process scripts and forms, describing the course structure, and the identifying the resources required.

Related more to the limited amount of knowledge on the web development process, identifying appropriate web page development stages is another area requiring further research. Although the stages identified within the PWP framework are appropriate for its objectives, they may not provide an accurate view of the actual process stages. Therefore, if the model developed is to gain wider acceptance, adequate and accurate descriptions of the development process must be defined. Closely related to this issue, is identifying further quality process elements. Identifying other elements for inclusion in the PWP framework would strengthen the quality management focus of the model, and if appropriate may even allow the definition of more process levels.

On a larger scale, developing an ISO/IEC 15504 based assessment and improvement model for web development organisations would be an interesting follow on research project. Along the same lines, the PWP framework could be extended to include team development processes, and web site development. Such extensions would require an enlargement of the scope of the improvement model to incorporate the entire development lifecycle.

Other issues related to the general lack of information on the web development process include definitions of a set of requirements, an adequate design, and an appropriate measure of size. Although the Visual Page Component (VPC) outlined in Section 4.2.1.2 is an appropriate metric for size measurement within

the PWP framework, it is only a theoretical construct with no validating empirical evidence. While the argument presented for the use of a VPC is valid, it may not be the only solution to the problem of measuring web page size. Another slant on the issue is that visual and source code measurements may ultimately relate to different development phases. The process of designing a web page could involve the positioning, resizing, and construction of screen objects. If the size and position of screen objects can all be specified in design, the actual implementation of the web page only involves inserting the appropriate elements into the source code. Therefore, counting visual web page objects may give a measure of the effort required in design, while counting elements may give a measure of the effort required in implementation. Apart from possibly providing a more accurate measure, being able to measure size as well as time in separate phases (rather than over the development as a whole) may also allow improved planning and estimation.

This problem of measuring the size of a web page (in terms of the effort required to create it), has not been addressed by any research, and is therefore a prime candidate for future focus. Within the issue of size counting (in the context of the model presented in this thesis) there are also several other issues that need further exploration, including identifying the contribution of source code element development to the size measure. However, the problem of measuring the size of a web page in terms of the effort required to create it is another problem that must be solved, and the VPC metric defined within this thesis is just one of many possible solutions.

5.5 Conclusions

The similarities between software development and web page development suggest that assessment and improvement models similar to those developed for improving the software process may be suitable for improving the web page development process. Considering this issue, the research question formulated in this thesis was: *How can the web page development process be improved?*

There are several possible solutions to this question. These may include adopting new development methodologies, increasing spending on development activities, hiring more development staff, and adopting new technologies. Considering the issues explored in Chapter 2, an engineering approach was taken to this research. By using a proven process improvement strategy as a basis, the improvement model presented attempts to solve the problem of improving the web development process.

The solution to the research problem that was identified by this thesis is a *prescriptive* method for improving the personal web development process. In this context, prescriptive can be used to describe the defined method for going about improving the process, implying that this is the only way to improve. However, the PWP is more than a one-stop solution to the problem of improving the web development process. It is also a framework for implementing other models. The generic structure of the model implies that many different processes can be used in place of the process defined in this thesis. In essence, it is not precisely following the process that matters, rather that a process exists at all (Schneider, 1998). Analogous to the PSP (see Section 3.2.3.1), the PWP is a meta-process for

considering, analysing, and improving the personal web page development process.

As identified in Section 5.3, the lack of a feasibility study into the effectiveness of the model developed, makes it impossible to empirically determine whether the solution developed solves the research problem. However, as experience with the PSP validates it as a solution for improving the personal software development process, there is a high chance that the PWP model will indeed solve the research problem.

Overall, the objectives of the research have been achieved. Firstly, based on the lack of defined processes for web page development, and the applicability of software assessment and improvement mechanisms, the arguments for improving the web page development process are conclusive. This thesis is testament to the achieving the second research objective. It presents the process improvement model for personal web page development, and outlines issues in its implementation. Finally, the implementation issues and future consequences of the improvement model are considered, and discussed.

In summary, the lessons learnt from this research include:

- A defined process would benefit web page development;
- Adapting methods from other fields can be challenging;
- Methods from other fields can be adapted for use in web page development;
- More research needs to be conducted into the web development process;
- A feasibility study must be conducted to evaluate the framework.
- Creative activities can be defined and disciplined;

Summarising Lowe and Webby (1998: 2), the potential benefits from understanding and improving the web page development process include improved application quality and reliability, productivity increases, improved development visibility, improved job satisfaction, increased user confidence and satisfaction, reduced development and maintenance costs, improved management (and developer) control of the process, and more comprehensible and easier to maintain applications. If as expected, the model described within this thesis proves to be effective in improving the personal web page development process, then these benefits are likely to transpire. While the material presented within this thesis does not convey the complete picture, it makes a start in applying defined and disciplined methods to a new, innovative, and highly creative field.

REFERENCES

Andrews, K., 1996, 'Applying Hypermedia Research to the World Wide Web,' *Proceedings of* ACM *Hypermedia 1996*, URL: http://hyperg.iicm.tu-graz.ac.at/apphrweb, Accessed: 17/08/1998.

Basili, V. R., G. Caldiera, and H. D. Rombach, 1994, 'The Goal Question Metric Approach,' in J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering*, Vol. 1, pp. 528-532. Also available URL: ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.ps, Accessed: 08/05/1998.

Bichler, M., and S. Nusser, 1996a, 'Developing Structured WWW-Sites with W3DT,' *Proceedings of WebNet'96*, URL: http://aace.virginia.edu/aace/conf/webnet/html/223.htm, Accessed: 06/10/1998.

Bichler, M., and S. Nusser, 1996b, 'Modular Design of Complex Web-Applications with SHDT,' *Proceedings of the Fifth Workshop on Enabling Technologies*, URL: http://wwwi/wu-wien.ac.at/w3dt/wetice.html, Accessed: 09/11/1998.

Blustein, J., 1998, 'alt.hypertext Frequently Asked Questions,' URL: http://www.csd.uwo.ca/~jamie/hypertext-faq.html, Accessed: 10/11/1998.

Boehm, B., 1981, *Software Engineering Economics*, Englewood Cliffs, NJ: Prentice-Hall.

Botafogo, R. A., E. Rivlin, and B. Shneiderman, 1992. 'Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics,' *ACM Transactions on Information Systems*, Vol. 10, No. 2, pp. 142-180.

Bray, T., 1996, 'Measuring the Web,' *Proceedings of the Fifth International World Wide Web Conference*, URL: http://www5conf.inria.fr/fich_html/papers/P9/Overview.html, Accessed: 20/08/1998.

Copi, I. M., 1972, *Introduction to Logic*, New York: The MacMillan Company.

Creech, M. L., 1996, 'Author-Oriented Link Management,' *Proceedings of the Fifth International World Wide Web Conference*, URL: http://www5conf.inria.fr/fich_html/papers/P11/Overview.html, Accessed: 20/08/1998.

Curtis, B., H. Krasner, and N. Iscoe, 1988, 'A Field Study of the Software Design Process for Large Systems,' *Communications of the ACM*, Vol. 31, No. 11, pp. 1268-1287.

Daskalantonakis, M. K., 1994, 'Achieving Higher SEI Levels,' *IEEE Software*, Vol. 11, No. 4, pp. 17-24.

Deming, W. E., 1982, *Out of the Crisis*, Cambridge, MA: MIT Centre for Advanced Engineering Study.

De Troyer, O. M. F., and C. J. Leune, 1998, 'WSDM: A User Centred Design Method for Web Sites,' *Proceedings of the Seventh International World Wide Web Conference*, URL: http://www.elsevier.nl:80/cas/tree/store/comnet/free/www7/1853/com1853.htm, Accessed: 17/08/1998.

El Emam, K., B. Shostak, and N. H. Madhavji, 1996, 'Implementing Concepts from the Personal Software Process in an Industrial Setting,' *Proceedings of the Fourth International Conference on the Software Process*, Los Alamitos, California: IEEE Computer Society Press.

Ferguson, P., 1998, 'Using the Personal Software Process (PSP^{SM}) in Capability Maturity Model (CMM^{SM}) Based Improvement,' *Proceedings of the 1998 Software Technology Conference*.

Fowler, K., 1997, 'SEI CMM Level 5: A Practitioners Perspective,' *CrossTalk*, September 1997.

Fowler, P., and S. Rifkin, 1990, 'Software Engineering Process Group Guide,' *Technical Report CMU/SEI-90-TR-24* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Galliers, R. D., 1992, 'Choosing Information Systems Research Approaches,' in R. D. Galliers (Ed.), *Information Systems Research: Issues, Methods and Practical Guidelines*, Alfred Waller Ltd. Publishers.

Garzotto, F., L. Mainetti, and P. Paolini, 1995, 'Hypermedia Design, Analysis, and Evaluation Issues,' *Communications of the ACM*, Vol. 38, No. 8, pp. 74-86.

Garzotto, F., P. Paolini, and D. Schwabe, 1993, 'HDM – A Model-Based Approach to Hypertext Application Design,' *ACM Transactions on Information Systems*, Vol. 11, No. 1, pp. 1-26.

Gasston, J. L., 1996, 'Process Improvement; An Alternative to BPR for Software Development Organisations,' *Software Quality Journal*, 5, pp. 171-183.

Glass, R. L., 1994, 'The Software Research Crisis,' *IEEE Software*, November 1994, pp. 42-47.

Goldenson, D. R., and J. D. Herbsleb, 1995, 'After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success,' *Technical Report CMU/SEI-95-TR-009* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Gosling, J., B. Joy, and G. Steele, 1996, *The Java Language Specification*, URL: http://java.sun.com/docs/books/jls/html/index.html, Accessed: 30/10/1998.

Gruhn, V., and S. Wolf, 1995, 'Software Process Improvement by Business Process Orientation,' *Software Process – Improvement and Practice*, Pilot Issue, pp. 49-56.

Haley, T. J., 1996, 'Software Process Improvement at Raytheon,' *IEEE Software*, Vol. 13, No. 6, pp. 33-36.

Hardman, L., and B. Sharrat, 1990, 'User-Centred Hypertext Design: The Application of HCI Design Principles and Guidelines,' in R. McAlesse and C. Green (Eds.), *Hypertext: State of the Art*, pp. 252-259.

Hayes, W., and J. W. Over, 1997, 'The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers,' *Technical Report CMU/SEI-97-TR-001* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Herbsleb, J. D., A. Carleton, J. Rozum, J. Siegel, and D. Zubrow, 1994, 'Benefits of CMM-based Software Process Improvement: Initial Results,' *Technical Report CMU/SEI-94-TR-013* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Humphrey, W. S., 1990, 'People Considerations in Process Models,' *Proceedings of the Sixth International Software Process Workshop*, Los Alamitos, California: IEEE Computer Society Press.

Humphrey, W. S., 1992, 'Introduction to Software Process Improvement,' *Technical Report CMU/SEI-92-TR-7* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Humphrey, W. S., 1995, *A Discipline for Software Engineering*, Reading, MA: Addison-Wesley.

Humphrey, W. S., T. R. Snyder, and R. R. Willis, 1991, 'Software Process Improvement at Hughes Aircraft,' *IEEE Software*, Vol. 8, No. 4, pp. 11-23.

Hutchings, T., M. G. Hyde, D. Marca, and L. Cohen, 1993, 'Process Improvement That Lasts: An Integrated Training and Consulting Method, *Communications of the ACM*, Vol. 36, No. 10, pp. 105-113.

Ibrahim, R. L., and I. Hirmanpour, 1995, 'The Subject Matter of Process Improvement: A Topic and Reference Source for Software Engineering Educators and Trainers,' *Technical Report CMU/SEI-95-TR-003* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Isakowitz, T., E. A. Stohr, and P. Balasubramanian, 1995, 'RMM: A Methodology for Structured Hypermedia Design,' *Communications of the ACM*, Vol. 38, No. 8, pp. 34-44.

ISO/IEC JTC1/SC7, 1996, *ISO/IEC 15504 Software Process Assessment – Part 1: Concepts and Introductory Guide*, Working Draft V1.00.

Khajenoori, S., and I. Hirmanpour, 1995, 'Personal Software Process: An Experiential Report,' *Proceedings of the Eighth SEI CSEE Conference*, USA: Springer.

Lowe, D., and R. Webby, 1998, 'Web Development Process Modeling and Project Scoping: Work in Progress,' *Proceedings of the Eighth Australian Web Engineering Conference*, URL: http://btwebsh.macarthur.uws.edu.au/san/webe98/Proceedings/Lowe/WebE98-LoweWebby.htm, Accessed: 16/08/1998.

Netscape Communication Corporation, 1998, *JavaScript Guide*, URL: http://developer.netscape.com/docs/manuals/communicator/jsguide4/index.htm, Accessed: 30/10/1998.

Network Wizards, 1998, 'Internet Domain Survey, July 1998,' URL: http://www.nw.com/zone/WWW/report.html, Accessed: 25/10/1998.

Newmarch, J., 1998, 'Metadata Now,' URL: http://pandonia.canberra.edu.au/web/meta/index.html, Accessed: 08/11/1998.

Nielsen, J., 1992, 'The Usability Engineering Life Cycle,' *IEEE Computer*, Vol. 25, No. 3, pp. 12-22.

Nielsen, J., and D. Sano, 1994, 'SunWeb: User Interface Design for Sun Microsystem's Internal Web,' *Proceedings of the Second International World Wide Web Conference*, URL: http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/HCI/nielsen/sunweb.html, Accessed: 20/08/1998.

Paulk, M. C., C. V. Weber, B. Curtis, and M. Chrissis, 1995, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Reading, MA: Addison-Wesley.

Pressman, R. S., 1996, 'Software Process Impediment,' *IEEE Software*, Vol. 13, No. 5, pp. 16-17.

Rout, T. P. (Ed.), 1998, *Software Process Assessment and Improvement*, Southampton: Computational Mechanics Publications.

Schneider, R., 1998, 'PSP Evolution,' E-mail to PSP Users List: psp-users@tqc.com.au, Accessed: 05/11/1998.

Schwabe, D., and G. Rossi, 1995, 'The Object-Oriented Hypermedia Design Model,' *Communications of the ACM*, Vol. 38, No. 8, pp. 46-46.

Schwabe, D., G. Rossi, and S. D. J. Barbosa, 1996, 'Systematic Hypermedia Design with OOHDM,' *Proceedings of the Seventh ACM Conference on Hypertext*, URL: http://www.cs.unc.edu/~barman/HT96/P52/section1.html, Accessed: 06/10/1998.

Sherwood, C., and T. Rout, 1998, 'A Structured Methodology for Multimedia Product and Systems Development,' *Sixth ACM International Multimedia*

Conference, URL: http://www.cit.gu.edu.au/~cathie/ACM/ACM%20_Paper.html, Accessed: 22/09/1998.

Simpson, J., and E. Weiner (Eds.), 1993, *Oxford English Dictionary Additions Series (Volume 2)*, London: Clarendon Press.

Thomson, H. E., 1998, 'SwIM – A Practical Approach for Software Process Improvement,' in T. P. Rout (Ed.), *Software Process Assessment and Improvement*, Southampton: Computational Mechanics Publications.

World Wide Web Consortium (W3C), 1998a, *HTML 4.0 Specification*, URL: http://www.w3.org/TR/1998/REC-html40-19980424, Accessed: 25/10/1998.

World Wide Web Consortium (W3C), 1998b, *Extensible Markup Language (XML) 1.0*, URL: http://www.w3.org/TR/1998/REC-xml-19980210, Accessed: 30/10/1998.

Wigle, G. B., and G. Yamamura, 1997, 'Practices of an SEI CMM Level 5 SEPG,' *CrossTalk*, November 1997.

Williams, L. A., 1997, 'Adjusting the Instruction of the Personal Software Process to Improve Student Participation,' *Proceedings of the 1997 ASEE/IEEE Frontiers in Education Conference*, URL: http://fairway.ecn.purdue.edu/~fie/fie97/papers/1305.pdf, Accessed: 06/09/1998.

Younessi, H., 1998, 'Software Process Improvement: A Multi-Dimensional Perspective,' in T. P. Rout (Ed.), *Software Process Assessment and Improvement*, Southampton: Computational Mechanics Publications.

APPENDIX A – PROCESS LEVEL 0 FORMS AND SCRIPTS

APPENDIX B – PROCESS LEVEL 0.1 FORMS AND SCRIPTS

APPENDIX C – PROCESS LEVEL 1 FORMS AND SCRIPTS

APPENDIX D – PROCESS LEVEL 2 FORMS AND SCRIPTS