Design and Empirical Evaluation of an Agile Web Engineering Process

Américo Sampaio[1], Alexandre Vasconcelos[1], Pedro R. Falcone Sampaio[2]
[1]Centro de Informática, Universidade Federal de Pernambuco - Recife, Pernambuco Brazil
[2]Computation Department, University of Manchester Institute of Science and Technology United Kingdom
e-mail: americo.sampaio@ufpe.br, amlv@cin.ufpe.br, p.sampaio@co.umist.ac.uk

Abstract

Web applications present important characteristics that must be properly addressed by software engineering methods, processes and techniques. In most cases, Web systems have to be delivered complying with severe time constraints due to its strategic nature for the client's business. In this work we describe an agile process for Web-based application development, XWebProcess, which is based on Extreme Programming and aims at building high quality Web applications in a time effective way. The process is described using the SPEM meta-modeling language to facilitate its understanding and further improvements. An experiment was conducted to assess the time effectiveness and the quality of the proposed process. The results have shown that XWebProcess is equally agile when compared to Extreme Programming, moreover, surveys conducted as part of the experiment pointed out that XWebProcess is suitable to Web development in dimensions such as requirements gathering, user interface and navigation design, and software testing.

1. Introduction

Accelerating application development and reducing time to market is a highly valuable feature of a software process. For web-based applications, in particular, project development efforts often need to comply with severe time constraints imposed by the strategic business importance of the web. In many cases, development teams often resort to "short cuts" to accelerate the development process, adopting ad-hoc approaches to build web applications [4]. In these situations the success of the project relies heavily on the skills and knowledge of the people in the software team, with the usual negative side effects on flexibility, documentation quality and robustness of the application. Therefore, it is highly important to have a more rigorous and organized approach to build high quality web applications while retaining agile properties in the development effort.

In this paper we describe XWebProcess [3,32], an agile process for web-based application development. XWebProcess is grounded on the principles underlying Extreme Programming and is aimed at building high quality web applications in a time effective way. XWebProcess has been tailored to deal with important issues concerning web development such as: complex user interfaces and navigation; non-functional requirements (distribution, concurrency, load balancing), testing, and infrastructure support.

The agility of XWebProcess has been compared to XP via an experimental setting which shows its superiority in supporting web development dimensions such as requirements gathering, user navigation design, and software testing, while retaining the agile properties of Extreme Programming.

Experimental results described in this paper help to validate the core features of XWebProcess and also have an important impact considering the limited availability of

literature assessing web application development processes empirically. Within the literature, a substantial number of contributions claim that their proposed methods facilitate the construction of software on time, however, without providing empirical evidence. This paper will not only provide quantitative data about the time effectiveness of XWebProcess, but will also present qualitative data surveying XWebProcess' effectiveness in supporting web application development.
The remainder of this paper is organized as follows. Section 2 introduces basic concepts for agile processes. Section 3 presents an overview of XWebProcess describing how the process was created and modeled. Section 4 describes the experiment and survey conducted to analyze XWebProcess empirically. Section 5 presents related work and a comparison of existing Web development processes. Finally, section 6 presents a summary and future work.

2. Agile Processes and Extreme Programming

There is a growing concern that the development efficiency of traditional software process models is affected by too much emphasis placed on documenting and following rigid plans [1,2,9,10,11]. To address the need for speeding up software development while retaining software quality imperatives, several key ideas have been developed by the agile methods community and a synthesis of the ideas can be found in the "Agile Manifesto" [6]. The core elements of the Manifesto are:

- *Individuals and interactions* over processes and tools;
- *Working software* over comprehensive documentation;
- *Customer collaboration* over contract negotiation;
- *Responding to change* over following a plan.

The perspectives on the left are regarded as more important than the items on the right. Therefore, agile processes, like Extreme Programming (XP) [1,2], DSDM [7] and Crystal [8] focus on constant delivery of functionality and interaction among team members and clients. The main goal is to deliver working software in a fast pace, without having to focus on documentation and plans. These processes are achieving encouraging results specially when requirements change frequently, development cycles are short and the development team is small [9,10,11].
Within the spectrum of agile methods available, Extreme Programming (XP) became the most popular and is now employed in several large software development organizations. XP relies on four guiding values: communication, feedback, simplicity, and courage.
Communication states that members of the team should interact constantly to solve problems and to discuss important issues. The client, or someone who represents him, should work alongside the development team explaining the requirements and business rules.
Feedback states that each team member (programmer, manager, client, etc.) should give constant feedback on the problems found in order to correct them quickly.
Simplicity is defined in [1,2] as selecting the easiest (most simple) solution that works. XP aims at producing simple design and implementation solutions without excessive overkill relating to changes that can occur in the future. Although this is highly controversial, some practices of XP like refactoring and automated testing often have a positive effect in reducing possible problems.
Courage in XP relates to bold measures such as: refactoring large parts of code when necessary, throwing low quality code away, and changing outdated requirements frequently.

XP also adopts practices that guide how development teams comply with the values mentioned above. The power of XP´s practices should not be evaluated in isolation but as a whole, i.e. the weakness of some practices are minimized by others, and the group as a whole provides a powerful way to build software. An overview of some XP practices is described below. For a comprehensive description see [1,2].

The Planning game: A Release in XP should be as small as possible (2-3 months). Each release is divided in iterations (2-3 weeks), where story cards (simple use-cases) are implemented. The effort estimated to implement each story is measured by the programmers with the client's assistance. In XP the client works together with the programmers and is responsible for setting priorities for the stories that are going to be implemented. The estimates rely on past experience and as the project evolves, the team gets to know their speed, improving estimates.

Pair Programming: All code is produced by a pair of programmers working side-by-side. While one programmer focuses on coding and algorithms, the other suggests improvements and prevents possible errors. The pairs are changed frequently, enabling each participant to play different roles and work in different stories. The reason for doing so is to enable every team member to have an overall view of the system avoiding possible problems in case someone leaves the team.

Refactoring: Is a series of improvements, suggested in [12], that can be done in the code without changing its functionality. Associated with *Automated Testing*, another XP practice, the quality of the code is maintained, since every time a new functionality is produced all tests must be checked (100% should pass). So refactoring prevents the code from being badly structured, while testing prevents errors from being introduced.

The Collective Code Ownership states that functionality can be altered or refactored by any team member. Having a *Coding Standard* that is followed by all team members facilitates the task of working with different functionalities frequently and benefits maintenance.

3. XWebProcess

A software process is defined as a set of activities undertaken to develop, maintain and manage software systems [28]. These activities can be composed by other activities and are performed by individuals who have a certain role in the process (developer, manager, customer) and can use tools and models to automate and facilitate the task undertaken. As the process progresses, artifacts (source code, documents, models) are produced and updated and serve as input to the construction of other artifacts.

Some core elements of web applications that have to be considered in a web engineering process are [16,29,30]:

- Non-functional requirements such as concurrency, load balancing, security and distribution, which play an important role in the operation of web applications. The number of concurrent users can be substantial and appropriate software and hardware infrastructure may also be necessary.
- Different kinds of clients accessing the system via distinct browsers and protocols. Therefore it is important to identify different OS and hardware architectures underlying the system.
- Navigation and presentation aspects. In web applications, user interfaces often contain graphics, animations, links and text requiring attractive UI designs and simple navigability.

Evidence from web projects [16] also points out to other factors relevant to web application development processes: web systems are constantly changing and being integrated with other systems; web projects have short development cycles - typically three

to six months; and web systems are built by a small multidisciplinary team - typically 3 to 10 people.

XWebProcess seeks to combine the key elements of Extreme Programming with process structures tailored to tackle the characteristics and factors of web engineering identified above, therefore providing an efficient and effective approach to the construction of web applications.

The main reasons for leveraging Extreme Programming as the base software process are:

- XP is an agile process that is showing positive results in many software development projects. A recent survey [17] presents some important quantitative data about a large number of XP projects built by different software companies worldwide. The projects varied in size, kind of application and application domain. Almost all XP projects finished on time and on budget and, among them, 28% represented web projects.
- Recent contributions [29,30,31] acknowledge the value of XP for web application development while recognizing the need for adaptations in the XP process to better suit the web application problem landscape.
- The wide availability of literature and project case studies simplifying the task of best practice identification.

The creation of XWebProcess can be summarized in four key steps. First, the must have features of a web engineering process were identified (e.g. disciplines, activities, artifacts, roles, etc.). Second, XP is modeled using SPEM [5] to obtain a better abstraction for representing the process and also to help its adaptation towards web application development. Third, adaptations of XP´s elements and the inclusion of novel elements (disciplines) were performed to tailor the method to a web engineering framework. Finally, the process elements of XWebProcess were detailed and specified using SPEM. Overall, XWebProcess can be seen as a XP-based extension tailored to web application development as shown in Figure 1.

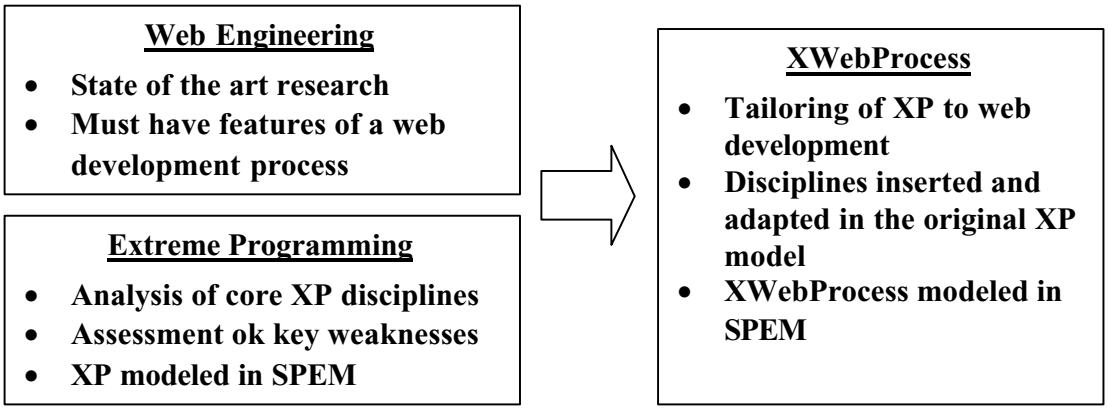

Figure 1. XWebProcess creation steps

3.1. Description of XWebProcess

In this section, XWebProcess is described in two views using the software process modeling language SPEM [5] to facilitate process construction [13]. The use of SPEM enables an abstract description of the software process core elements (artifacts, roles, activities, etc.) and the description of how they relate to each other. The SPEM notation also helps to illustrate the adaptations and tailorings performed over XP to target web application development. SPEM was chosen due to its OMG standard status for software

process modeling and due to the extensive endorsement provided by software companies such as IBM, Rational Software and Unisys.

The first view of the process model uses UML´s activity diagram with the discipline[1] stereotype defined in SPEM as shown in Figure 2. This model helps to understand how the process behaves through time (dynamic view). The disciplines highlighted are the ones adapted or inserted in the original XP modeling, shown in [3].

XWebProcess starts with an exploratory discipline encompassing experiments with technologies, architectures and system prototypes aiming at verifying viability of the project and defining initial requirements.

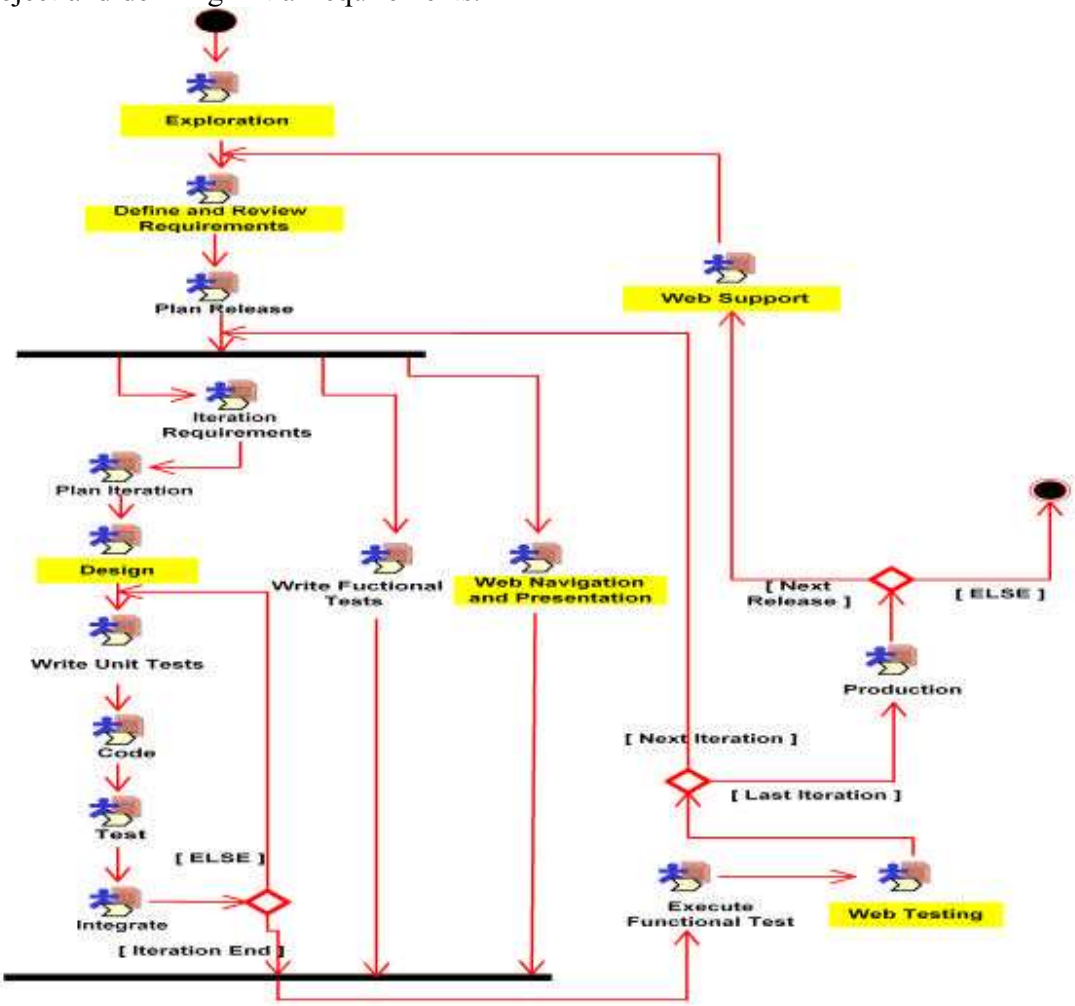

Figure 2. Dynamic modeling of XWebProcess

Afterwards, clients and programmers write story cards representing requirements of the next release. Programmers estimate the effort of implementing each story based on their past experience. The decision of what stories are implemented first is left to clients, who define priorities for the stories and select the ones with higher priority to be implemented first.

The number of stories implemented in the release depends on the speed of the team and on the difficulties estimated for each story. Therefore, programmers and clients need to agree and plan the release together.

[1] A discipline in SPEM represents a set of related process elements (artifacts, activities, roles) grouped by a common theme or objective. Examples of disciplines are: analysis and design, coding, testing, etc.

Inside a release, many iterations occur. In each iteration, stories are implemented and tested. Every iteration contains a set of disciplines, which are frequently enacted, including: plan iteration, design, writing of unit tests, coding, testing and integration. Moreover, during an iteration the requirements can change and previous estimates can be reassessed to reflect new customer's needs. Functional test cases as well as web navigation and presentation design are done in parallel with the previous set of activities in order to generate important artifacts (functional test cases and web components) used later.

When an iteration ends, it is important to verify if the functionalities implemented conform to what was specified previously. So, while functional testing is done to assure that the system does what is intended to, web testing is performed to verify if the system works appropriately when considering non-functional issues.

If the iteration corresponds to the last iteration of the release, the current version of the system enters into production. This can be done in the client's company or in another place that simulates the real production environment. After the first release of the system is delivered the web support activities start. The process finishes when all stories are implemented and delivered to the client.

It is important to mention that XWebProcess is a general web development process that does not depend on any specific technology, method or tool. It can be supported either with .NET or J2EE platforms. It also works with OOHDM [18] or OOWS [19] that are design methods for web applications. What is important to consider is if the specific method, tool or technology will have a negative impact on the agility of the process.

The highlighted disciplines in Figure 2 relate to core issues that a web development process should tackle. While some of them were adapted from the previous modeling of XP, not shown in this paper but shown in [3], others were inserted in XWebProcess. Table 1 explains the core disciplines and why they were added or modified in XP.

Table 1. Inserted and adapted disciplines

Discipline	Extension	Description
Exploration	Modified	Modified to include prototype sessions. During initial explorations is important to investigate if the system is viable or not. In web applications it is important to conduct prototype sessions with clients and business managers to help define initial requirements, scope and business goals of the system.
Define and Revise Requirements	Modified	Modified to include an architecture design activity. In web development the construction of a sound and flexible architecture is vital for the system's success, because web applications are constantly being integrated with other systems and incorporating new technologies.
Design	Modified	Modified to include the design data layer activity. It is not clear in the definition of XP where this important activity is addressed therefore we decide to include it here.
Web Navigation and Presentation	Inserted	Introduced due to the importance of navigation and presentation in web applications. Web user interfaces can be complex including graphics, sound, animations, etc. It is also common to have different navigation paths that can be followed by users.
Web Testing	Inserted	Introduced to contemplate extensive testing. In web application, testing some requirements, especially non-functional, is fundamental. It is essential to verify issues like performance, network load, number of users, etc.
Web Support	Inserted	This discipline focus is to deal with the organization of the hardware and software components that form the website. In web applications there are many distinct components (hypertext, figures, code, database, etc.) that can be distributed along the network. Therefore, it is important to have a good organization of those components in order to make corrections and updates easy.

All disciplines shown in Figure 2 were also modeled using class diagrams with SPEM stereotypes to describe how the elements relate to each other, simplifying the understanding of what artifacts are produced or consumed by each activity and what role to perform or assist in an activity.
Due to lack of space, only the highlighted disciplines are presented here. Some of these disciplines were modified from the original XP model (Figures 3, 4 and 5), with modifications also highlighted, and the others (Figures 6, 7 and 8) were inserted in the original model. For Figures 3, 4 and 5 only the highlighted parts are described. The complete description of the process is available in [3].

3.1.1. Exploration (Figure 3)

This discipline was modified to include prototype sessions. It is a responsibility of the web designer (SPEM "process role" stereotype) to execute the Web Prototyping activity (SPEM "activity" stereotype) whose outcome is the web prototype (SPEM "work product" stereotype). The designer can be guided by prototyping techniques (SPEM "guidance" stereotype) to execute the activity.

3.1.2. Requirements (Figure 4)

Modified to include the define architecture activity. In XWebProcess the architect executes this activity and the outcome is the system architecture model. In the case of web applications it is very important to design a sound architecture since the architectural model will serve as a guide for other activities like analysis, design and coding. The architect should be an experienced developer that is able to balance issues like: reuse, maintainability, flexibility and also efficiency.

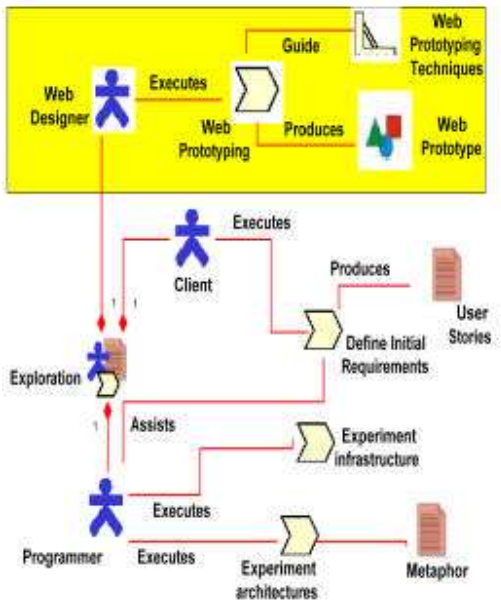

Figure 3. Exploration

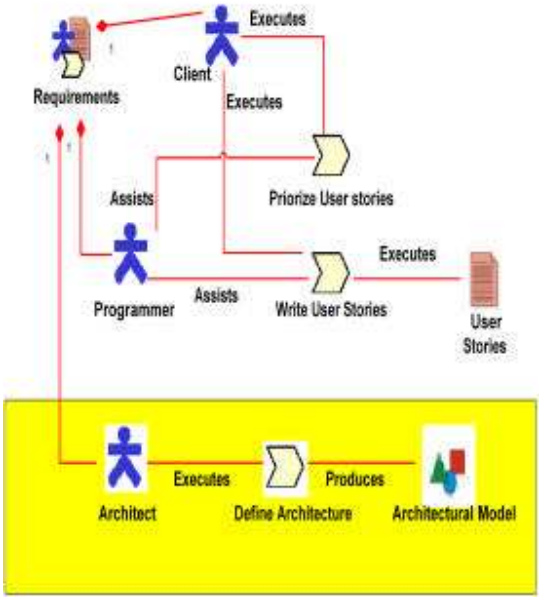

Figure 4. Requirements

3.1.3. Analysis and Design (Figure 5)

This discipline was modified to include the design data layer activity. The DBA is responsible for performing this activity which will give rise to the database and information architecture design. In the case of data-intensive web applications, this activity may also involve tasks such as data quality assessment, definition of the data mediation and refreshing strategy, data security and recovery policies.

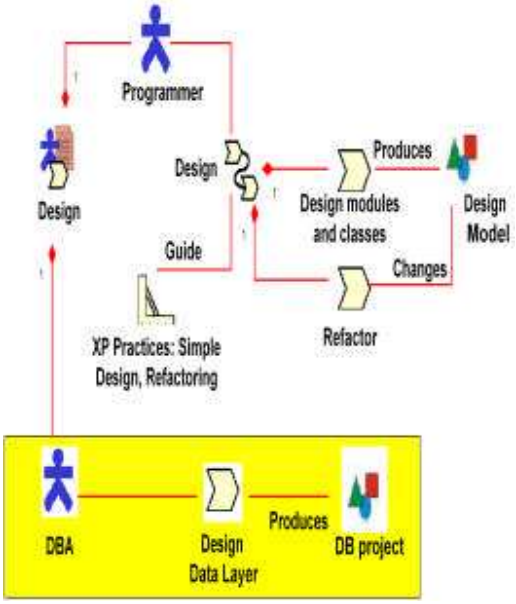

Figure 5. Analysis and Design

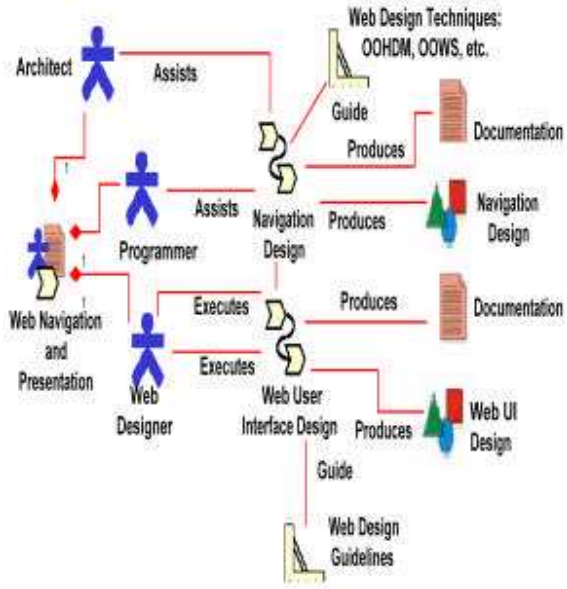

Figure 6. Web Navigation and Presentation

3.1.4. Web Navigation and Presentation (Figure 6)

Introduced due to the importance of navigation and presentation in web applications. The discipline includes three roles: programmer, architect and web designer. The web designer executes navigation design activities assisted by the programmer and architect roles. To produce the navigation design artifact, some techniques like OOHDM [18] or OOWS [19] can be used to provide guidance on how to perform these tasks in a structured way.
Other important set of activities relate to the design of the web user interface. The web designer is also responsible for this and creates the elements of the website content, such as hypertext, sound, animation, etc. Some guidelines for doing attractive web designs can be followed to improve the appearance and organization of the web content. In the case of web applications an attractive design can make the difference of whether the website will reach success alongside visitors.

3.1.5. Web Testing (Figure 7)

Introduced to contemplate extensive testing. The programmer, guided by web testing guidelines, is responsible for performing the tests, which can be automated by testing tools, and executes the web testing activity whose outcome is the test report. The support annalist assists the programmer in case any setup configuration is needed to run the test such as special files, devices and environment variables. In addition, the client can also assist the programmer towards validating functional and non-functional web test requirements.

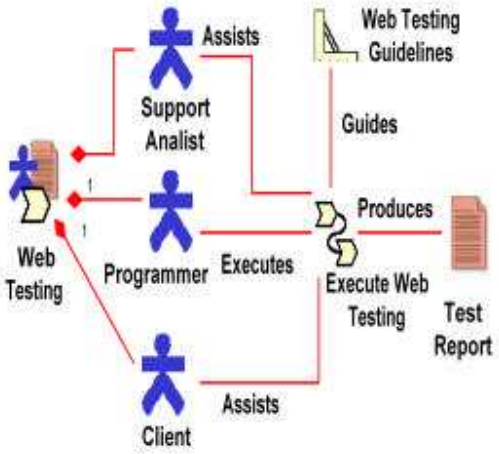

Figure 7. Web Testing

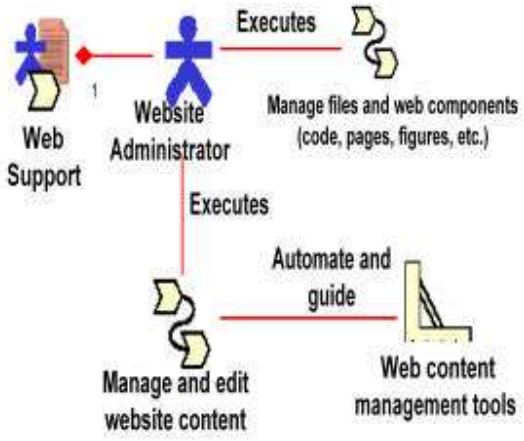

Figure 8. Web Support

3.1.6. Web Support (Figure 8)

This discipline focus is to deal with the organization of the hardware and software components that compose the website. The website administrator role is responsible for two major set of activities: content management and infrastructure management. The first one can be facilitated by a content management tool and is more important in some types of web applications such as newspaper, magazines and portals where content changes frequently. The infrastructure management activity is related to the distribution of components that form the web application (scripts, server pages, business code, figures, audio, etc.). The components should be distributed in an intelligent way so that the use of the infrastructure is maximized.
In the next section we analyze what benefits all this disciplines bring to web development. An experiment and a survey were conducted to analyze XWebProcess in terms of effort and qualitative factors.

4. Empirical Evaluation

Both processes XP and XWebProcess were evaluated in the context of an empirical study assessing the effort spent in both processes to develop the same web project. The experiment was designed to demonstrate that XWebProcess retains the agile properties of XP despite the extensions performed on XP to enhance web application development quality. The experiment was planned according to guidelines specified in [20-27]. Data about the effort spent in each discipline and artifact were collected for both processes and compared. Furthermore, a survey was executed to obtain qualitative feedback about the capabilities, usefulness and agility of both processes in helping to produce high quality web applications.

4.1. The Experimental Setting

The project upon which the study was done consisted of the implementation of a mid size web application. The same application was implemented by eight teams of five programmers each. All team members involved in the experiment were senior year students at UFPE in Brazil with at least three years of exposure to programming. The project was also made a formal coursework for the software engineering's course grade.

The project application was a web bookstore with the following core functional requirements:

[FR1] employee information access: the system shall provide functions for inserting, deleting and updating employee records. Only a logged valid employee (with a valid login and password) can have access to these functions.

[FR2] book information access: the system shall provide functions for inserting, deleting and updating book records. Only a logged valid employee (with a valid login and password) could operate these functions.

[FR3] client information access: the system shall provide functions for registering and updating clients. The client is responsible for registering and if successful, he is allowed to purchase books.

[FR4] sale support: the system shall enable customers to browse the shop and add books to a shopping basket. Books can be searched by title or ISBN, until the client decides to checkout and close the sale. The sale is confirmed by the generation of a code number that can be used later for consultation.

Some important features of the experiment were:

1.The experiment involved forty programmers divided into eight teams. Four teams implemented the system using XP, while the other four implemented using XWebProcess. The allocations of programmers to teams was randomic;

2.The experiment was conducted independently by the teams. There was no communication among them, but they had freedom to determine their own schedule to perform the tasks, as long as they took notes on the time spent in each task and quality aspects observed;

3.The technologies chosen for the implementation were: JSP and Servlets for the presentation layer; Java for the business layer; and JDBC for the data layer;

4.The level of experience of the development teams was homogeneous since they were students from the same class and year, who have similar work experience and background;

5.All students had experience with Java programming (more than two years) and database programming (more than one year);

6.No student had previous project experience with the web technology chosen (JSP and Servlets) which they learned during the experiment;

7.Automated testing using JUnit was explained before the experiment started and was used by all teams;

8.The time for the roll-out of the system was twenty days divided in two ten-day iterations.

The teams were introduced to each process by taking classes and reading their definition in [3]. An important observation is that students belonging to the XP process stream did not attend classes of XWebProcess, what could have influenced their experiment. All students played one or more roles in the process such as: programmer, tester, web designer, tracker, client and architect. The authors of this paper played the role of clients, helping in the requirements definition, and also project managers, seeing if the subjects were following the process correctly.

The main goal of the experiment was to compare XP and XWebProcess in terms of effort and quality of the outcome. We wanted to prove that the adaptation we have done by adding elements into XP to create XWebProcess had no significant impact on the process agility, what will be shown by the quantitative data collected. Moreover, we also believed that the adaptations we have done were necessary and that XWebProcess is more suitable than XP to develop web applications. This claim will be backed up empirically by the

qualitative survey executed. The definition of the experiment is summarized in the goal definition template below [25,26].

Object of Study: XP and XWebProcess.
Purpose: Evaluate the use of both processes in web application development.
Quality Focus: Effort spent.
Perspective: System developers.
Context: The experiment was performed by forty undergraduate students as subjects, divided in eight teams of five programmers. Four teams used XWebProcess and the other four used XP to develop the same web system. All teams have a previous training in XP and JUnit and the XP teams attended classes of XWebProcess only after finishing the experiment.

4.2. How experimental data was collected

During the development of the system, each team collected data using an Excel data sheet. All members of a team colleted data while working in a specific activity, for example: coding, testing, etc. However, there was only one official data sheet that was updated constantly by the tracker of each team with the information gathered by other team members. An outline of the data sheet used is shown in Figure 9.
The data sheet shows an example of how it was filled during the experiment: three team members (atfs, egml, pmb) were involved in the activity of requirements definition inside the XPDRR discipline[2]. The code for each discipline (e.g. XPDRR) was defined based on the names shown in Figure 2. The activity created the story cards in thirty minutes and the time spent in the whole activity is shown by the delta (one hour and twenty minutes).
All teams involved in the experiment collected data using the same data sheet. The tracker of each team was responsible for updating the sheet frequently with the data collected by other members of his team. In the end of the experiment all teams delivered one sheet containing data about the work done during the whole experiment (twenty days).

Date	Start	Stop	Interruption	Delta	Iteration	Discipline	Activity	People (login)	Artifacts	Obs.
01/12/2003	13:20	14:50	00:10	01:20	1	XPDRR	Define Iteration Requirements	atfs,egml pmb	Story Cards (30 Min)	-

Figure 9. Data sheet

The main goal of the data sheet was to collect data about the effort spent in each process enabling the comparison of XWebProcess' effort vis-à-vis the effort undertaken by the teams developing using XP. Therefore, basically with this data we can answer questions such as:

- How much time is spent in each discipline of XP and XWebProcess?
- How much additional effort the adaptations and extensions done in XWebProcess represent?
- What are the advantages and disadvantages of using XP or XWebProcess?

Besides collecting this quantitative data, we collected qualitative data based on a survey. All the subjects involved in the experiment were asked to answer the survey, which helped

[2] XPDRR is the Define and Revise Requirements discipline that relates to requirements definition in XWebProcess. This discipline is described in detail in [3].

in obtaining their perspectives on the advantages of both processes. It is important to mention that the teams that used XP did not attend classes on XWebProcess before the experiment, but they attended after the experiment had finished to answer the survey.
The survey encompasses questions about the quality of both processes: how easy is the task of learning the processes; how well defined and structured both processes are; how the model specifications available for both processes helped in the task of understanding and following the process; and which process was perceived as the most suitable for web development. The goal of the survey was to gauge if the modeling done provided benefits to the developer in the learning and application stages and also to assess if the features available in XWebProcess enhanced the final outcome of a web application project. Details about the survey can be seen in [3].

4.3. Data Analysis and interpretation

Table 2 shows the results (effort in person-hour) obtained by the eight teams involved in the experiment. Data was collected as each team enacted the activities defined in the process being followed (XP or XWebProcess) and filled in the time spent as well as the number of people in the spreadsheet shown in Figure 9. When the project was finished the data collected was gathered resulting in the total effort spent by each team.

Table 2. Data Results

Effort Spent (Person-Hour)							
XP				XWebProcess			
T1	T2	T3	T4	T5	T6	T7	T8
121.03	106.83	98.17	90.67	129.25	125.05	110.67	90.68

The t-test was used to perform the statistical analysis as defined in [25]. The analytical goal relates to verifying the null hypothesis (NH) stated in equation (1). The NH hypothesis assumes that the effort spent to the develop a web system using both processes is the same. This can be verified by calculating the t statistic as described in [25]. The Alternative hypothesis (AH) is assumed to be true if NH is refuted and in this case indicates that the efforts are not the same.

(1) NH: $_x = _y$, i.e. the expected mean values of the effort spent are the same.
AH: $_x \quad _y$: Reject NH if $|t| > t_{/2,f}$ where f is the degree of freedom and is the significance level. $t_{/2,f}$ is obtained from a statistical table.

The x values represent the effort spent in each of the four XP teams (T1,T2, T3, T4) and the y values represent the effort spent in XWebProcess teams (T5, T6, T7, T8). After calculating the t value we want to confirm that the alternative hypothesis does not hold. This is summarized in equation (1).
We performed the analysis using one of the Excel´s data analysis tools for the t-test. The results obtained are summarized in Table 3.

Table 3. Data Analysis

General Results		
	XP Teams	XWebProcess Teams
Mean	104.175	113.9125
Variance	169.8617	303.1819
Observations	4	4
T-Test Results		
t stat	-0.89542	
t Critical two-tail ($t_{\alpha/2,f}$)	2.446914	
Conclusion	$\|t\| < t_{\alpha/2,f,}$ therefore NH is true	

The results show that t = -0.8954. The value $t_{/2,f}$ can be seen in statistical tables [25] and in this case represents the critical value for the two-tailed t-test ($t_{/2,f}$ = 2.446 for = 5% and f = 6). Therefore $|t| < t_{/2,f}$ and the null hypothesis (NH) is true. This result means that the effort spent in both processes is the same proving what we assumed before.
It is important to observe that in absolute values the effort spent by XWebProcess teams were higher than in XP teams. This can be seen by the mean values in Figure 5 (104.1 in XP and 113.9 in XWebProcess) and also by the data shown in Table 2. The higher number of process elements such as disciplines, activities and artifacts in XWebProcess demands more effort, what explains these results. However, the statistical analysis performed by the t-test shows that this effort is not significant and both processes can be considered equal in terms of effort.

4.3.1. Qualitative analysis

The key qualitative results surfacing from the survey are:

- Both processes presented advantages relating to less formalism and less documentation favoring more communication and interaction among team members with a positive result on speed of execution;
- All subjects involved in the experiment rated XWebProcess more suitable for web development than XP. In particular, web user interface and navigation design disciplines promoted the development of a better website in terms of organization and appearance. Moreover the use of prototyping sessions benefited requirements gathering and improved communication with clients.
- All subjects reported that the web testing discipline of XWebProcess had a positive impact on enforcing time, load and security constraints;
- All subjects using XWebProcess reported that despite having to cope with more activities and artifacts than XP there was a pay off in the quality of the outcome;
- 95% of the subjects using XP agreed that the process needs to be adapted and tailored to web development as done in XWebProcess;
- All subjects reported that the SPEM model helped in understanding the process structure.

All participants of the experiment reported that amongst the key successful practices of XWebProcess was the practice of anticipating and rapidly correcting requirements misalignments through frequent cycles of communication between business stakeholders and the development team and also by using prototyping techniques. This feature generally supported by most agile methods has been lately mentioned as a key cornerstone of evolutionary software project management in successful software engineering [33].

5. Discussion and Related Work

Web application engineering is still short on contributions addressing the process of web application construction. The existing application development literature emphasizes web development technologies such as middleware, script languages, server processing (JSP and ASP) and development platforms (J2EE, .NET).
The AWE process [16] is an example of a web engineering process developed after eliciting key web application development success factors to be contemplated within a software process. The success factors were identified through a survey conducted with several web application solution providers in the United Kingdom. AWE is not an agile method per se, limiting its focus on what should be treated in some disciplines of the

software development cycle like requirements, design, implementation and testing. Moreover, it describes the roles (web designer, programmer, etc.) necessary to execute the activities in the process. AWE is described textually and there is not a clear definition of neither its structure, i.e. how its disciplines relate, nor its dynamics, i.e. how the process flows through the project lifetime. In addition, this process is still being subject to a case study and has not been assessed experimentally yet.

Henderson-Sellers and Lowe [29] have extended the OPEN process to give support for web development. New activities, tasks, techniques and roles were created and adapted in the original process. In this case, there is a clear definition of the process structure but the dynamics of the process is still difficult to understand. As the AWE process, OPEN-Web is textually described and is overloaded in terms of disciplines elements. OPEN-Web has not been assessed experimentally either.

XWebProcess is described using SPEM, which is a standard OMG language for describing software processes. The modeling in SPEM facilitates the understanding of both structure and dynamics of the process providing UML models that are easy to read (evidence shown in the qualitative survey previously described). Furthermore, given the compact description provided, it is highly suitable for further adaptation and improvements. Another advantage of XWebProcess is that it has an initial experimental assessment available providing quantitative and qualitative results on the process effectiveness. A comparison of the three processes is presented in Table 4.

Table 4. Comparative table for existing web development processes

Dimension	AWE Process	OPEN-Web Process	XWebProcess
Process Description	Textual	Textual	Textual and modeled with SPEM
Process Learning Curve	Difficult to understand structure and dynamics	Easy to understand structure and difficult to understand dynamics	Easy to understand structure and dynamics
Process Nature	Lightweight (agile)	Heavyweight	Lightweight (agile)
Assessment	Case study still being conducted	No assessment yet	Experimental study conducted

6. Summary and Future work

Delivering high quality web applications complying with severe time constraints is still an elusive goal for a software process. The agile way of building software is a viable alternative to traditional methods that has already shown encouraging results over the last years [17]. Despite the positive results so far, there is still substantial room for improvement by finding the optimal balance between agile principles and process disciplines that will ensure the quality of the delivery.

XWebProcess is an agile method for building web applications that seeks to retain the agile properties of Extreme Programming while maintaining development disciplines that have a positive impact in the quality of the delivery. The set of process disciplines supported have been chosen after assessing the key characteristics of a web application project that demand special consideration within the process structure.

Despite the extensions made to Extreme Programming, the agility of XWebProcess when compared to XP was not altered while the experimental results also provided evidence that

web development dimensions such as requirements gathering, user navigation design, and software testing were improved by XWebProcess.

Although the experiments and surveys conducted gave important feedback on the adequacy of both processes (XP, XWebProcess) in supporting web application development, there are still challenging issues that need to be further explored in the quest for finding the optimal balance between speed and quality. Some of the noteworthy issues are:

- How speed and quality indicators unfold on mid and large scale web application projects (e.g., more than 50000 LOCs);
- What other disciplines can be added to improve web application quality without sacrificing agility;
- How XWebProcess performs when compared to other processes that are also tailored to web application development (e.g., AWE and OPEN).

Answers to the issues above will provide important empirical results and best practices empowering web application developers with proven methods for accelerating web application development without sacrificing software quality.

References

1. Beck, K. Extreme Programming Explained. Addison-Wesley, 1999.
2. Jeffries, R., Anderson, A., and Hendrickson, C. Extreme Programming Installed. Addison-Wesley, 2001.
3. Sampaio, A. XWebProcess: Um processo ágil para o desenvolvimento de aplicações Web. MSc Thesis, March 2004, Universidade Federal de Pernambuco.
4. Pressman, R.S. What a Tangled Web We Have. IEEE Software, January – February 2000, pp 18-21.
5. Object Management Group. Software Process Engineering Metamodel Specification. Version 1.0, November 2002.
6. Agile Manifesto Web Site. Available at: http://www.agilemanifesto.org.
7. Dynamic System Development Method (DSDM) Web Site. Available at: http://www.dsdm.org.
8. Crystal Web Site. Available at: http://www.crystalmethodologies.org.
9. Cockburn, A. Agile Software Development. The Agile Software Development Series, Addison Wesley, 2001.
10. Ambler, S. Different Projects Require Different Strategies. Available at: http://www.agiledata.org, 2002.
11. Lindval M., Basili, V., Boehm, B., et al. Empirical Findings in Agile Methods. Lecture Notes in computer Science, Volume 2418, pp 0197-0207, September 20th 2002.
12. Fowler, M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.
13. Heinecke, H., Noack, C., and Schweizer, D. Constructing Agile Software Processes. In Proceedings of Extreme Programming Conference (XP2002), Alghero, Sardinia, Italy 2002.
14. Jacobson, I., Booch, G., and Rumbaugh, J. The Unified Software Development Process. Addison-Wesley, 1999.
15. Object-oriented Process, Environment and Notation (OPEN) Web site. Available at: http://www.open.org.au/
16. McDonald, A., and Welland, R. Agile Web Engineering (AWE) Process. Department of Computing Science Technical Report TR-2001-98, University of Glasgow, 2001.

17. Rumpe, B, and Schröder, A. Quantitative Survey on Extreme Programming Projects. In Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002, May 26-30, Alghero, Italy, pp. 95-100, 2002.
18. Schwabe, D. and Rossi, G. The Object-Oriented Hypermedia Design Model. In Communications of the ACM, volume 38(8), pages 45-46, 1995.
19. Pastor, O., Fons, J., Abrahão, S., and Rámon, S. Object Oriented Conceptual Models for WEB Applications. In 4th Iberoamerican Workshop on Requirements Engineering and Software Environments (lIDEAS´2001), San Jose, Costa Rica, 2001.
20. Pfleeger, S. L. Design and Analysis in Software Engineering – Part1: The Language of Case Studies and Formal Experiments. Software Engineering Notes, vol 19, no 1, October 1994, pages 16-20.
21. Pfleeger, S. L. Experimental Design and Analysis in Software Engineering – Part2: How to Set Up an Experiment. Software Engineering Notes, vol 20, no 1, January 1995, pages 22-26.
22. Pfleeger, S. L. Experimental Design and Analysis in Software Engineering – Part3: Types of Experimental Design. Software Engineering Notes, vol 20, no 2, April 1995, pages 14-16.
23. Pfleeger, S. L. Experimental Design and Analysis in Software Engineering – Part4: Choosing an Experimental Design. Software Engineering Notes, vol 20, no 3, July 1995, pages 13-15.
24. Pfleeger, S. L. Experimental Design and Analysis in Software Engineering – Part5: Analyzing the Data. Software Engineering Notes, vol 20, no 5, December 1995, pages 14-17.
25. Wohlin, C., et al. Experimentation in Software Engineering: An Introduction. Kluwer Academic Puslishers, 2000.
26. Basili, V., et al. Experimentation in Software Engineering. IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, July 1986.
27. Seaman, C. Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, Vol. 25, No. 4, July/August 1999.
28. Sommerville, I., Software Engineering, 6th Edition, Addison Wesley, 2003.
29. Henderson-Sellers, B., Lowe, D., and Haire, B., OPEN Process Support for Web Development, Annals of software Engineering, vol. 13, pp 163-202, 2002.
30. Lowe, D., and Eklund, J., Client Needs and the Design Process in Web Projects, Journal of Web Engineering, vol. 1, p. 23-36, 2002, Rinton Press.
31. Wallace, D., Raggett, I., and Aufgang, J., Extreme Programming for Web Projects (XP Series), Addison-Wesley, 2002.
32. Sampaio, A., Vasconcelos, A., and Sampaio, Pedro R. Falcone, XWebProcess: Agile Software Development for Web Applications, Accepted Poster Presentation to appear in International Conference on Web Engineering – ICWE04, Munich Germany, 2004.
33. The Challenges of Complex IT Projects, Report presented by The Royal Academy of Engineering and The British Computer Society in April 2004. Available at http://www1.bcs.org.uk/portal/showSection.asp?contentid=7560&link=/Docs.