

Agile Web Engineering (AWE) Process

Andrew McDonald and Ray Welland

Department of Computing Science,
University of Glasgow,
Glasgow, Scotland.
G12 8QQ

02 December 2001

Abstract

This document describes the Agile Web Engineering (AWE) Process for the construction of Web-based applications. AWE is a lightweight process that has been developed to tackle the problems associated with the development of Web-based applications: short development life-cycle times; small multidisciplinary development teams; delivery of bespoke solutions integrating software and data. In addition AWE encourages: more focus on requirements analysis, including a clear analysis of business needs; better testing and evaluation of deliverables; and clear consideration of the issues associated with the evolution of Web-based applications. By identifying and managing the interaction between business, domain, software and creative design strands within Web Engineering projects, AWE provides a roadmap that allows Web-based endeavours to deliver solutions that satisfy End-Users, who are ultimately the litmus test for success.

Contents

Figures	3
1. Introduction	4
2. The Agile Web Engineering (AWE) Process Life-Cycle	7
2.1 Why do we need a Web Engineering Process?	7
2.2 Why an Agile Process?	10
2.3 The AWE Process Life-Cycle	13
2.3.1 Business Analysis	15
2.3.2 Requirements	17
2.3.3 Design	18
2.3.5 Testing	19
2.3.4 Implementation	19
2.3.6 Evaluation	20
2.3.7 Iterative and Incremental Development	22
2.3.8 Beginning the AWE Process	25
3. The Stakeholders	27
3.1 End-Users	27
3.2 The Client	28
3.3 Domain Expert	28
3.4 Business Expert	28
3.5 Software Engineer	29
3.6 Creative Designer	29
3.7 Team Leader	30
4. Team Organisation, Collaboration and Communication	31
4.1 Individual Team Demographics, Organisation, Communication and Collaboration. ...	31
4.2 Applying AWE to Large Web Application Development	34
5. Techniques and the AWE Process	36
5.1 Formal vs. Informal Communication	36
5.2 Multi-disciplinary and Uni-disciplinary Communication	37
5.3 Adopting, Creating, Tailoring and Evaluating Techniques.	39
6. Architecture	41
6.1 Usability Issues	41
6.2 Infrastructure Support	43
7. Conclusions	45
7.1 Further Work	46
8. Acknowledgements	47
9. Glossary	48
10. References	51

Figures

Figure 1	8
Figure 2	9
Figure 3	14
Figure 3.1	20
Figure 3.2	21
Figure 3.3	23
Figure 3.4	24
Figure 3.5	25
Figure 4	32
Figure 5	35
Figure 6	37
Figure 7	38
Figure 8	39
Figure 9	44

1. Introduction

The dramatic growth of the Web has made an impact on many areas of every day life. Web-based applications have forced many radical paradigm shifts in the Entertainment, Government, Commerce and Education sectors of modern society and beyond. The sprawling impact of the Web and the Internet, combined with the rapid ad-hoc approach with which most Web engineering projects are realised, have lead to concerns about the stability and success of Web-based application development [8, 30, 37, 38 & 40].

Software engineering as a discipline has been an active area in both research and industry for more than thirty years. Software engineering endeavours to provide guidance to those involved in software development. The objectives of software engineering are to produce systems that: are reliable and robust; address the problems they were developed to tackle and are delivered on time and within budget. These criteria have proved elusive over the years especially with the development of new technologies and different uses for software systems.

It could be argued that the Web is no different to other new technologies software engineering has had to deal with in the past. However, there are a number of identified criteria that differentiate Web-based application development from traditional software development. These include [25 & 26]:

1. Short-development life-cycle times, typically 3 months or less;
2. Delivery of bespoke systems integrating software and data;
3. Multidisciplinary development teams;

In addition, if the success of Web-based application development is to increase then Web-based projects need to focus more on the following [25 & 26]:

4. More rigorous requirements analysis, including a clear analysis of business needs;
5. Better testing and evaluation of Web-based deliverables;
6. More focus on the issues associated with the evolution of Web-based systems.

Many software processes and methodologies have been presented over the past thirty years. Software processes and methodologies try to tackle and provide guidance on dealing with the problems associated with software engineering projects. Essentially a roadmap guiding the development of software projects, identifying who is doing what, where, why, how and when. Recently many have begun to classify software processes as either monumental (also referred to as heavyweight) or agile (also referred to as lightweight) [17 & 21].

Monumental processes include variants on the Waterfall Approach [41] or Stageswise Model [4] such as Structured Systems Analysis and Design Method [13] (SSADM). More recent monumental process developments such as the Rational Unified Process [22] (RUP) have been influenced heavily by the risk driven developments [5] derived in the 1980's and developments in the object-oriented design community [23] in the early 1990's. Monumental processes try to cover a wide range of software development activities, in the case of RUP relying primarily on the organisations and developers involved to create a subset of the process and its supporting techniques and tools to suit their particular organisational and project needs. Most monumental processes are process oriented and predictive in nature, determining very early in the development life-cycle the problem and proposed solution.

At the opposite end of the process spectrum lie the recent developments classified as agile processes, these include Extreme Programming [3 & 24] and Dynamic Systems Development Method [14] (DSDM). Agile processes tend to focus on specific kinds of software development activities and have common characteristics such as small development teams and shorter development life-cycle times, in addition to focusing on software deliverables as opposed to documented deliverables. Agile processes encourage small teams of highly skilled developers to start with the initial agile or lightweight process and expand it to suit their organisation and projects. Agile processes are people oriented and encourage and embrace change, allowing nearly the full project development time to define the problem and proposed solutions in their entirety.

It can be argued that monumental processes allow organisations and projects to use a smaller skilled set of key software professionals who make more important higher-level project decisions. These critical high-level decisions are then used to guide less knowledgeable developers who work at a lower-level where decisions are more rigid. Based within the framework of the higher-level decisions, lower-level decisions are seen to be less critical to project success. Many in the agile community [7 & 17] would argue that this approach is flawed, and that lower-level decisions are just as critical to project success and avoiding project failure.

The objective of this research is not to get into a debate regarding monumental vs. agile processes [17]. Indeed, we believe that both approaches have their relative merits for different types of projects. There is no reason to suppose that the same methodology would be appropriate for developing a commercial billing system and a process control system. However given the identified criteria associated with Web-based development, points 1-6, we believe that the agile route lends itself more to tackling these identified issues. This is primarily due to the focus agile processes place on: embracing change; people; small development teams; and short development life-cycles.

The following document outlines the Agile Web Engineering (AWE) Process that amongst other things tries to help organisations and development teams address the issues raised in points 1-6. Throughout the rest of this document we will refer to two examples of Web application development. Both these examples are hypothetical, although much of the material associated with these examples are based upon our own experience. The examples serve to help illustrate features of the AWE process by placing them in the context of Web-based development scenarios. The first example, to be referred to as the Web-based Teller System, describes the replacement of a legacy system to support branch teller activities in a banking organisation. Throughout the rest of this document we will refer to the development of both the legacy teller system and the proposed Web-based teller system. The Web-based Teller System can be classified as an Intranet development project that is amongst many proposed to operate in branches of the banking organisation. The second example, referred to as the E-commerce Museum and Art Gallery Online-Shop, describes the addition of an e-commerce shopping facility to a museum and art gallery Web site. The Museum and Art Gallery Web site, was developed for a primarily audience of school children, the secondary audience comprising academic peers and other Web browsers. The existing Web site represents a substantial investment in resources. It is essential that the addition of the e-commerce facility enhances the existing presence and does not interfere with the primary objective of the Web site, which remains as an educational resource for children unable to attend the museum or art gallery in person. The E-commerce Museum and Art Gallery Online-Shop can be classified as an Internet development project.

The next section, Section 2, outlines the AWE Process life-cycle, detailing each stage and how the stages should be addressed using an iterative and incremental approach. Section 2 also discusses the business, domain, creative design and software models and how they should operate in relation to the AWE Process life-cycle. The following section, Section 3, identifies the various stakeholders or roles that should be reflected within the AWE Process. Section 4 discusses communication and collaboration issues that need to be addressed both between and within the software, creative design, business and domain models. In addition, Section 4 discusses an approach for assisting with orthogonal communication between developers of similar disciplines (uni-discipline developers) on different projects within the same organisation or on separate sections of the same Web presence. Orthogonal communication between developers of similar disciplines on different projects is crucial if the AWE Process is to scale to large Web engineering endeavours. Section 5 discusses the issues associated with developing, tailoring, integrating and evaluating techniques to support the AWE Process. Issues relating to the architecture of Web-based applications are addressed in Section 6. The last four sections contain our conclusions, acknowledgements, glossary and references respectively.

2. The Agile Web Engineering (AWE) Process Life-Cycle

2.1 Why do we need a Web Engineering Process?

Traditional software engineering projects are primarily concerned with the creation of software components with supporting systems, which are often generic. These software components and supporting systems are often developed independently of the data upon which they will operate. Web engineering on the other hand results in deliverables, comprising software components and supporting systems that are developed in parallel with the creation of the data that they will operate upon or in conjunction with. In essence, each Web engineering project results in a bespoke solution comprising data and software. In addition, the importance of understanding End-User usage has never been so critical to the success of a class of software applications as it is with Web engineering projects. Consider the number of different ways information in a Web application can be displayed to End-Users, and the ease with which End-Users can find and change to alternative Web-based solutions should they lack satisfaction with their current usage experience. We believe, as do others [11 & 33], that if one is to build successful Web applications then great focus has to be placed on understanding End-User usage of the proposed system.

Consider the models involved in any traditional software engineering process. We would argue that there are only three models of any great significance reflected by traditional software engineering processes: the software model, the business model and the domain model. The software model reflects a view of the issues associated with developing a software solution that achieves the business objectives reflected by the views of the business model. The domain model reflects views of the domain to which the business objectives and the proposed software solution are to be applied. For example, consider the development of the existing legacy teller system in our Web-based Teller System example. The domain model would reflect the issues associated with branch and teller activities, detailing working procedures and environmental conditions under which these procedures are carried out. The business model would reflect the business objectives of the branch and teller activities, detailing the perceived inefficiencies of the current procedures and the proposed benefits of the new automated system. The software model would reflect the issues associated with developing the proposed system to automate the branch teller activities. The primary impact reflected by the traditional software engineering process used to develop the legacy system was in the software model, driven by views from the business and domain models. Rarely do traditional software engineering processes explicitly give feedback into the business and domain models.

Figure 1 shows the impact business, domain and software models have upon each other in such traditional software engineering process. The software model is impacted only by a partial section of business and domain models. This impact is exerted usually only once or twice during the

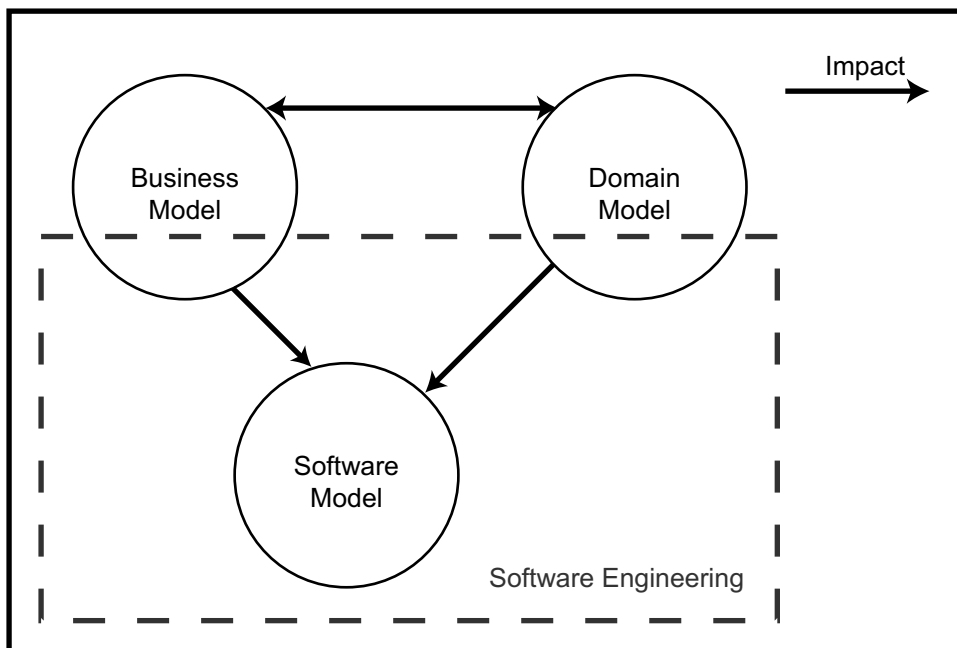


Figure 1. Shows the Impact the Software, Business and Domain Models exert upon each other in Traditional Software Engineering Processes.

development life-cycle and the information that the software model holds regarding the business and domain models is primarily seen to be static. Rarely does the software model have a radical impact on the business and domain models. As a result, many of the software solutions resulting from these traditional software engineering processes are largely implementations of existing business practice.

Web Engineering on the other hand is complicated by the addition of a creative design model, that reflects the issues associated with the aesthetic aspects of the user interface. In addition, Web engineering requires the business and domain models to design and develop data, influence Web site structure and therefore requires them to not only impact and affect change in the software and creative design models, but requires the software model to impact and affect change in the business and domain models. The business and domain models also have to address impact and affect change between them in Web engineering. See Figure 2 for the impact business, domain, creative design and software models have upon one another in Web engineering. We believe that the development of Web-based applications often requires a degree of re-engineering in the business, domain and software models. Such is the impact exerted between the four models in Web-application development, that if an organisation wishes to harness this impact to their benefit, then a re-engineering initiative is required to adapt the models in order to ensure the success of the proposed system. It is crucial that those organisations and individuals involved in Web-based endeavours understand the impact exerted amongst the models in Web engineering. Indeed many of the e-words, such as e-Revolution and e-Transformation, associated with Web-

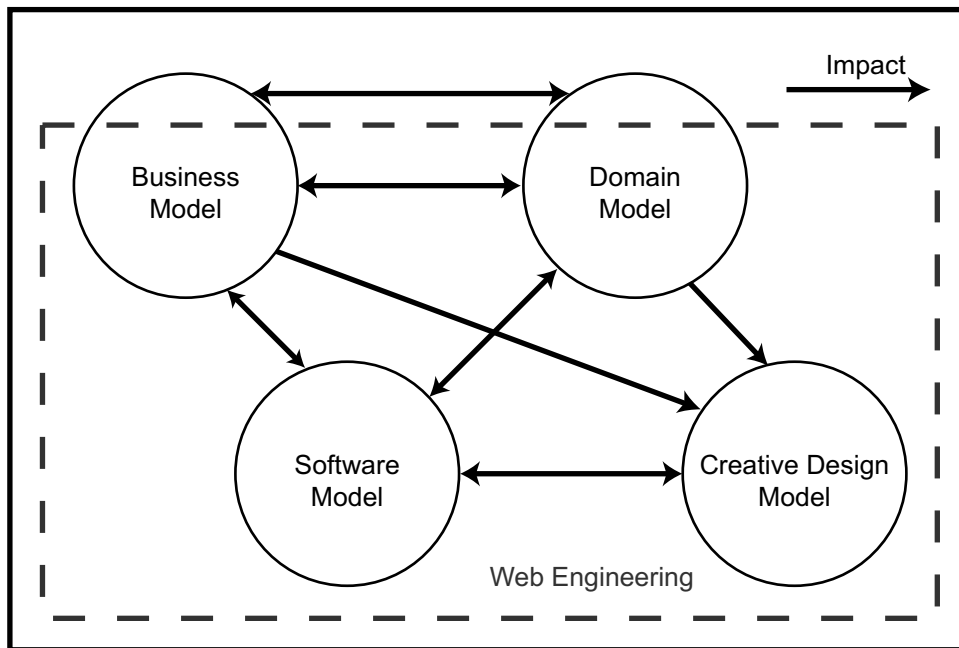


Figure 2. Shows the Impact the Software, Business, Domain and Creative Design models have on each other in Web Engineering.

based developments indicate the importance of such re-engineering activities. Ultimately, we believe that organisations and individuals that do not understand the significance of re-engineering during Web application development risk the success of their projects and long term survival of their organisation.

This is not to say that the business and domain models can develop their elements independently. Rather that the contribution business and domain models have in Web engineering is so substantial and critical to the success of the Web application, that both should have development models with different views from the software model reflected in the development team. It is not realistically possible for any one individual stakeholder to understand every view reflected from each of the models impacted in Web engineering. However, it is crucial that every different type of developer, whether business expert, domain expert, manager, creative designer or software engineer, educate and collaborate with others within the team to achieve the best solution given the project or organisational problems or goals to be tackled. See Sections 3 and 4 of this report for a detailed explanation of the stakeholder roles reflected in the AWE Process and how these different developer roles should interact during the development life-cycle. In addition, developers within the team are required to understand the impact exerted on their model and to feed this understanding back into the respective re-engineering process. Business Process Re-Engineering [19 & 20] is a topic in its own right. The AWE Process tries merely to make those involved in Web-based development aware of the importance of Business Process Re-Engineering and to provide communication mechanisms and Web application development structure that will support re-engineering activities.

Consider the development of the Web-based Teller System to replace the legacy teller system. The problem is complicated by a software model that has to have an impact on the business and domain models. For example the impact of Web-based technologies may render current operational methods redundant and may present new business opportunities. Therefore, the impact exerted by the Web-based technologies will have a direct impact on the business and domain models. The creative design model, newly introduced, is partially responsible for the issues associated with the user interface design, focusing on the aesthetic aspects of the Web application. The business and domain models will also have to impact the creative design model. For the first time it may be possible to introduce the corporate identity or branding across all teller applications, allowing for easy End-User distinction between the Web-based Teller System and other external Web applications. Conflicts will often arise between aesthetic design and usability. The relationship between the software model and the creative design model now has a two way impact.

The impact exerted between the software, creative design, business and domain models in Web engineering has to be reflected in a Web engineering process, if it is to tackle the problems associated with Web application development. It is essential that the development team is aware of the various interactions between the models in Web engineering during the development life-cycle, and can harness these not only with Web-based deliverables but also with Business Process Re-engineering activities. We believe that current software processes lend themselves poorly to reflecting the influences exerted between the models in Web engineering. The next part of this report goes on to describe Agile processes and why we believe the agile route lends itself well to addressing the problems a Web engineering process has to tackle.

2.2 Why an Agile Process?

The term agile has recently been used to categorise a number of lightweight approaches to building software. These include: Extreme Programming [3] (XP), Adaptive Software Development [21] and Dynamic Systems Development Methodology [14] (DSDM). Seventeen advocates and methodologists of the aforementioned and other agile processes convened in February 2001. The result of this meeting was the formation of the Agile Alliance [2] and the production of The Manifesto for Agile Software Development [15].

The following quote from The Manifesto for Agile Software Development¹ gives a summary of it's purpose:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

¹ *Note: We have added a numbering scheme to The Manifesto for Agile Software Development to aid the identification of specific statements.*

- (i) Individuals and interactions over processes and tools.
- (ii) Working software over comprehensive documentation.
- (iii) Customer collaboration over contract negotiation.
- (iv) Responding to change over following a plan.

That is, while we value the items on the right, we value the items on the left more."

-Kent Beck et al.

Ultimately we believe that the developers and organisations involved in Web engineering projects are the primary factor in the success or failure of Web application development. Given the diversity of disciplines required to develop Web-based applications, we are of the opinion that the AWE Process, or any other process or methodology, can only hope to have a second order effect on project success. Thus, we hold the belief that the agile route with its focus on people, point (i), lends itself strongly to Web-based development.

Our belief that people are the most important factor in project success is the fundamental reason why we have not tried to develop a monumental process to tackle the problems associated with Web application development. Many monumental processes attempt to codify good practice and experience in too much detail and for developers who do not understand the importance of what they are doing! This often results in development projects using monumental processes as cook-book recipes, where developers are lulled into a false sense of security by following the recipe in detail rather than using the ingredients selectively to help them build software deliverables that solve their problem space. Ultimately many development teams use such monumental processes rarely as their designers intended [1]. Projects using such processes often spend most of their resources producing volumes of documentation as opposed to focusing on delivering working software systems that address the problems projects are meant to tackle. The AWE Process places the onus on developers and organisations involved in Web engineering to deliver Web-based solutions that satisfy their project goals. This is the first and foremost objective of AWE. However, we do not believe that documentation or techniques are unimportant. On the contrary, we are of the opinion that documentation, techniques and tools enhance and play a critical role in the success of many Web-based projects. It is just that we believe the most important project deliverable is the Web-application itself, as stated in point (ii).

Consider the models involved in Web engineering, see Figure 2. We believe the rate of change within these models and the impact change within one model has upon other models necessitates that the AWE Process support collaboration (iii) and the ability to adapt and respond to change (iv) effectively. While every project needs contractual agreements and guidelines to help ensure project success, we believe that collaboration and the ability to adapt to the impact change exerts

between each of the models is critical to project success. Thus, we believe contractual arrangements need to be flexible enough to support collaboration and adoption of change between the models involved in Web engineering.

In addition to the previously stated purpose of the Manifesto for Agile Software Development the document also includes twelve principles¹. The following quote lists the twelve principles as stated in the Manifesto for Agile Software Development:

"We follow the following principles:

- (A) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- (B) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- (C) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- (D) Business people and developers work together daily throughout the project.
- (E) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- (F) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- (G) Working software is the primary measure of progress.
- (H) Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
- (I) Continuous attention to technical excellence and good design enhances agility.
- (J) Simplicity—the art of maximizing the amount of work not done—is essential.
- (K) The best architectures, requirements and designs emerge from self-organizing teams.
- (L) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly."

-Kent Beck et al.

AWE is an iterative and incremental process, we believe this will allow for: early and continuous delivery of valuable software (A); the ability to harness changing requirements, even late in development (B); and the delivery of working software frequently (C). The AWE Process supports multidisciplinary development treating business experts, domain experts, and creative designers as developers along side software engineers. AWE encourages developers to work in

¹ Note: We have added identifying letters to the twelve principles from *The Manifesto for Agile Software Development* allowing us to easily refer to them in this report.

close proximity to each other (D) on a daily basis, in an environment that supports productive development. We believe that close working proximity and informal (face-to-face) communication (F), primarily through the browser experience, between developers, allows for faster and more productive development. We believe it is the responsibility of each developer (E), team and organisation to continually deliver Web applications that satisfy End-Users usage and that ultimately this is the primary measure of project success (G). The AWE Process's primary aim is to deliver working Web applications that satisfy End-User usage and therefore project objectives. Delivery of working software should be sustainable through AWE at a constant rate (H). Ultimately it is the responsibility of every developer to make AWE work through integration of techniques and attention to simple (J), high quality design (I) and implementation of Web-based solutions. AWE places the responsibility on the development team to influence the architecture, requirements and design of the proposed Web application (K). After each life-cycle iteration the development team is required to reflect and evaluate the development process (L).

The remainder of this section outlines the AWE Process life-cycle and it's respective stages that need to be addressed during project development.

2.3 The AWE Process Life-Cycle

The AWE Process identifies all the major stages we feel need to be addressed during Web-application development. Anyone who has experience of software processes, particularly variants on the Waterfall Model, may initially shudder at the diagram in Figure 3. While many of the names for each stage: Business Analysis, Requirements, Design, Implementation, Testing and Evaluation, should look familiar from other processes and methodologies, the similarity should end there. The only deliverable that is required to be produced from the AWE Process is the Web application itself. That is not to say that you will not benefit from the production of intermediate documents, diagrams and other notations. It is just that we do not impose these upon you. The onus is on the organisation and the developers to find, integrate, evaluate and create techniques, if necessary, to support the activities outlined in Figure 3. See Section 5 for a detailed discussion of selecting, integrating, evaluating and developing techniques to support the AWE Process.

Recently the agile process Extreme Programming (XP) has gained a lot of attention. In XP developers are encouraged to communicate through the project source code, this is achieved through pair programming and other XP approaches. Communication through source code on a small software engineering project is viable due to the nature of the deliverables (software components), the technical background of the developers and the small size of the team. Unlike software engineering, Web engineering results in deliverables that comprise data integrated with software. Web engineering does comprise small development teams, however these teams are multidisciplinary in nature and in the vast majority of cases will not be able to communicate

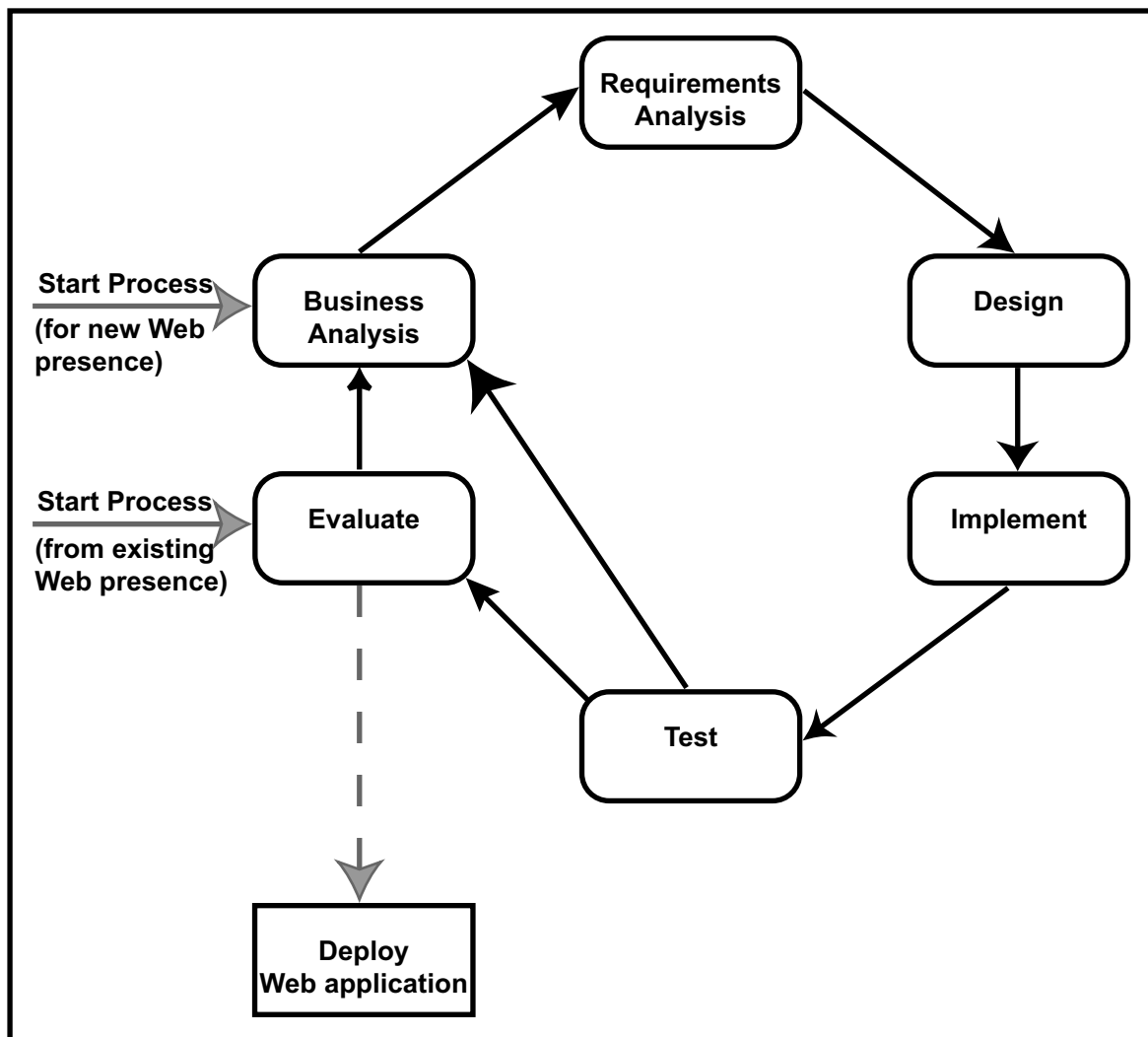


Figure 3. The Agile Web Engineering (AWE) Process Life-Cycle.

through the source code of the Web application. Given the importance of understanding End-User usage of the proposed system, we believe all team members should communicate through the Web application browser experience. Even if a developer is working on the database in an n-tier Web application there will be issues relating to the browser experience that all developers must be involved in. For example, the database design may require the Web application to collect data from the user in a particular order. Should this order reveal poor user satisfaction with the browser experience then the database design should be changed to allow the Web application to focus more on satisfying End-User usage. See Section 4 for more details on collaboration and communication within the development team.

Consider the E-commerce Museum and Art Gallery Online-Shop. The Museum and Art Gallery in question already has a large Web presence, developed over five years, with the primary focus of fulfilling an education role aimed at school children who are unable to attend the Museum and Art Gallery due to the geographical distances. The primary audience of the existing Web site is

children, the secondary audience being the general public and academic peers with interests in the museum collections. With such a diverse audience to address, the developers have put a lot of effort into ensuring the Museum and Art Gallery browser experience of its primary and secondary audiences is highly satisfactory. One of the mechanisms through which this has been achieved is development focus around the browser experience, with continuous feedback through evaluation with samples from the intended primary and secondary audiences.

In addition to communication through source code XP advocates the use of metaphors to reflect architectural issues relating to project development. We believe that this indicates a relatively static and stable architectural environment. However, Web applications need to be highly scalable and portable, in addition to allowing easy replacement and upgrading of supporting software infrastructure (e.g. database management system), if they are to ensure the longevity and growth expected by many business plans. For these reasons we felt the need for the AWE Process to give specific focus on architectural issues relating to Web application development. See Section 6 for a more detailed discussion on architectural issues in the AWE process.

Consider our first example, the Web-based Teller System. Initially the first system will be trivial as the prototype will only be piloted in three branches. However deployment of the final system will be in hundreds of branches. As a result, the architecture will have to scale in order to cope with such a dramatic increase in the number of Teller Positions. The architecture in the E-commerce Museum and Art Gallery Online-Shop will be the same for deployment as for development, although it should be noted that the development work will not be carried out on the live Web presence.

2.3.1 Business Analysis

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

Many Web sites are developed because the organisations involved are perceived to “need a Web

site”, rather than to fulfil properly thought out business objectives. Should you find yourself developing such a Web-based system, then you should ask the question. Why does the organisation need a Web site? Note, “because our competitors have one”, is not a good enough answer to continue with project development. You must address what competitive edge your competition is gaining through the Web and how you aim to re-address this by your Web-based endeavours.

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

For example, the first iteration of the AWE process on the Web-based Teller System might be to access customer bank account information. The first iteration may not allow the user to reason or perform transactions with an account, or understand how to present the information in the most meaningful way. All the system may do is to allow the user to see customer account details. The first iteration of the AWE process on the E-commerce Museum and Art Gallery Online-Shop may be to present all the information from the product database on a series of Web pages. Again, the user may not be able to reason with this information, but the project cannot be completed unless this function can be performed by the proposed system.

Most projects will begin the AWE process at the Business Analysis phase. If the team find it difficult to describe or agree the problem space then it may be prudent to backtrack to the Evaluation phase. The Evaluation phase in this instance can be used to evaluate an existing Web presence or those of competitors to enable the team to better understand the problems. In addition to using the Evaluation phase to better understand the problem, the team should also give thought to the Evaluation plan that should answer the question: have we built the right product? In other words, a solution that solves the problems identified.

Consider again the first iteration of the AWE process for our second example the E-commerce Museum and Art Gallery Online-Shop. During the Business Analysis phase the development team come to the conclusion that they want to increase the number of lines in the online shop to include everything sold by the Museum and Art Gallery. For example, slides of the Art Gallery’s picture collection, currently sold only by mail order. The team decide to evaluate other online Art Gallery Web presences to see if anyone else is selling slides online and to learn from these sites.

Our survey of Web engineering in practice [25 & 26] showed that the majority of Web development teams had problems with understanding what the requirements were and in controlling continuous changes to the requirements. If you intend to use the AWE process then all developers should leave this phase with a clear understanding of the problems to be tackled, at least for each iteration of the development cycle. Each iteration of the life-cycle should address the problems that the team perceive to present the greatest project risk. In addition, the team should also begin to address evaluation criteria to be used in the Evaluation phase to assess whether or not the problems have been solved.

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

2.3.2 Requirements

It is crucial that every member of the team agrees the problems to be solved before the requirements activities begin. The Requirements phase is about defining what the proposed solution will do (functional requirements), and what constraints are to be placed upon the proposed solution (non-functional requirements). In Web application development, there normally exist a large number of non-functional requirements that have to be properly addressed if the Web development process is to be successful. Great consideration should be given to addressing the non-functional requirements, in particular those associated with architecture, including the user-interface and usability. For a more detailed discussion of architectural issues, see section 6.

AWE does not suggest techniques to support the process. That is not to say we do not encourage the use of techniques, just that we do not impose any upon you. There exist many resources [11, 22, 23, 24, 27, 42 & 48] a team can use to investigate techniques to support the requirements phase.

In the Web-based Teller Example a large number of initial functional requirements were derived during a feasibility phase in order to gain project funding. The development team starts by addressing these requirements to see if they are still valid and go about ranking them according to their importance to project success. Those that are perceived to present the greatest risk to the project by the development team are addressed first. The E-commerce Museum and Art Gallery Online-Shop development team have decided to use story cards [3] to capture requirements.

After the requirements have been identified, the team should place focus on starting to create a test plan for use in the Test phase. The goal of this exercise is to enable the team to answer the question: have we built the product right? It is essential that entire team agree as to the tests being devised to address this question. Otherwise, the project runs the risk of falling short or over engineering the solution to agreed requirements. See section 2.3.5 for a detailed discussion on the Test phase.

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

2.3.3 Design

Design involves understanding, co-ordinating and communicating all the major issues, before implementation, of building a complex Web application. These issues should be independent of the lower-level implementation details. The AWE process when applied to large Web application development projects requires communication between similar types of developer in separate teams on the same Web presence. This communication serves to enable re-use, not only deliverables but also architectural design patterns. It is important to understand that if one requires a Web application to grow and mature quickly, and one wants to avoid serious portability and redesign problems, then great attention has to be paid to architectural concerns. See section 6 for a more detailed discussion of the architectural issues associated with Web engineering.

With respect to techniques that support the design of Web applications, there are many good examples that can be used to assist with the design of Web-based systems [9, 42 & 48]. We do not impose these or any other techniques upon you. We put the responsibility on every developer involved, to ensure that the team has an adequate grasp of the major higher-level issues associated with Web application development, in order to realise a potential solution. If you find this difficult, good, it is not easy. If you feel techniques help you address higher-level issues associated with Web application design, then use them. If you find that techniques are not helping then don't use them. However, remember that you are responsible for the design of your Web application. All the AWE process aims to achieve is to make developers aware of the major issues you need to address during Web application development.

In the Web-based Teller System the team decide to use the Web Application Extension [9] for the Unified Modelling Language (UML) [16] to assist with the design of the middleware for the Web application. In the E-commerce Museum and Art Gallery Online-Shop, the development team decides to use story boards to model the user interface and a typical user scenario when

trying to purchase an item from the shop.

2.3.4 Implementation

We consider implementation, or design at the lowest level [7], to be just as important as high-level design. While there are many differences between the Design and Implementation phases, both involve decisions that have a critical impact on project success. The agile process XP encourages developers in teams, mostly comprising software engineers, to communicate through the source code of the project deliverable. This is achieved through pair programming and extensive testing. Unfortunately, while communicating through source code may be suitable for those reflecting views from the software model and the creative design model, it is not practical for communication amongst all developers in a multidisciplinary team. Instead, we recommend communicating through the browser experience. All developers should collaborate and focus their development efforts around the Web interface, using collaborative sessions to discuss and review the browser experience. Only then can you hope to begin to address the issues crucial to End-User satisfaction with your Web application.

2.3.5 Testing

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

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

Testing requires input from the Requirements, Design and Implementation phases. The requirements phase contributes very high level tests, for example: given input X the system will respond with output Y; the system will be able to cope with 400 concurrent users at one time. The design and implementation phases contribute tests and information for tests at a much lower level, for example, if $Z > 10$ during transaction type A, then the system will queue T for R seconds. See Figure 3.1 for a description of the phases involved in the creation of the Test Plan.

Many Web developers create their tests as they design and implement the system, however this can often lull teams into a false sense of security. Developers should be aware that a good

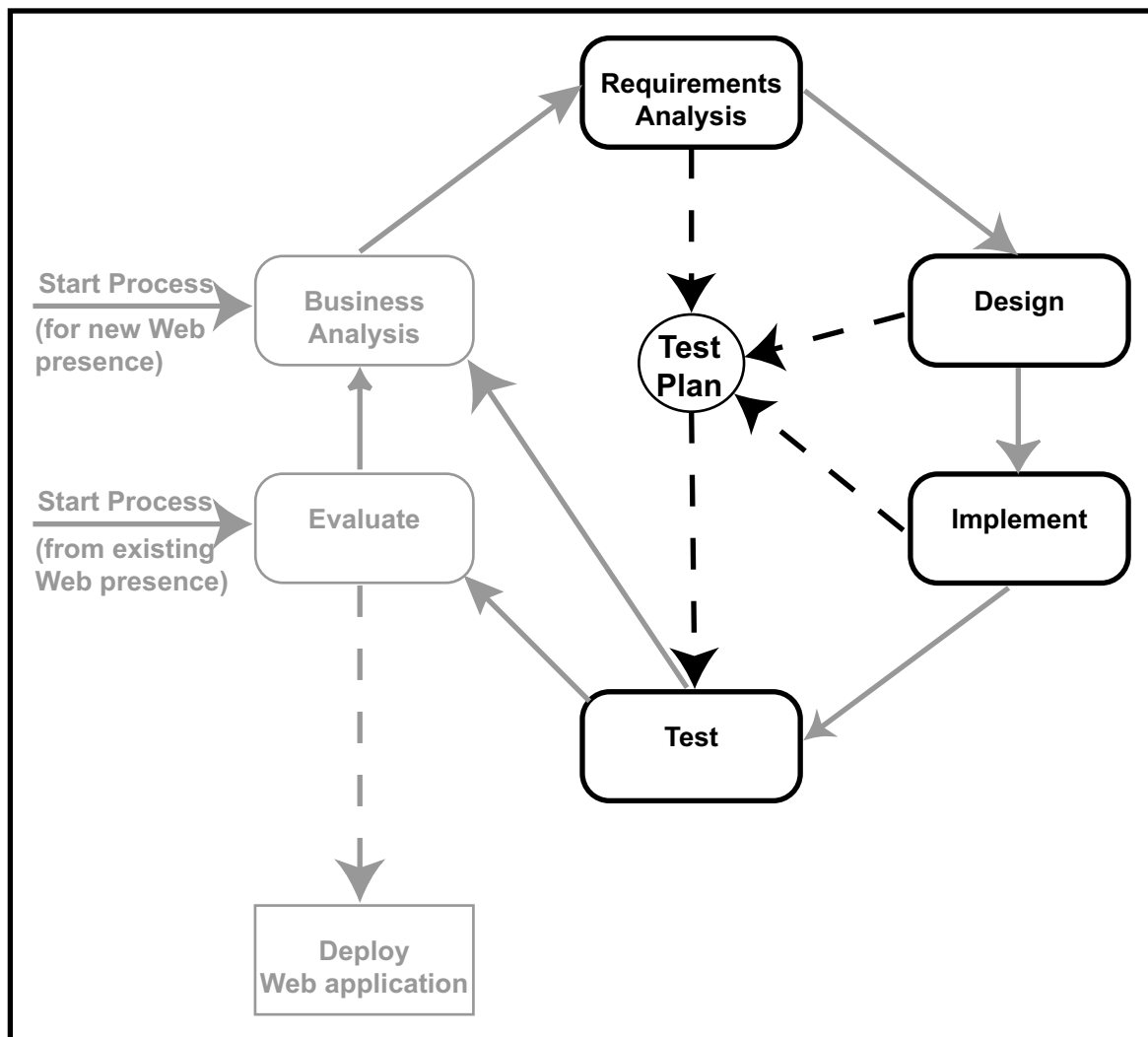


Figure 3.1. Describes the Agile Web Engineering (AWE) Process Life-Cycle, showing the Phases involved in the creation of the Test Plan.

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

2.3.6 Evaluation

The Evaluation Plan should be derived from the issues or problems identified in the Business Analysis phase. See figure 3.2 for a description of the phases involved in the creation of the Evaluation Plan. Essentially the team must address the Evaluation phase by asking themselves the following question. Have we built the right product? It is imperative that the team objectively evaluate what has been delivered independent of design and implementation issues. Only then will everyone be able to assess whether or not the project is solving the problems to be ad-

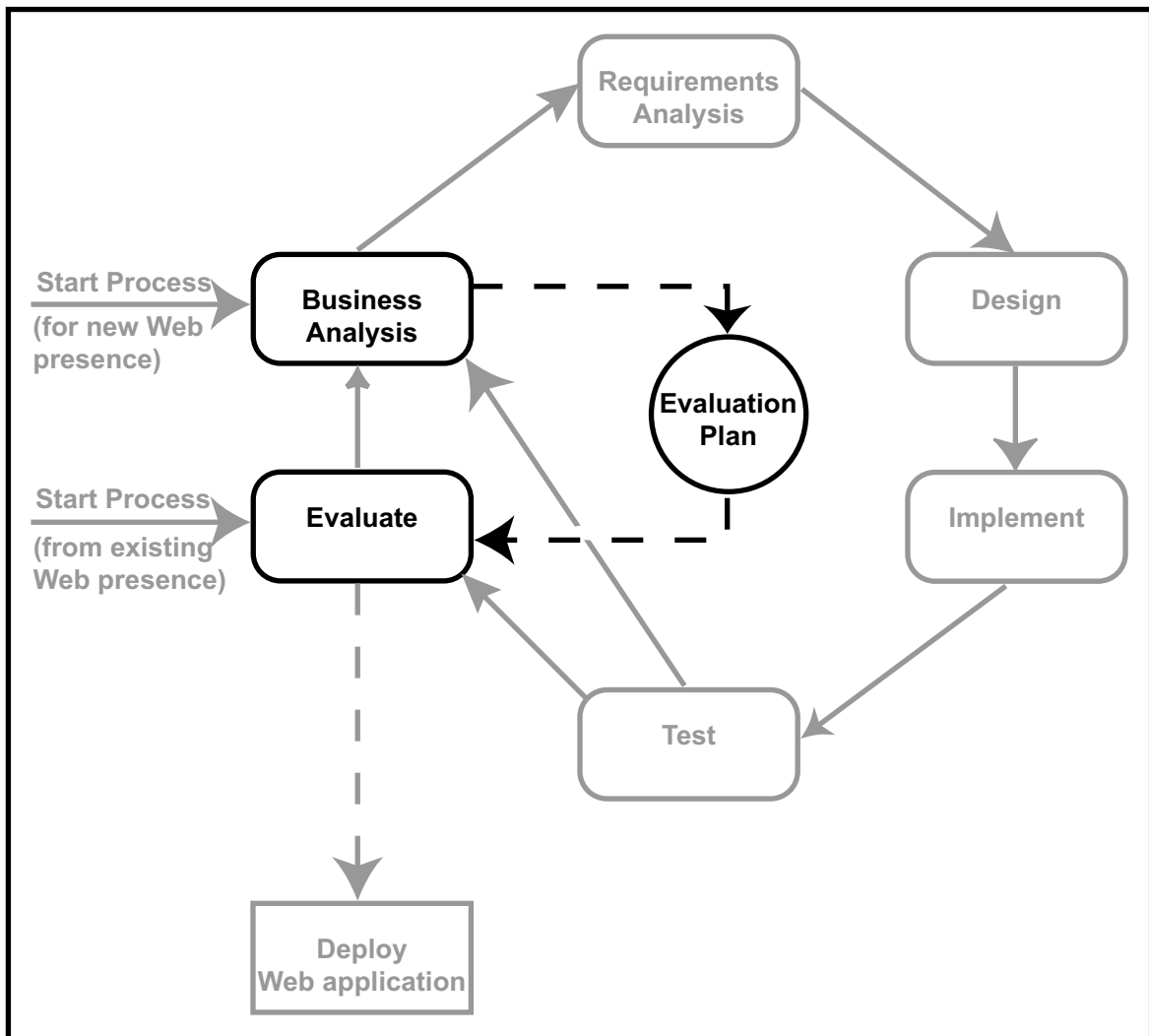


Figure 3.2. Describes the Agile Web Engineering (AWE) Process Life-Cycle, showing the Phases involved in the creation of the Evaluation Plan.

addressed. Evaluation when carried out by skilled developers measuring End-User usage often leads to a greater understanding of the problems or issues to be solved. This new understanding of the problem space needs to be feed back into the Business Analysis and Requirements phases.

Understanding what is involved in carrying out an evaluation of Web-based applications is an area in its own right. There are many good sources [10-12, 31-36, 39 & 44] of information to guide and assist a development team with this task. The intention of this sub-section is to make the team aware of some of the issues involved in evaluating a Web application.

It is imperative that a representative sample of End-Users is used to evaluate the deliverables. While End-User substitutes are a valuable resource there is no substitute for evaluation with End-Users. It is also important that the developers understand how to interact with End-Users during evaluation, just asking them what they think or want is known to be flawed [34]. It is imperative to observe usage of your deliverables by End-Users in a situation that mimics the normal usage environment.

In our experience, some developers involved in Web application development avoid Evaluation as they see it as an obstacle to delivering their project. Ultimately this comes down to whether the team wish to create Web-based deliverables or solve problems with Web applications. It is understood that Evaluation can be time consuming and expensive [12]. However, this does not necessarily equate to longer development times and greater project costs. If you want to address the issues associated with Web application development then evaluation will help you understand them sooner. The earlier the team understand the problems to be addressed the better the chances they have of solving them. It may not be practical or economically feasible to carry out an evaluation phase during every iteration of the AWE life-cycle. However, we believe that ultimately the Evaluation stage will help achieve shorter life-cycle times and lower-development costs by helping the team to understand what the needs of End-Users' are.

2.3.7 Iterative and Incremental Development

We believe that iterative and incremental development cycles are crucial to making the AWE process work. Many Web development projects start out with a feasibility phase followed by a lengthy initial requirements analysis and definition phase. The predictive approach, where very early in the development life-cycle the problem and proposed solution are determined and remain largely static, is often useful from the perspective of providing project characteristics (budget, time-scales, resource allocation) that satisfy the initial demands from senior management. However, the predictive approach often lulls organisations into the illusion that they understand everything that needs to be addressed by the proposed system. Usually on such projects there follows the design and implementation of the proposed system, with little or no attention paid to re-addressing the problems (Business Analysis) and the definition of the proposed solutions (Requirements). The failings in the proposed solution are never properly identified until after the testing phase is completed, and even then, without a proper evaluation phase, problems are not noticed until the system is live and being used by End-Users.

It is well understood by many software professionals, that the quicker a problem is addressed during the life-span of a system the more cost effective it is to address the problem. We believe that the longer the development teams continue with project development the more they understand about the problem space and whether the proposed system will effectively tackle the issues that need to be addressed in this problem space. Ultimately if the team cannot re-address the problems (Business Analysis) and the proposed solution (Requirements) then the learning curve gained during development presents no value to the organisation or project in question. This predictive approach to Web application development presents the biggest challenge to the adoption of the AWE Process. Many organisations and projects will need to challenge the inertia that surrounds Web and software development activities if they are to employ AWE on their Web engineering projects.

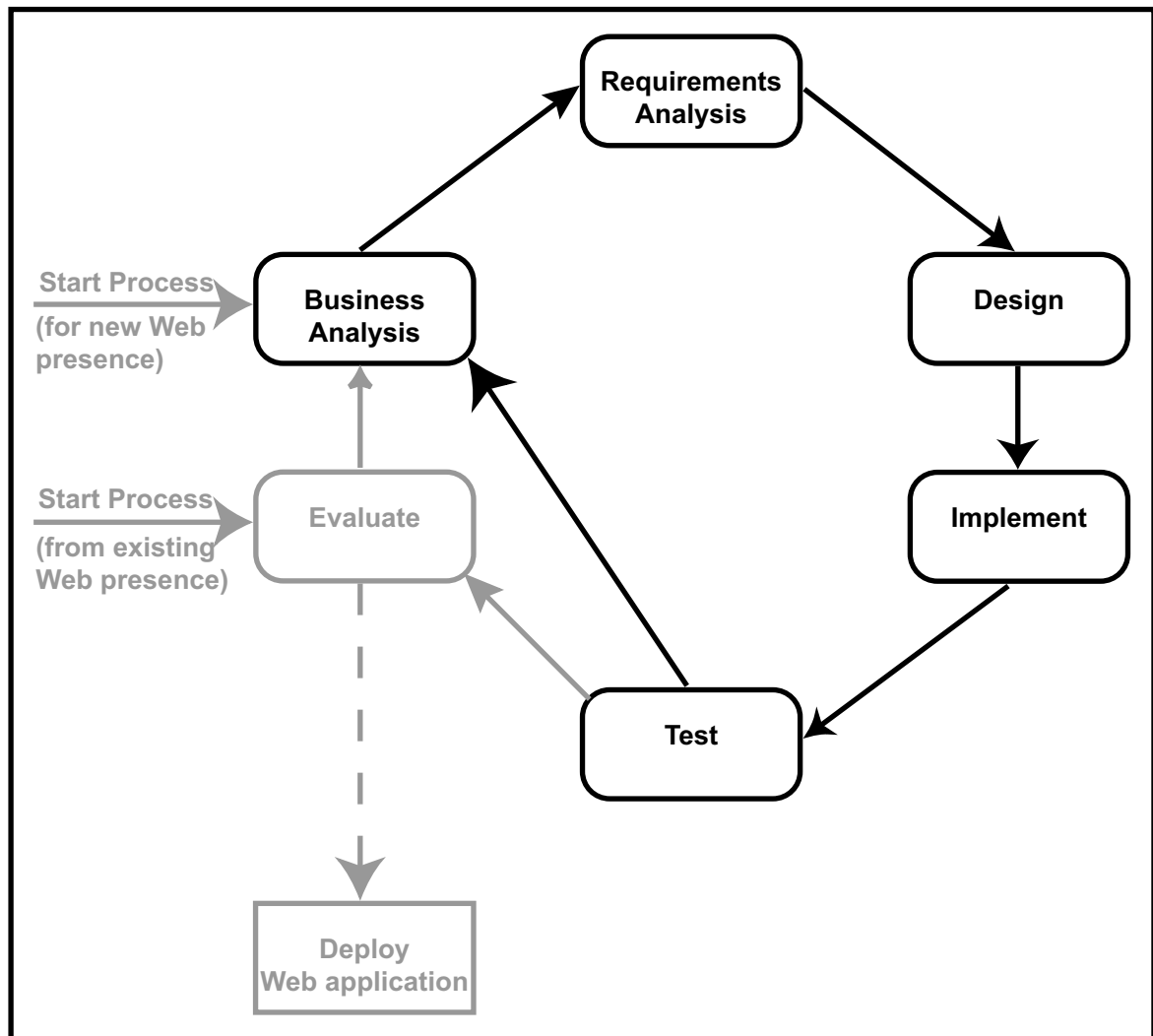


Figure 3.3. Describes a typical Iteration of the Agile Web Engineering (AWE) Process Life-Cycle.

The predictive approach to large Web application development is often compounded by the fact that those involved in feasibility, requirements analysis and definition (Business Analysis and Requirements) are different to the teams who develop the proposed solution (Design, Implementation, Testing, Evaluation and Deployment). Using different people for project definition and development presents a communication barrier to those tasked with building the proposed solution. We believe that it is crucial that all developers are involved in each phase of the AWE life-cycle. The predictive approach is particularly popular on large Web development projects, where if one truly intends to harness the impact exerted between all the models in Web engineering, then one must question the validity of the predictive approach.

When adopting an iterative and incremental approach to developing a software system it is essential to identify the criteria that will be used to select what problems are to be addressed first. Most of the selection criteria employed are centred around the identification of those problems that are perceived to present the highest risk. Indeed, the AWE process is no exception. Each

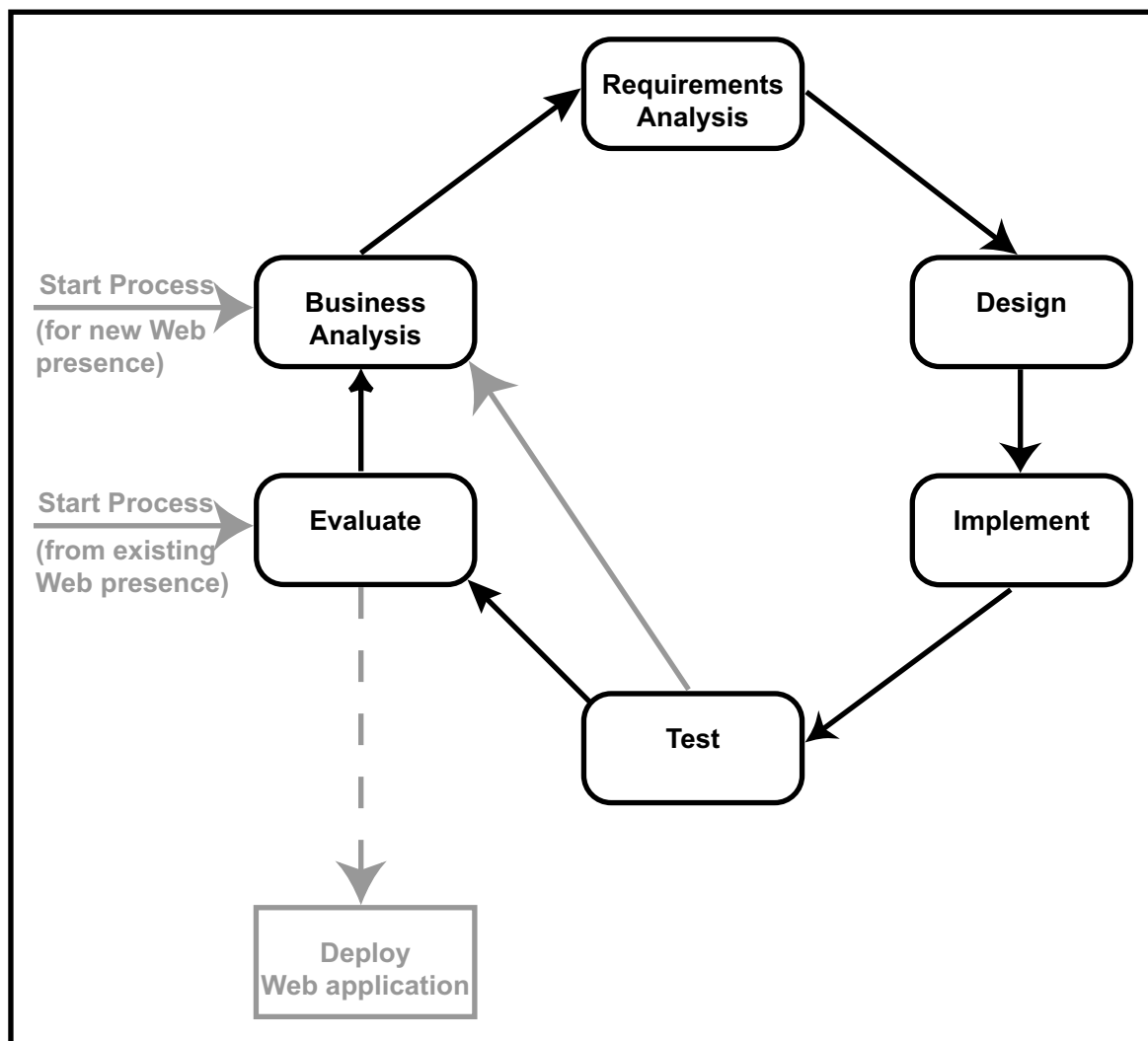


Figure 3.4. Describes an Iteration of the Agile Web Engineering (AWE) Process Life-Cycle with an Evaluation Phase.

iteration should focus on those problems that present the highest risk to project success should they not be achieved. Each iteration should focus on solving a subset of the problems that present the highest risk, ensuring that previous iterations efforts are not compromised by the incremental increase in development scope. It is also prudent to remember that the simpler the solution to a problem the easier it will be to change and enhance the overall proposed solution.

As mentioned previously it is essential to involve End-Users in an evaluation phase before one can confidently predict that the proposed solution will solve the problems identified in Business Analysis. However, evaluation with End-Users is expensive and time consuming [12]. This presents a serious conflict with the agile manifesto goal of short development life-cycle times. Ultimately we do not recommend Evaluation with End-Users during each iteration of the AWE Process. Figure 3.3 describes a typical iteration of the AWE life-cycle.

Evaluation should be used only when there is a substantial deliverable that can be measured against the identified problems and when the return on investment is seen to be worth the costs involved. We make no predictions on how often the Evaluation phase should be included, with the exception of the last iteration before deployment, where it is essential to evaluate the deliverable in order to have confidence in the deployed solution. Figure 3.4 shows an iteration of the AWE life-cycle with the Evaluation phase included. It is the responsibility of the developers and organisations involved to judge how often to include the Evaluation phase.

2.3.8 Beginning the AWE Process

There are two phases that can be used to start the AWE Process life-cycle: the Business Analysis phase and the Evaluation phase. If you are starting a new Web-based endeavour then you should enter the AWE Process life-cycle through the Business Analysis phase. However, it should be noted that greater problem understanding may be gained through Evaluation of competitors Web applications or other Web-based solutions. If you are evolving an existing Web application or presence then it is prudent to start the AWE Process life-cycle through the Evaluation stage,

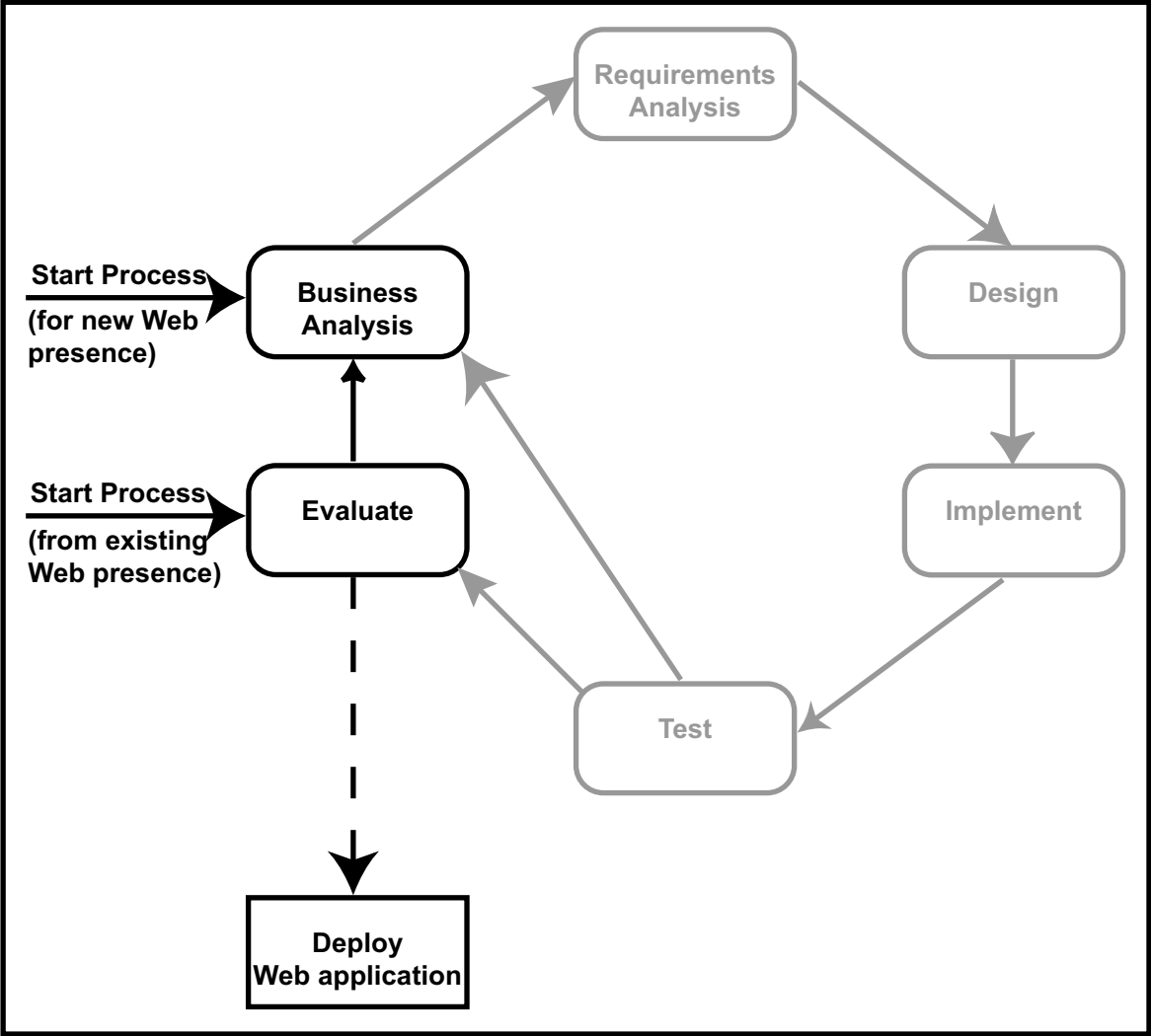


Figure 3.5. Describes the Agile Web Engineering (AWE) Process Life-Cycle, showing entry Phases to the AWE Process.

where the existing Web application or Web presence is evaluated using previous project goals. Entering the AWE Process through the Evaluation phase when evolving an existing Web application will help tremendously with the Business Analysis phase. Figure 3.5 shows the entry phases to the AWE Process life-cycle.

The next section goes on to discuss the various stakeholder roles that are represented in the AWE Process.

3. The Stakeholders

Within each software process there are a number of stakeholders or roles that reflect different viewpoints or interests within the development framework. The AWE Process is no exception. Each of the roles represented below reflects an important viewpoint onto the development process. There can be multiple stakeholders in each role. However it is important that each role be filled by someone who has the ability to competently reflect the issues associated with each viewpoint.

3.1 End-Users

End-Users are the litmus test for success. The multidisciplinary nature of Web development often leads many projects to ignore or not pay enough attention to the End-Users. While many disciplines are involved in Web engineering it should never be forgotten that the End-Users of the system ultimately decide the usefulness of the deliverables and therefore the success of the project!

The AWE Process advocates close involvement with End-Users, both in the Business Analysis stage to help developers understand the problem, and in the Evaluation stage to validate the deliverables against the problems or goals previously identified in the Business Analysis phase. Many projects in our experience use clients to validate the deliverables. Substituting Clients for End-Users will not in the majority of cases help developers understand usage of the proposed system.

Consider our example Web-based Teller System, the End-Users should be made up of a random sample of banking staff who will use the proposed system. It is crucial that the sample of End-Users is representative of all who will use the proposed system, including: bank tellers themselves, supervisors, auditors, managers and any other bank staff who will be required to use or interact with the proposed system. In our other example of the E-commerce Museum and Art Gallery Online-Shop, the sample of End-Users becomes more complex. As mentioned earlier the existing Museum and Art Gallery Web site was designed primarily as an educational tool for school children, with secondary audience comprising academic peers and members of the public. The intended audience for the E-commerce Museum and Art Gallery Online-Shop are visitors who can purchase goods using credit cards, this excludes the Museum and Art Gallery Web sites primary audience. Therefore, it is crucial that the sample of End-Users include school children and potential customers. The reason for including children in the sample of End-Users is to ensure that the primary audience is not alienated by restriction to this proposed section of the Museum and Art Gallery Web site, thus ensuring that the primary function of the Web site is not diluted by the addition of the Online-Shop.

3.2 The Client

The Client will ultimately be the individual representing the party or body who pay for the development. While client satisfaction is important, many in Web development make the mistake of validating the deliverables using the Client. End-Users should be the primary mechanism for validating the deliverables. Substituting Clients for End-Users may make the development process a lot easier by achieving great customer satisfaction during development, however developers should ask themselves at what cost? If End-Users don't understand, don't like the system or find they cannot perform the tasks they want to, they will go elsewhere and the Client will not realise the potential of their Web presence. The Client may stop Web development or probably will go elsewhere for their next Web solution. So, its important to convince the Client that validation by End-Users is more significant than their verification of the deliverables.

Consider again our examples of the Web-based Teller System and E-commerce Museum and Art Gallery Online-Shop. The clients for the Web-based Teller System and E-commerce Museum and Art Gallery Online-Shop would be a Head Office business representative, responsible for teller business, and a senior member of staff of the Museum and Art Gallery respectively.

3.3 Domain Expert

Domain Experts provide data or content for Web applications. Web-based systems can present information relating to wheels for motor vehicles or information relating to a museum collection in the numismatics field. Unfortunately Domain Experts cannot just create data and send it to the Software Engineering and Creative Design developers. All developers must collaborate to create an End-User experience that tackles the issues raised in the Business Analysis phase. This is difficult. It requires collaboration, compromise and a lot of hard work.

It should be noted that the majority of Domain Experts reside outwith contracting organisations, and are most frequently found in different departments to the core Web development team, within in-house organisations. For example, in our Web-based Teller System a domain expert may be a branch supervisor or deputy manager, who understands the teller activities and the activities associated with those of the teller, and the context within which these activities are carried out. In our E-commerce Museum and Art Gallery Online-Shop the Domain Expert would be a member of staff in the existing Museum and Art Gallery Shop. This member of staff would be expected to contribute information relating to current shop sales information, that may be crucial to understanding the needs of potential customers.

3.4 Business Expert

In addition to providing content, Business Experts are seen to provide guidance on achieving the business objectives of the Web application and are more involved in contributing to the overall

structure of the Web presence. Business Experts should provide a pivotal role in keeping the project focused on business objectives. Business Experts also need to be able to approve the majority of decisions regarding the change of focus many projects have to overcome.

It should be noted that the majority of Business Experts reside outwith contracting organisations, and are most frequently found in different departments to the core Web development team, within in-house organisations. The Business Expert in our Web-based Teller Example would be a representative from head office who is able to make decisions regarding business objectives relating to the proposed solution and any associated Business Process Re-Engineering activities that need to be addressed. The Business Expert in the E-commerce Museum and Art Gallery Online-Shop would be the current shop manager, again who is able to inform and educate issues associated with current shop activities and functions and make any decisions relating to Business Process Re-Engineering activities that need to be addressed.

3.5 Software Engineer

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

Software Engineers not only have to be able to listen and solve the issues facing the business and domain models with software systems and software infrastructure, but they also have to advise on the impact Web and Internet technologies will have within the business and domain. For example, putting an email address on a Web site will only be effective if there is someone who is going to read and be able to respond to the message effectively. Often organisations do not have a culture of reading or being able to respond to the End-User's satisfaction via email. The Software Engineer must communicate effectively these issues to those concerned in the organisations in question. The Software Engineers involved in the Web-based Teller System will primarily comprise database developers and middleware programmers. In our other example, the E-commerce Museum and Art Gallery Online-Shop, the Software Engineer will be a programmer with a combination of database and server-side Web scripting experience.

3.6 Creative Designer

Creative Designers are from an artistic or graphic design background and are involved in creating the corporate design of the Web presence, helping to brand the look and feel of Web application. Often Creative Designers, due to their artistic background, lose sight of usability issues in favour of a more artistically creative design. All developers must be aware that what is being developed is a Web-based tool. While artistically enhanced Web presences can be a powerful asset in gaining early Client satisfaction one should never forget that End-Users will decide the

success of the project. If End-Users cannot perform the tasks they are trying to accomplish, with ease and satisfaction, they will not be happy. If you are building an Intranet site End-Users may have no option but to use the system or go elsewhere for employment, but poor user interface design costs many organisations time and money. If you are developing an Internet application then the problem is compounded by the fact that your End-Users are only seconds away from your competitors. Finding a balance between usability and creative design is a difficult task, but one that developers must address with the issues of the project and organisations involved, or risk the success of the project itself.

In both our examples of proposed Web development projects the Creative Designer will be a Web developer responsible for aesthetic aspects of the browser experience. It is essential that the Creative Designer is aware of the issues associated with building the user interface and can find the balance between function and form to achieve the most user friendly system given the project constraints. This will involve close collaboration with others in the development team and a clear understanding of the usage of the proposed system.

3.7 Team Leader

The Team Leader is responsible for coordinating and directing the development team towards the best solution available given the project constraints and characteristics. With such a wide diversity of developers to guide, Team Leader is one of the most difficult jobs one can attempt. Realistically at best one can only hope to be experienced in one of the other four developer roles: Business Expert; Domain Expert; Creative Designer or Software Engineer. With such a wide diversity of knowledge required to build most Web applications the Team Leader must respect all developers involved and appreciate the contribution that each different type of developer brings to the project. Where necessary though the Team Leader must ensure that the team make decisions regarding the development of the proposed system with the best interests of the project in question.

The following section goes on to discuss team structure and how developers should work together in order to deliver the best system possible given the criteria and characteristics of the proposed project.

4. Team Organisation, Collaboration and Communication

4.1 Individual Team Demographics, Organisation, Communication and Collaboration.

Our survey of Web engineering in practice [25 & 26] classified organisations involved in Web application development into one of three categories: contractors, outsourcers or in-house. Contractors are vendors who service Web engineering projects that are put out to tender by organisations that are classified as outsourcers. The in-house category describes organisations that primarily build their Web applications within the organisation. Throughout these different types of development organisation we found there were a number of characteristics that could help us describe the developers and teams involved in Web engineering. Given a development team of eight developers, two of the eight developers will be representing roles from the software model, two representing roles from the creative design model, one representing a role from the domain model, one representing a role from the business model and the other two developers involved in a management role. With respect to gender breakdown of a typical Web development team 70% of the developers were male, 30% being female. The average age of a developer in a Web development team was found to be twenty-six years. We also found the average number of developers comprising a team to be six, with the average length of experience in software or Web development to be less than three years.

Figure 4 identifies the AWE Process sphere of influence. All model areas reflected in the grey circle (AWE Process sphere of influence) impact and are impacted by the AWE Process. The areas marked 1 to 4 respectively indicate the number of developers roles and models that influence and are impacted by the AWE Process. Team Leaders are the exception in that they are roles that can influence and have impact in all the numbered segments represented under the AWE Process Sphere of Influence. As we can see there are very few segments in the AWE Process Sphere of Influence that do not impact two or more models. We believe that if the AWE Process is to be successfully adopted then we need to ensure communication and collaboration between the various models reflected in the AWE Process.

Consider the segment in the AWE Process Sphere of Influence marked with the number 4. This segment represents part of the development process that impacts all four models and requires input from all four models. For example, during the development of the E-commerce Museum and Art Gallery Online-Shop, the development team has to decide whether the shop will reside under a different URL from the rest of the Museum and Art Gallery Web presence. The Domain Expert is concerned that without a separate URL the online shop will not be distinctive enough from the rest of the site to attract customers. The Business Expert is concerned that a separate URL will not harness the benefits of the highly publicised existing Web presence URL and may

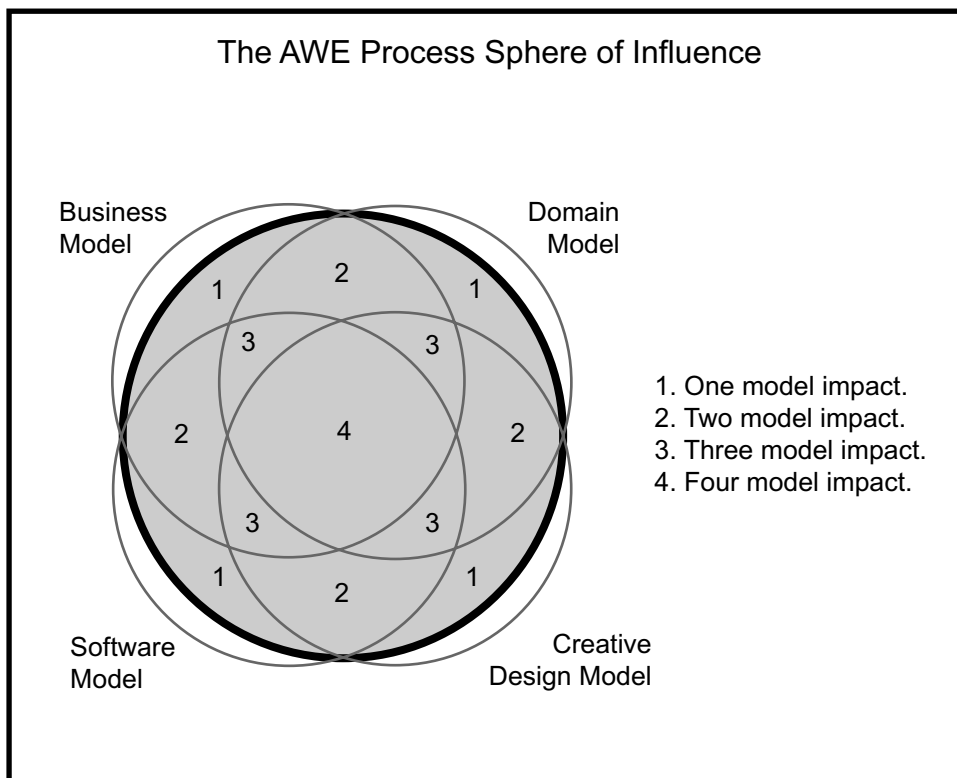


Figure 4. Shows the AWE Process Sphere of Influence, and the impact Web engineering projects have upon each of the models involved.

require additional marketing budget to publicise this new presence. The Creative Designer has spent a great deal of effort in creating an online corporate identity that is tied into the existing Web presence URL and does not want to dilute this by having to incorporate another additional URL into the corporate identity. The Software Engineer knowing that the End-Users have built up familiarity with the existing URL is concerned that a new URL may result in the Museum and Art Gallery having to rebuild the customer confidence that already exists with the current URL. There are many different issues and views to be addressed by this problem; the result could have a radical impact on all of the four models in Web engineering.

Consider the segments in the AWE Process Sphere of Influence marked with the number 3. The segment that is created by the intersection of the Software, Creative Design and Business Models may produce scenarios where the Software Engineer, Creative Designer and Domain Expert are required to develop elements of the Web interface. The segment marked with the number 2 between the Software Model and the Creative Design Model may reflect issues to be resolved between usability and aesthetic design. For example the Software Engineer may feel that the User Interface has too many colours and that End-User performance is being impeded due to the complexity of the Design. The Creative Designer may feel that losing some of the colours is diluting the corporate identity of the Museum and Art Gallery. The segments marked with the

number 1 reflect areas where there is no impact on other models. For example the selection of the version of Apache as the Web server in the segment marked 1 in the Software Model. The Software Engineer may make this decision based on previous experience with one particular version of Apache rather than on any other criteria.

Like Extreme Programming (XP) [3] the AWE process advocates a close working environment for all developers within an individual team. A close working environment, given quality developers, allows for a reduction in the formality of communication between and amongst team members. More informal communication mechanisms enable team members to affect a greater rate of development, resulting in shorter development life-cycle times. However the more informal the communication mechanisms the greater the risks of: the project losing sight of its primary objectives, little if no attention being paid to evolution and maintenance issues, individual developer personality resulting in one model overpowering project development with lack of focus on project critical issues from other models. The following discusses each of these problems in more detail, giving examples that will assist in identifying these problems, allowing teams to try and address these problems before they become fatal to the project in question.

- The more informal the communication during the development of a Web application the greater the risk of the project losing sight of its primary objectives. If staff turnover is too great or if the developers are not sufficiently self-disciplined to keep focused on the project goals then the project runs the risk of failing to address its primary objectives identified in the Business Analysis phase.
- One of the problems of using informal communication during the development of any application, is that if the development team or a proportion of the team are not going to be involved in the maintenance and evolution of the deliverables then all the information required to do so cost-effectively is lost. One of the solutions proposed by Kent Beck [3] to address this problem between outsourcing and contracting organisations is the process of in-sourcing. In-sourcing works by steadily replacing contractor development staff with development staff from the outsourcing organisation throughout the duration of project development. For example, given a project starting with nine contractors for a period of 6 months, the outsourcing organisation may demand that one of their in-house development staff join the development team at the end of each month. After 6 months when the project is being delivered the development team comprises three contractors and six in-house development staff. Thus when the project is delivered by the contracting organisation, enough of the development knowledge resides within the outsourcing organisation to satisfactorily reduce the risks involved in the outsourcing organisation maintaining the deliverables.

- One of the main problems that development teams have to be aware of is the interests of one model overpowering or dominating the entire development process. This is a problem compounded with the use of informal communication as one role with an overpowering personality can unwittingly lead the project away from the achievement of its primary objectives. All members of the development team must appreciate and realise the importance of the other roles in the development of Web-based applications. Respect for the ability and input of each developer role is required by all developers if the AWE Process is to be adopted successfully.

Clearly we believe that informal communication in a close working environment is crucial to the success of the AWE Process, but if this approach is to be adopted by your organisation then you need to ensure quality in your development staff, and awareness of the aforementioned issues.

4.2 Applying AWE to Large Web Application Development

Consider any large software engineering or Web engineering development project, with one hundred or more developers. It is usual on such a large endeavour to find the developers broken down into smaller teams with the goal of tackling a smaller sub-set of the overall project. Each team views the other development teams deliverables through a predefined or coordinated set of interfaces. Essentially each team views other teams' deliverables as a black box. Consider the Web-based Teller System and the large number of developers that were given the challenge of developing the legacy teller system. One team was responsible for the user interface, one for the middleware, one for the database design, one for testing the system and so on. Each of these teams works on a different type of problem requiring different skills and expertise. Large Web engineering projects on the other hand will split the teams down into groups responsible for different sections of a larger Web presence. These teams will perform similar tasks and we believe that each project will benefit from not only viewing every other project as a black-box, but also by also allowing orthogonal uni-discipline developer communication as illustrated in Figure 5. Orthogonal uni-discipline communication will allow inter-project sharing of resources and collaboration amongst teams developing similar approaches to common problems. This type of communication will allow developers to avoid duplication of effort and inconsistency throughout the Web presence.

Orthogonal uni-discipline developer communication can be achieved through a number of mechanisms including: communal social occasions, weekly meetings, and teleconferences. Many agile processes do not scale to large teams of developers. We believe that the nature of Web development in conjunction with orthogonal uni-discipline developer communication will allow the AWE process to be applied enterprise wide. It is important to note that Figure 5 also includes a co-ordination team. A co-ordination team is responsible for guiding the inter-team communica

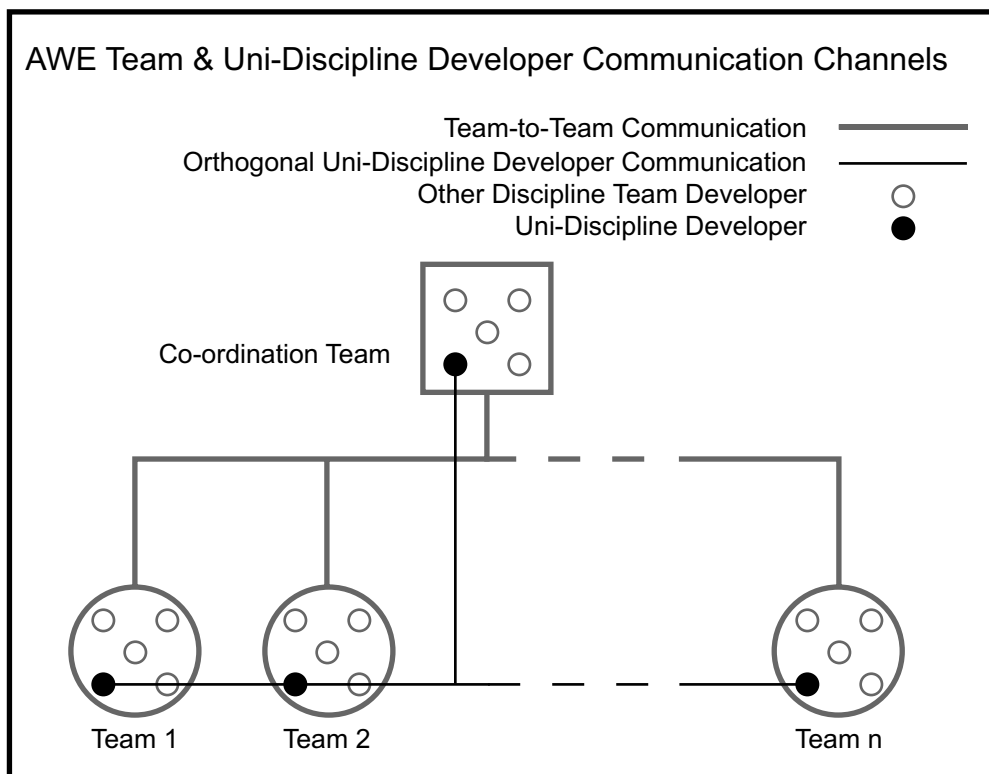


Figure 5. Shows Team and Uni-Discipline Developer Communication Channels Essential to Successfully Applying AWE to Large Web Engineering Projects.

tion process and for leading the inter-team initiatives that will enhance the objectives of the larger Web presence.

For example, consider the banking organisation involved in developing the Web-based Teller System. Since the Web-based Teller System is intended to be one of many Intranet systems that is being developed to run on branch computers, it is going to be essential that there is orthogonal uni-discipline communication between all developers working on Intranet projects. For instance, all programmers decide that they will meet once a week for lunch to discuss work related issues. During a lunch meeting it is discovered that certain types of users may be required to remember more than ten separate user names and passwords for multiple Intranet applications. Upon discovery of this problem the co-ordination team member decides to raise this issue with senior management to try and co-ordinate a single sign on initiative. This single sign on initiative is sold as a cost and effort saving initiative to Team Leaders and as a mechanism for improving End-Users usage of the proposed systems to the Clients of the respective Intranet projects.

The next section goes on to discuss techniques and how they should be integrated to support the adoption of the AWE Process.

5. Techniques and the AWE Process

Unlike many other processes, such as the Rational Unified Process (RUP) [22] that relies heavily on Use-Cases and the Unified Modeling Language (UML) [16], the AWE Process is technique independent. This is not to say that we consider techniques unimportant. On the contrary, we believe that techniques are crucial to understanding many problems and realising Web-based solutions to those problems. However many have researched and developed techniques [9, 18, 27, 35, 40, 43 & 48] that can support different tasks within Web application development. While we consider these techniques useful and powerful aids to many Web development projects, we do not believe all teams, organisations and projects are suited to any one particular technique. We believe this to be the case due to the wide diversity of projects and individuals involved in Web application development, and not due to any flaw in the techniques themselves. Instead, the AWE Process places the onus on the development team to discover, adopt, create and evaluate techniques to support the AWE Process where they feel they are required.

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

5.1 Formal vs. Informal Communication

Let us begin by describing what we mean by formal and informal communication. Formal communication comprises detailed, accurate and up to date documents that describe the Web application and the issues involved in understanding and defining project objectives, designing, building, testing, deploying and maintaining the system. Informal communication comprises less structured people-focused mechanisms for understanding and defining project objectives, designing, building, testing, deploying and maintaining the Web application. Informal communication is also valuable for sharing experience, understanding different viewpoints and team building. Informal communication takes the form of conversations, discussions and collaborative working sessions. Many in the agile community [7 & 15] believe developers work faster and better through informal communication mechanisms during project development. This argument is based on a number of assumptions: close proximity of developers to each other in a working environment; small teams, typically less than ten developers; and development teams that maintain the project deliverables. Figure 6 shows the relationship between the need for formal communication in a Web development project and close proximity of developers to each other in a working environment. We believe the further the separation, within a team of developers, in a working environment the greater the need for formal communication.

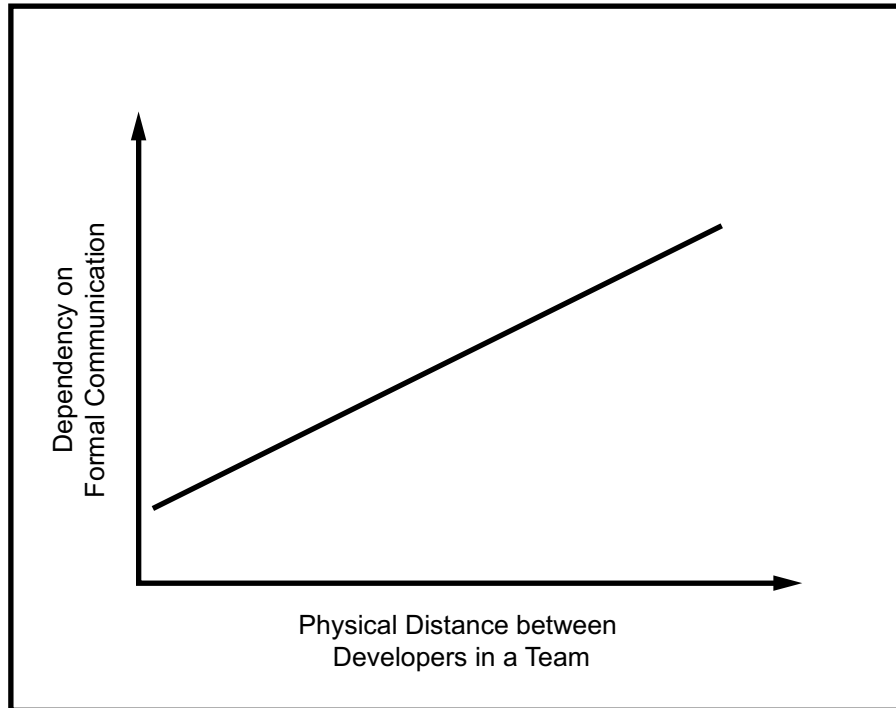


Figure 6. Shows relationship between Formal Communication and Physical Distance between developers on the same project.

Given the characteristics of Web based development teams [25 & 26], in particular the small size and the multidisciplinary nature of the teams, we believe that informal communication is a crucial factor in the success of many Web application development projects. The time to market pressures that are associated with Web-based development clearly encourage the adoption of informal inter team communication. However there is a clear conflict between time to market and the maintenance and evolutionary pressures of Web-based development. This conflict has a great impact on contracting and outsourcing organisations as the development team will often not be involved in maintaining and supporting the evolution of the Web application. Figure 7 shows the relationship between the need for formal communication in a Web development project and the time to market pressures resulting in the need for shorter project development times. It is important that teams understand that the greater the adoption of formal communication techniques the longer the project development time will be.

5.2 Multi-disciplinary and Uni-disciplinary Communication

The multi-disciplinary nature of Web development rules out the use of many existing software engineering techniques, due to their technical foundations, for multi-disciplinary team communication. However, this is not to say that we discourage the use of existing software engineering techniques, quite the contrary. Software engineering has spent many years striving to tackle generic problems in software development, much of these activities have produced well

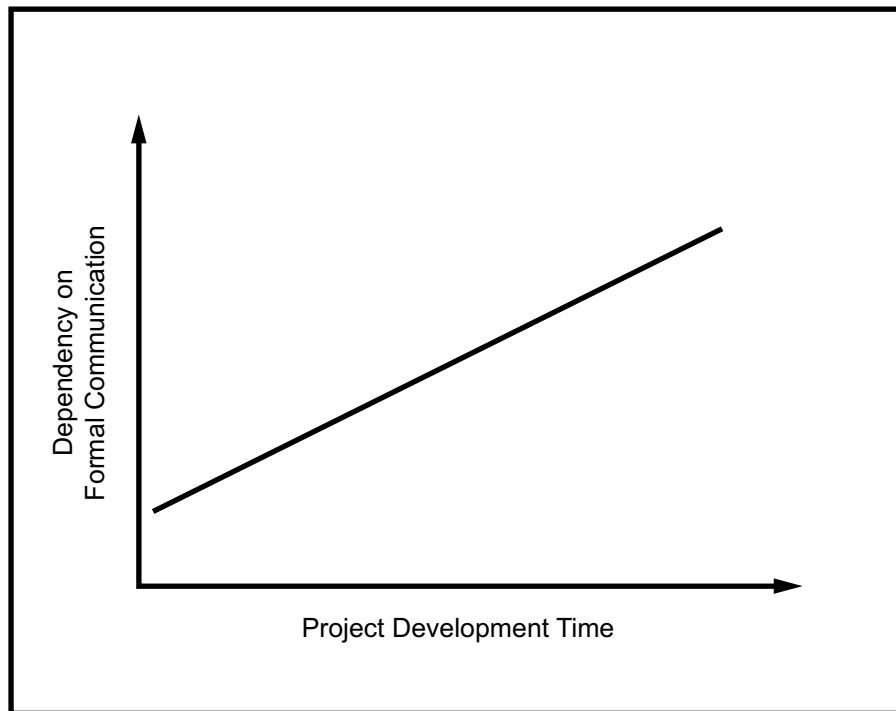


Figure 7. Shows relationship between Formal Communication and the increase in Project Development Time

proven researched techniques that support common tasks in the software model in Web engineering. However, most of the software engineering techniques will only prove useful during uni-disciplinary communication sessions for the software model. This still leaves the problem of supporting multidisciplinary communication and other uni-discipline communication sessions in the creative design, business and domain models. With respect to other uni-discipline communication, there will probably be existing techniques that can be adopted or tailored to support the AWE Process.

We do not encourage or discourage informal or formal communication mechanisms. The diversity of development environment and business and domain models that make up Web-application development mean that the development team will have to take on the responsibility of discovering, adopting, creating and evaluating techniques to support AWE. It is crucial that developers understand the responsibility that AWE places on their shoulders with respect to technique integration. Successful adoption of AWE without this understanding will only be applicable to small, trivial Web engineering projects. It is important that developers are aware that if their project shows any of the following characteristics: team members not in close working proximity to each other; development teams that are not going to maintain the project deliverables; number of developers greater than ten, then serious effort will have to be given to getting techniques that work for your project and the adoption of the AWE Process.

5.3 Adopting, Creating, Tailoring and Evaluating Techniques.

We believe that iterative and incremental development life-cycles apply themselves well to Web engineering. Initially each project should come up with a set of criteria that identify all the project risks that can be reduced by techniques. Focusing initially during the first iterations on the highest risk objectives of the project, each team and developer is responsible for contributing and collaborating on the task of successfully integrating techniques that will work with the AWE Process. The rapid change within the business, domain and software models within Web engineering, and the major impact change has between all models, requires constant review and evaluation of the techniques being used. It is essential that each team constantly review, ideally after each AWE Process life-cycle iteration, the contributions that each technique is making to the project. Figure 8 shows a typical iterative and incremental development AWE project life-cycle indicating where technique evaluation should be carried out. The iterative and incremental style of the AWE Process life-cycle is based upon the Boehm's Spiral Model [5]. See Section 2 of this report for a more detailed description of the AWE Process life-cycle. During evaluation where a technique is observed to be working in intra team communication (amongst a single team), continue to use it, ensuring that you inform other teams through inter team communication (between many different teams) to reduce other team and developer learning curves. When techniques appear to be problematic, review what works and what doesn't. If the technique can

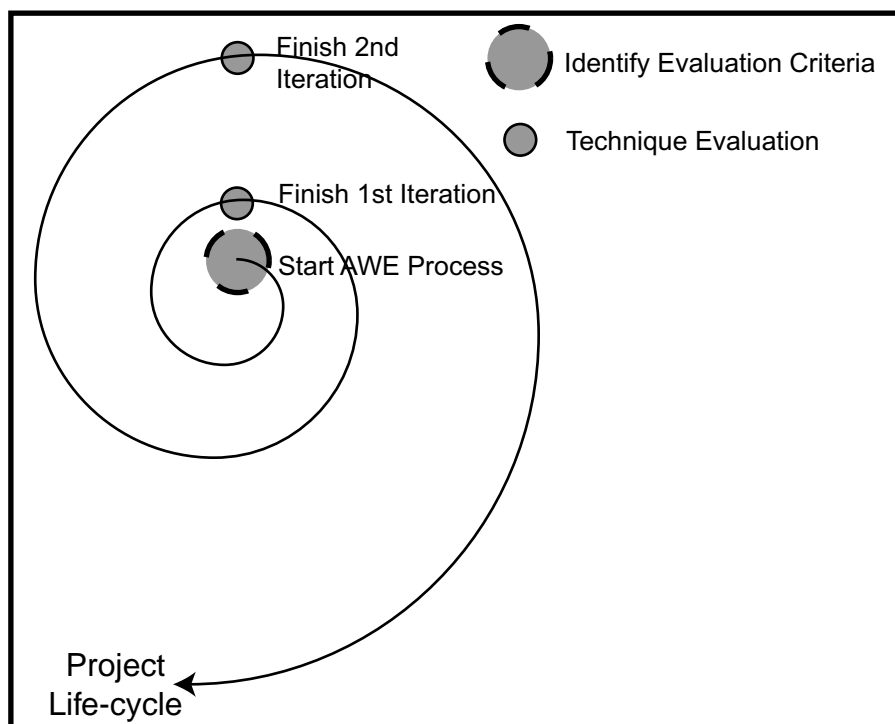


Figure 8. Shows where Identification of Evaluation Criteria and Technique Evaluation Stages should occur on an iterative and incremental Project Life-cycle.

be altered and the team believes that the alterations indicate successful adoption within the project, continue to use it, else get rid of the technique and look for another to take its place if required. Again, remembering to inform others through inter team communication channels of your decision to drop a technique and why.

For example, the developers in a team on the Web-based Teller System example decide to use UML [16] to model the proposed Web application. UML is seen to be ideal as it has very close integration with a number of CASE tools. After the first iteration the team agree that UML on its own is not completely suitable for modeling Web applications. Upon further investigation the team discover the Web Application Extension (WAE) [9] for UML. The team agrees that they will use UML again on the second iteration using the WAE for UML. The second evaluation session, after the second life-cycle iteration, reveals greater satisfaction with the WAE for UML, and the team decide to adopt it as a project standard. Then using inter team communication the Web-based Teller System development team inform other Intranet project teams in the bank of their findings with UML and the WAE.

The next section goes on to discuss architectural issues, including usability and user-interface design, in Web engineering and how these affect the AWE Process.

6. Architecture

One of the twelve principles of the agile manifesto states, “The best architectures, requirements and designs emerge from self-organizing teams”. In a way, we agree and disagree with this principle when applied in the context of Web application development. Two major issues require a degree of architectural understanding and planning in Web application development: the complexity involved in evolving a small Web application into a truly large Web presence, and the rapid change in underlying technological infrastructure used to build Web applications.

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

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

Gartner [6] classify Web applications into one of two categories, opportunistic or systematic. Opportunistic Web applications are small, less significant, Web-based solutions that have little or no potential for expansion compared with systematic Web applications. An example of an opportunistic Web application would be a conference Web site, where structure and content may change, but at such a rate not to require detailed architectural planning. This does not mean that opportunistic Web applications can ignore the architectural issues, just that the significance of architectural issues is lessened in comparison with systematic Web-based systems. An example of a Systematic Web application would be an Internet banking application, where underlying infrastructure will have to expand to support the exponential growth rate of users, functionality and information. The rest of this section goes on to discuss each of these issues in some detail, outlining the concerns development teams must address to ensure scalability, portability, reuse, and a satisfactory End-User experience of Web applications.

6.1 Usability Issues

Many Web sites have evolved into extremely large presences over the past decade. The growth of

information and functionality in many Web applications makes the task of managing each End-User experience extremely difficult. Ensuring that information is easy and intuitive to ascertain, and applications provide functionality that satisfies End-Users usage demands on a systematic Web application is something that we believe requires a good deal of planning and forethought. It is particularly important for those involved in large Web application development to address these issues, primarily through the co-ordination team. All developers must be aware of the importance of creating a user-friendly experience for every type of End-User of your Web application.

Understanding End-Users' usage of your Web Application is crucial to successfully realising the potential of your Web-based endeavors. A recent survey by Jakob Nielsen [33] calculated that as much as 44 percent of potential e-commerce sales are not completed because users cannot use the site. In addition, Nielsen goes on to describe the lack of adherence to usability guidelines [36], stating that most Web sites comply with only one third of usability guidelines. The report suggests that if usability was improved on the average e-commerce site, there is a potential to increase sales by up to 76 percent.

Consider the E-commerce Museum and Art Gallery Online-Shop development project. The Business Expert is keen to get as much information for marketing purposes regarding customers as possible. In order to satisfy this business objective, the team decide to force the End-Users to fill in a detailed form before each transaction can be completed. During evaluation of the application the team soon observe that having to fill in the form is frustrating and in conflict with End-Users usage requirement, which is to complete an online transaction. As a result more than fifty percent of potential sales are lost as the majority of End-Users decide that they will not fill in the form. After evaluating the results the team decide to make filling in the marketing form optional and offer a 5% discount to all End-Users who take the time to fill in the form. After a second evaluation of the Web application the team find that End-User satisfaction is much greater since the mandatory marketing form was made optional. In addition it is discovered that the information gathered is much more accurate than that collected in the first evaluation where the form was mandatory.

It is important to get good involvement with End-Users in order to successfully adopt the AWE Process. You will only truly achieve the potential of large Web-based solutions if you are able to understand and satisfy your End-Users usage requirements. There is a large collection of published literature [10, 11 & 31] that can assist teams with the challenges of dealing with usability and Human Computer Interaction (HCI) issues. HCI is a large field in its own respect, and beyond the scope of this publication. However, be aware, it is your responsibility as an individual, team, and organisation to tackle the challenges presented by End-Users, who are ultimately the litmus test for success.

End-Users, if used and interacted with correctly [34], can contribute enormous benefits to Web engineering projects. However, it is also important to be aware of the costs involved in dealing with End-Users. End-User validation is time consuming and can be expensive [12]. Only through understanding and a commitment from everyone as to the importance of End-User involvement, will AWE allow you to harness the true benefits of Web-based developments.

6.2 Infrastructure Support

Recently there has been a move towards Web-based enterprise frameworks¹ in the enterprise application development community. Examples include Sun Microsystems's Java 2 Enterprise Edition (J2EE) platform [45-47] and Microsoft's .NET framework [28 & 29]. The major goal of these initiatives is to provide a mechanism that will allow infrastructure independence from hardware, operating systems and to some degree software infrastructure (database, application server, Web container, Web server). If strictly adhered to these enterprise frameworks can help ensure: portability; scalability; lower total cost of ownership; easier skills transfer amongst developers; vendor independence; platform and software infrastructure independence. Be aware though, if one truly wants to develop Web-based solutions that have these characteristics, then other standards initiatives have to complement those of enterprise framework adoption. Markup language standards, for example HTML 3.2 & XML 1.0, and client-side browser scripting standards for Java Applets, are important to ensure greatest possible browser compatibility, which ultimately allows you to deliver to a wider potential market. In addition, Structured Query Language standards such as SQL92 and SQL98 will help ensure database flavour independence. In addition avoiding stored procedures will also help ensure database flavour independence, as they can often be platform or RDBMS proprietary.

The co-ordination team in the banking organisation that is developing the Web-based Teller System decides to adopt J2EE as its enterprise framework. Where possible teams are encouraged to stick within the J2EE specification. During the development the Web-based Teller System the banking organisation buys another bank and decides to expand the project to include replacing the newly purchased banks teller system. Upon further investigation the development team discover that the newly purchased bank has a completely different backend to the existing bank. However, as the developers have stuck to the J2EE framework and have kept their SQL to the SQL92 standard, interfacing with the new backend system is a far easier job than was expected.

If your goal is to produce opportunistic Web applications then these issues are less of a concern, however ignore them at your peril! For those who are embarking upon or are involved in systematic Web application development, you will want to address these issues and ensure that you can

¹ *An Enterprise Framework is a series of technologies, standards and processes, with supporting software infrastructure to help organisations develop solutions for the enterprise.*

harness the benefits of architectural planning. We do not wish to get into a religious debate regarding J2EE and .NET, the AWE Process will work with both these enterprise frameworks, although the techniques you chose to support AWE may vary depending on your enterprise framework. Figure 9 illustrates an 4-tier Web-based architecture, indicating the role of enterprise frameworks and guidance to help identify some of the previously mentioned issues. It should be noted that the architecture described in figure 9 is a utopian one that is difficult to achieve in the enterprise. Often there are many legacy systems (CORBA and CICS applications for example) and multiple database flavours to contend with in addition to external systems that are often interfaced with through a messaging system. While it may be difficult to achieve your ideal architecture, it is important that you at least have an ideal architecture in sight, else you severely reduce you chances of achieving one.

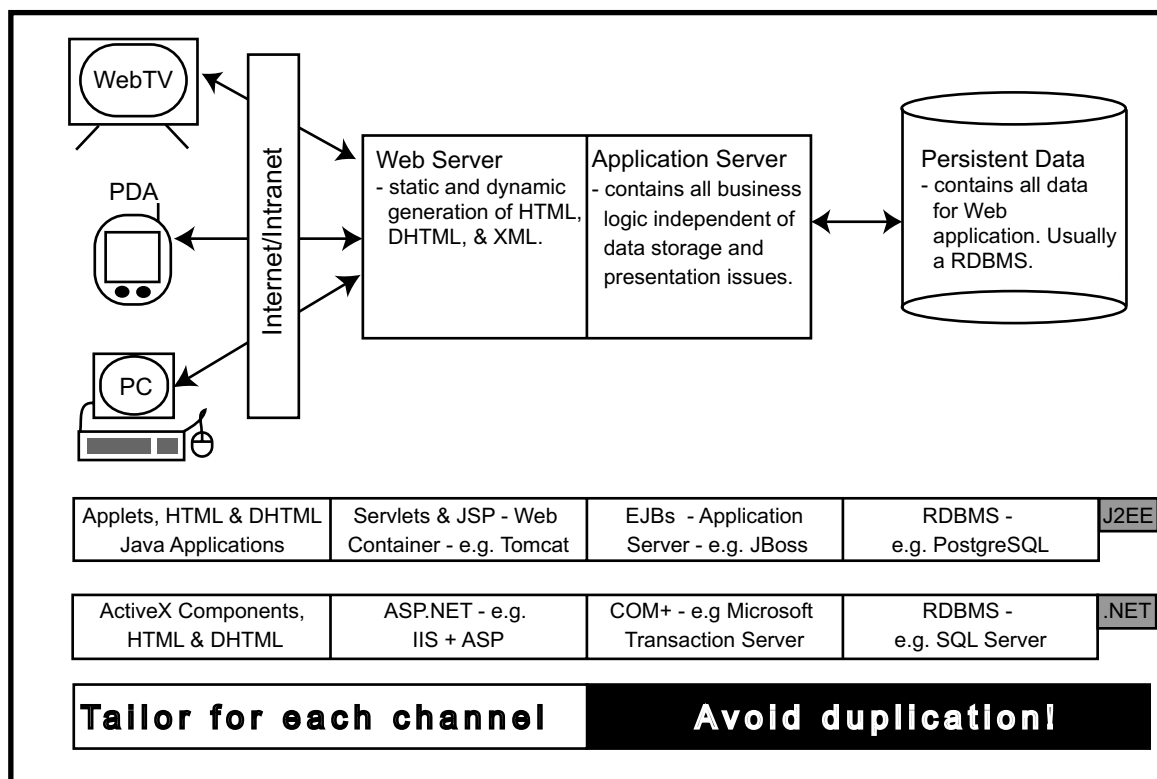


Figure 9. N-tier architecture showing ideal separation of Web application architecture with reference to .NET and J2EE.

Remember you are responsible for your architecture. You will get from your architecture what you put into it, as the saying goes “An enterprise without architecture is like a ship without a rudder - it will move but you cannot guide it.”, Source - META group.

The following four sections of this report detail our conclusions, acknowledgements, glossary and references respectively.

7. Conclusions

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

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

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

The development of Web applications requires multidisciplinary development teams. In order to scale AWE to large Web engineering projects it is essential that orthogonal uni-discipline communication lead by a co-ordination team, as discussed previously, occurs to ensure a consistent End-User experience and the avoidance of effort duplication. This again is one of the biggest challenges to the adoption of AWE on many Web-based projects, due to the geographical spread of developers and corporate inertia within many organisations.

End-Users are the litmus test for success. Developing Web-based systems without close involvement with End-Users is in our opinion flawed. It is essential that all developers are focused on

the End-Users of the proposed system and try to provide the best usage experience given the project resources available. Indeed, the evolution of Web-based systems requires a great degree of architectural focus. Web applications require architectural focus not only on usability but also on the application design, to try to ensure a scalable, robust, reliable and evolvable deliverable.

Remember, you are responsible for the success of your Web application development endeavours. If you were wishing for a cookbook recipe that offers guarantees of success, then not only are you looking in the wrong place, but we hope from reading this document that you have realised that such a wish is fantasy. You are responsible for solving your Web-based problems. We have merely tried to provide a road map for your journey. We hope the AWE Process helps you on your way.

7.1 Further Work

We are currently involved in testing the AWE process in a commercial environment. The period of study will last from October 2001 until September 2002. We hope to be able to provide initial reports of the success of AWE by early summer 2002, with a more detailed technical report titled 'Adopting The AWE Process' due in September 2002.

8. Acknowledgements

We would like to take this opportunity to thank Dr. Noel Winstanley and John B. Wilson for their comments. In addition we would like to thank our families for their understanding and support.

9. Glossary

.NET	.NET is an enterprise framework based around the Microsoft technology suite.
Agile Process	A term applied to a number of lightweight methodologies that have emerged since the mid 1990s. These methodologies value: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation and responding to change over following a plan. There are a number of Agile Processes, including: Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development and AWE.
Business Expert	A stakeholder responsible for business issues in a development team using the AWE Process.
Business Model	The Business Model reflects issues associated with business objectives during Web or software engineering endeavours.
Client	The Client is a stakeholder that represents the body or bodies who are paying for the project development.
Contractor Organisation	An organisation that tenders and services contracts for Web application development, that have been put out to tender by Outsourcing Organisations.
Creative Design Model	The Creative Design Model reflects issues associated with the aesthetic aspects of the user interface on a Web or software engineering project.
Creative Designer	A stakeholder responsible for aesthetic issues in a development team using the AWE Process.
Domain Expert	A stakeholder responsible for issues associated with the application domain to which Web applications are being applied. For example bank branches in the Web-based Teller System.
Domain Model	The Domain Model reflects issues associated with the domain to which the objectives reflected in the business models and the software model are to be applied during Web or software engineering projects.

E-commerce Museum and Art Gallery Online-Shop	A hypothetical example of an Internet application development project to add an online shop to a large cultural heritage Web site.
End-Users	Stakeholders representing the users of a proposed system in a development project using the AWE Process.
Enterprise Framework	An Enterprise Framework is a series of technologies, standards and processes, with supporting software infrastructure to help organisations develop solutions for the enterprise.
Formal Communication	Communication conducted through paper or electronic documents.
Heavyweight Process	A term used to describe Monumental Processes. Derived from the perceived heavy volume of documented deliverables produced by projects using Monumental Processes and their rather heavyweight management style.
Informal Communication	Communication conducted through face-to-face interaction.
In-house Organisation	An organisation that develops all its Web applications using developers from within its own organisation.
In-Sourcing	A technique used by Outsourcing Organisations to ensure In-house development expertise on projects outsourced to Contracting Organisations. In-Sourcing works by steadily replacing contractors with outsourcing organisation employees during the project development life-cycle.
Inter Team Communication	Communication, whether formal or informal, between different development teams on the same Web presence or Web development project.
Intra Team Communication	Communication, whether formal or informal, between developers in the same development team.
Java™ 2 Enterprise Edition (J2EE)	J2EE is an enterprise framework based around the Java programming language.

- Lightweight Process** A term often used to describe Agile Processes. Derived from the light weight nature of the documented project deliverables produced from projects using Agile Processes.
- Monumental Process** Monumental Processes are traditional software engineering processes that are heavily influenced by the engineering disciplines. Monumental Processes include the Rational Unified Process and Structured Systems Analysis and Design Method.
- Orthogonal Uni-discipline Communication** Communication, whether formal or informal, between similar types of developer in different development teams on the same Web presence or Web development project.
- Outsourcer** An organisation that uses Contracting Organisations to help develop their Web applications.
- Predictive Process** Predictive Processes are deterministic in nature, identifying very early in the development life-cycle the problem and proposed solution.
- Software Engineer** A stakeholder responsible for technical issues in a development team using the AWE Process.
- Software Model** The software model reflects the technical issues associated with the development of a software or Web applications.
- Team Leader** A stakeholder responsible for team guidance and project management issues in a development team using the AWE Process.
- The Agile Web Engineering (AWE) Process** An Agile Process for the development of Web-based applications.
- Web-based Teller System** A hypothetical example of an Intranet application development project in a banking organisation to replace a legacy teller system in bank branches.

10. References

- [1] Barry C. & Lang M., 'A Survey of Multimedia and Web Development Techniques and Methodology Usage', *IEEE MultiMedia*, April-June 2001, Vol. 8 No. 2, Page(s): 52-61.
- [2] Beck K. et al., 'Manifesto for Agile Software Development', *The Agile Alliance*, February 2001, <http://www.agilealliance.org/>
- [3] Beck K., 'Extreme Programming Explained', *Addison-Wesley*, 1999. ISBN: 0201616416.
- [4] Benington H. D., 'Production of Large Computer Programs', *Proceedings Symposium on Advanced Programming Methods for Digital Computers*, 28-29 June 1956, *Republished in Annals of the History of Computing*, October 1983, Page(s): 350-361.
- [5] Boehm B. W., 'A Spiral Model of Software Development and Enhancement', *IEEE Computer*, Volume 21 5, 1988, Page(s): 61-72.
- [6] Calvert M., Smith D. & Natis Y., 'Management Update: Don't Waste Money on Application Server Overcapacity', *Gartner, Inc.* Report Code: IGG-08292001-04, 29 August 2001.
- [7] Cockburn A., 'Characterizing People as Non-Linear, First-Order Components in Software Development', *Humans and Technology Technical Report*, TR 99.05, Oct.1999 7691 Dell Rd, Salt Lake City, UT 84121 USA, <http://members.aol.com/humansandt/papers/nonlinear/nonlinear.htm>
- [8] Coda F., Ghezzi C., Vigna G. & Garzotto F., 'Towards a Software Engineering Approach to Web Site Development', *Proceedings of the 9th International Workshop on Software Specification and Design, 20th International Conference on Software Engineering*, Kyoto (Japan), April 1998, Page(s): 8-17.
- [9] Conallen J., 'Building Web Applications with UML', *Addison Wesley*, 1999. ISBN: 0-201-61577-0
- [10] Constantine L. L. & Lockwood L. A. D., 'ForUse: The Web site for practitioners of usage-centered design', *Constantine & Lockwood Ltd.*, September 2001, <http://www.foruse.com/>
- [11] Constantine L. L. & Lockwood L. A. D., 'Software for use', *Addison Wesley*, January 2000. ISBN: 0-201-92478-1

- [12] Constantine L. L., 'Lightweights, Heavyweights and Usable Processes for Usable Software', *Keynote at Software Development 2001*, San Jose, 19 April 2001.
- [13] Downs E., Clare P. & Cole I., 'Structured Systems Analysis and Design Method: Application and Context', *Prentice Hall International (UK) Ltd.*, 1992. ISBN: 0-13-853698-8.
- [14] Dynamic Systems Development Method (DSDM) Limited, 'DSDM homepage', *Dynamic Systems Development Method Limited*, September 2001, <http://www.dsdm.org/>
- [15] Fowler M. & Highsmith J., 'The Agile Manifesto', *Software Development Magazine*, August 2001, <http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm>
- [16] Fowler M. & Scott K., 'UML Distilled: Applying the Standard Object Modeling Language', *Addison-Wesley Object Technology Series*, *Addison-Wesley*, 1997, ISBN: 0-201-32563-2.
- [17] Fowler M., 'The New Methodology', *MartinFowler.com*, March 2001, <http://www.martinfowler.com/articles/newMethodology.html>
- [18] Gaedke M., Gellersen H-W., Schmidt A., Stegemüller U. & Kurr W., 'Object-oriented Web Engineering for Large-scale Web Service Management', *Proceedings of the 32nd Hawaii International Engineering for Large-scale Web Service Management*, *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999, Page(s): 1-9.
- [19] Hammer M., 'Reengineering Work: Don't Automate, Obliterate', *Harvard Business Review*, 1990, Page(s): 104-112.
- [20] Hammer M. & Champy J., 'Reengineering the Corporation', *Nicholas Brealey Publishing Ltd, Revised Ed.*, 23 August 2001, ISBN: 1857880978
- [21] Highsmith J. A., 'Adaptive Software Development: A Collaborative Approach to Managing Complex Systems', *Dorset House Publishing*, December 1999. ISBN: 0932633404.
- [22] Jacobson I., Booch G. & Rumbaugh J., 'The Unified Software Development Process', *Addison-Wesley*, 1999. ISBN 0201571692.
- [23] Jacobson I., Christenson M., Jonsson P. & Övergård G., 'Object-Oriented Software Engineering: A Use Case Driven Approach', *Addison-Wesley Publishing Co.*, 1992. ISBN: 0-201-54435-0.

- [24] Jeffries R. E. et al., 'XProgramming.com: an Extreme Programming Resource', September 2001, <http://www.xprogramming.com/>
- [25] McDonald A. & Welland R., 'A Survey of Web Engineering in Practice', *Department of Computing Science Technical Report R-2001-79*, University of Glasgow, Scotland, 1 March 2001.
- [26] McDonald A. & Welland R., 'Web Engineering in Practice', *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, 2 May 2001.
- [27] Mendez E., Mosley N. & Counsell S., 'Web Metrics-Estimating Design and Authoring Effort', *IEEE Multimedia Web Engineering Part 1*, January-March 2001, Page(s): 50-57.
- [28] Microsoft Corporation, '.NET Homepage', 23 September 2001, <http://www.microsoft.com/net/>
- [29] Microsoft Corporation, 'A Guide to Reviewing the Microsoft .NET Framework: A Platform for Rapidly Building and Deploying XML Web Services and Applications to Solve Today's Business Challenges', White Paper, September 2001, <http://www.microsoft.com/net/>
- [30] Murugesan S., Deshpande Y., Hansen S. & Ginige A., 'Web Engineering: A New Discipline for Development of Web-based Systems', *Proceedings of the First ICSE Workshop on Web Engineering*, 16-22 May 1999, <http://fistserv.macarthur.uws.edu.au/san/icse99-webe/ICSE99-WebE-Proc/default.htm>
- [31] Nielsen J. & Norman D., 'The Nielsen Norman Group Web site', *The Nielsen Norman Group*, September 2001, <http://www.nngroup.com/>
- [32] Nielsen J., 'Designing Web Usability: The Practice of Simplicity', *New Riders Publishing*, Indianapolis, 2000, ISBN: 1-56205-810-X.
- [33] Nielsen J., 'Did Poor Usability Kill E-Commerce?', *Jacob Nielsen's Alertbox*, 19 August 2001, <http://www.useit.com/alertbox/20010819.html>
- [34] Nielsen J., 'First Rule of Usability? Don't Listen to Users', *Jacob Nielsen's Alertbox*, 5 August 2001, <http://www.useit.com/alertbox/20010805.html>
- [35] Nielsen J., 'Homepage Usability: 50 Websites Deconstructed', *New Riders Publishing*, Indianapolis, November 2001, ISBN: 0-73571-102-X.

- [36] Nielsen, J., Molich, R., Snyder, C., & Farrell, S., 'E-Commerce User Experience'. *The Nielsen Norman Group*, Fremont, CA, USA, 2000, ISBN: 0-9706072-0-2.
- [37] Pressman R. S., 'Can Internet-Based Applications Be Engineered?', *IEEE Software*, Volume: 15 5, September/October 1998, Page(s): 104-110.
- [38] Pressman R. S., 'What a Tangled Web We Weave', *IEEE Software*, Volume: 17 1, January/February 2000, Page(s): 18-21.
- [39] Ramsay M. & Nielsen J., 'WAP Usability Déjà Vu: 1994 All Over Again, Report from a Field Study in London, Fall 2000', *The Nielsen Norman Group*, December 2000, Page(s): 4, <http://www.nngroup.com/reports/wap>
- [40] Reifer D. J., 'Web Development: Estimating Quick-to-Market Software', *IEEE Software*, Volume: 17 6, November/December 2000, Page(s): 57-63.
- [41] Royce W. W., 'Managing The Development of Large Software Systems: Concepts and Techniques', In *WESCON Technical Papers*, v. 14, pages A/1-1-A/1-9, Los Angeles, August 1970. WESCON. Reprinted in *Proceedings of the Ninth International Conference on Software Engineering*, 1987, pp. 328-338.
- [42] Schneider G. & Winters J. P., 'Applying Use Cases: A Practical Guide', *Addison-Wesley Object Technology Series*, Addison-Wesley, 1998, ISBN: 0-201-30981-5.
- [43] Schwabe D., Esmeraldo L., Rossi G. & Lyardet F., 'Engineering Web Applications for Reuse', *IEEE Multimedia Web Engineering Part 1*, January-March 2001, Page(s): 20-32.
- [44] Shneiderman B., 'Designing the User Interface: Strategies for Effective Human-Computer Interaction', *3rd Edition*, Addison-Wesley, ISBN: 0201694972, July 1997.
- [45] Sun Microsystems, Inc., 2001, 'Java™ 2 Platform Enterprise Edition Web site', September 2001, <http://java.sun.com/j2ee/>
- [46] Sun Microsystems, Inc., 'Java™ 2 Platform Enterprise Edition Specification, v1.2', 23 November 1999, <http://java.sun.com/j2ee/download.html#platformspec>
- [47] Sun Microsystems, Inc., 'Java™ 2 Platform Enterprise Edition Specification, v1.3', 27 July 2001, <http://java.sun.com/j2ee/download.html#platformspec>

[48] Ward S. & Kroll P., 'Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process', *Rational Software Corporation*, July 1999, <http://www.rational.com/>