Assessing Agile Methods: An Empirical Study

Américo Sampaio[1], Alexandre Vasconcelos[2] and Pedro R. Falcone Sampaio[3]

[1]Computing Department, Lancaster University – UK
a.sampaio@comp.lancs.ac.uk

[2] Centro de Informática, Universidade Federal de Pernambuco
Recife, Pernambuco – Brazil
amlv@cin.ufpe.br

[3] School of Informatics
University of Manchester – UK
p.sampaio@co.umist.ac.uk

Abstract. Agile software processes emerged to address the issue of building software on time and within the planned budget. To adopt an agile process, it is imperative to analyze and evaluate its effectiveness in supporting high quality software development while complying with stringent time constraints. In this paper we describe an agile method for Web-based application development (XWebProcess) and an experiment conducted with a group of forty senior undergraduate students to assess the quality/speed effectiveness of the proposed method vis-à-vis the effectiveness of Extreme Programming (XP). The results have shown that the process proposed is equally agile when compared to XP, moreover, surveys conducted as part of the experiment pointed out that XWebProcess is more suitable to Web development in dimensions such as requirements gathering, user interface and navigation design, and software testing, therefore leading to better quality software.

Keywords. Software Process, Agile Processes, XP, XWebProcess, Experimentation.

1. Introduction

Delivering high quality web applications complying with severe time constraints is a highly challenging dilemma within the software engineering community [33]. In many software projects, 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, increasing the risk of potential negative side effects on usability, maintainability 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.

To address the challenges of speeding up development cycle times while retaining superior software quality standards, several methods, techniques and tools have been developed by the software engineering community. Examples of key contributions that have reached widespread uptake both in industry and academia are: software components [34], model-driven application development [35], and software product lines [36]. The popularity of these approaches stems principally from their ability to accelerate software development following a systematic and engineering-based approach.

More recently, the need for accelerating delivery cycle times while coping with fast changing requirements has also raised a new stream of research towards lightweight or agile software processes. Agile processes have reached a considerable interest both in industry and academia due to a proven track record in delivering complex software on time and within the prescribed budget [1,2,17]. Despite the encouraging results reported by the agile methods community, there is still a need for empirically understanding the quality/speed trade-offs in real life projects when following agile principles.

In this paper we describe an agile method for web-based application development (XWebProcess) and an experiment conducted with a group of forty senior undergraduate students to assess the quality/speed effectiveness of the proposed method. The results have shown that the process proposed is equally agile when compared to Extreme Programming (XP), moreover, surveys conducted as part of the experiment pointed out that XWebProcess is more suitable to Web development in dimensions such as requirements gathering, navigation design, and software testing, therefore leading to better quality software.

The experiments described in this paper adopt the time effort measure (man hours) as a reasonable proxy for assessing the agility or speed of a software process. The results reported in this paper also have an important impact considering the limited availability of literature assessing agile processes empirically. Within the literature, a substantial number of papers claim that their proposed methods facilitate the construction of software on time, however, without providing empirical evidence. This paper will not only provide evidence about agility, but will also present qualitative data assessing the quality of the software outcome produced by the processes under comparison.

XWebProcess was developed as part of our research in the field of web engineering [3,37,38]. Web engineering is a challenging domain given the severe time constraints and strategic nature of the majority of projects [4,16]. Although the main focus of this paper is to analyze and compare two agile processes (XP and XWebProcess) a brief description of each process is also presented helping to clarify the key differences of the two processes under comparison.

The remainder of this paper is organized as follows. Section 2 introduces basic concepts for agile processes, Extreme Programming, and software process modeling. Section 3 presents an overview of XWebProcess describing how the process was created and modeled. Section 4 describes the experimental background, planning, execution and analytical interpretation comparing XP and XWebProcess empirically. Section 5 illustrates the survey conducted to analyze qualitative dimensions regarding both processes. Finally, section 6 concludes the paper by summarizing and discussing the key lessons learned.

2. Background

Section 2.1 gives an overview about agile processes and XP. Section 2.2 explains software process modeling and the Software Process Engineering Metamodel (SPEM) [5] and why it was used in XWebProcess.

2.1. Agile Processes and Extreme Programming

Over the last decade, the software development community has been questioning about the efficiency of traditional software processes and models. In particular, late research in the field of software processes point out that mainstream processes such as Rational Unified Process (RUP) [14] dedicate too much emphasis to documenting and following rigid plans, thus, slowing down software development [1,2,6,9,10,11].

Agile processes like Extreme Programming (XP) [1,2], DSDM [7] and Crystal [8] emerged to address the issue of building software in a fast but at the same time structured pace. The lifecycle of agile processes is aimed at minimizing the number of activities performed and the quantity of artifacts produced in order to deliver working software to clients quickly. However, this must be done following a sound engineering approach avoiding ad-hoc short cuts in the development process.

The main goal of agile processes is to provide a roadmap of activities and practices that guide the project stakeholders to execute their tasks effectively and efficiently, focusing development effort on artifacts that are most valuable for clients and for improving the quality of the project deliverables. Another important issue is that agile processes favor using simple solutions to any problem. For example, if you can produce a model that represents the main design issues of a system by sketching a whiteboard and taking a digital picture during a thirty-minute brainstorming session, it is not necessary to spend hours to generate a detailed design model in a case-tool. The main thinking underpinning this approach is to produce a design model that provides useful information for stakeholders regardless of the technique and level of detail used.

Extreme Programming (XP) is one of the most important and probably the most popular agile process available today. XP has twelve practices that form the core of the process and guide the work of development teams. 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 the key practices is described below. For a comprehensive description of the practices see [1,2].

The Planning game: in XP there are two kinds of planning. The release planning, done in the beginning of the development cycle, defines the major features addressed by the next release. The most important features are selected based on technical estimates and business priorities. The releases in XP address a small set of features (small releases) and have a short time span (generally one to three months).

The other level of planning is iteration planning. In XP each release is divided in iterations (one to three weeks), where story cards (simple use cases) are implemented. The effort estimated to implement each story is measured by programmers with the client's assistance. In XP, clients work together with programmers and both are responsible for setting priorities for the stories that are going to be implemented (*on-site customer*). The estimates rely on past experience and with continued practice, teams tend to improve estimates.

Pair Programming: All code is produced by a pair of programmers sitting together side-by-side. While one programmer concentrates on the design of algorithms and coding, the other suggests improvements and reasons over strategies to prevent possible errors. Pair compositions are changed with frequency, enabling each team member to play different roles and work in different stories. The rationale for pair programming is to allow every team member to have an overall view of the system avoiding possible problems in case

someone leaves the team. Programmers write test code prior to writing business code and after the code is ready, tests are executed (automated testing). If all tests pass (100% correct) code is integrated with other system functionalities. The following activities are performed at an iteration cycle: write unit tests, write code, refactor code, run tests and integrate code (continuous integration).

Refactoring: Defines a series of improvements, suggested in [12], to be carried out over the code without changing its functionality. Refactoring is often performed during and after code writing when programmers identify more effective programming techniques. Automated testing also helps to maintain the quality of code due to the need for checking all tests after new functionality is produced. As a result, refactoring prevents code from being badly structured, while testing prevents the introduction of errors.

The *Collective Code Ownership* states that functionality can be altered or refactored anytime by any member of the team. Having a *Coding Standard* that everybody follows facilitates the task of working with different functionalities frequently and simplifies maintenance.

The practices described above provide an overall idea of how a project following XP is conducted. Therefore a XP project generally starts with a planning phase. The release plan gives an overall view that is divided into smaller plans (iteration plan). For each iteration cycle, the following activities are repeated: write test code, write code, refactor, execute tests and integrate. To perform these activities XP relies on the following main stakeholders:

- *Programmer:* Is the central role in XP. The programmer writes the unit tests, the code and helps clients execute tests and write user stories.
- *Client:* This role is also responsible for writing the user stories and executing the tests. Clients have the additional responsibilities of prioritizing requirements, pointing out stories that should be implemented in early releases.
- *Project Manager:* This role is responsible for controlling the project resources and also for tracking the progress of the activities.

The team composition and roles in XP projects emphasize the need for strong interaction with client team members, shared responsibilities and collective ownership for the overall success of the project.

2.2. Software Process Modeling and SPEM

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 specified role in the process, e.g., developer, manager, and customer, and can use tools and models to automate and facilitate tasks. As the process execution progresses, artifacts such as source code, documents and models are produced and updated, and can serve as input to the construction of other artifacts.

Software process modeling languages define abstractions to represent process elements like activities, roles and artifacts. Modeling languages have a key impact in providing representations of different views of a software process. For example, a model can represent the inter-relationships between the elements of a process showing its static structure (i.e., static view) or representing how the process flows throughout time (i.e., dynamic view).

The Software Process Engineering Metamodel (SPEM) [5] is an example of a process modeling language. SPEM is an official OMG standard developed to provide a common framework for describing software processes using UML notations and new stereotypes defined by its specification. SPEM is a suitable candidate for software process modeling due to its key characteristics:

- *OMG open standard status*: a process model defined in SPEM will have both syntax and semantics governed by an open international standard, facilitating understanding across practitioners, tool interchange and vendor independence;
- *Industry acceptance*: Mainstream software development companies like IBM, Rational and Unisys have contributed to SPEM and are committed to supporting the standard;
- *Synergies with UML*: SPEM employs the Unified Modeling Language notation and benefits from the body of tools and techniques developed by the UML community.

OMG also provides a four-layered modeling reference architecture, which SPEM complies with, as shown in Figure 1.

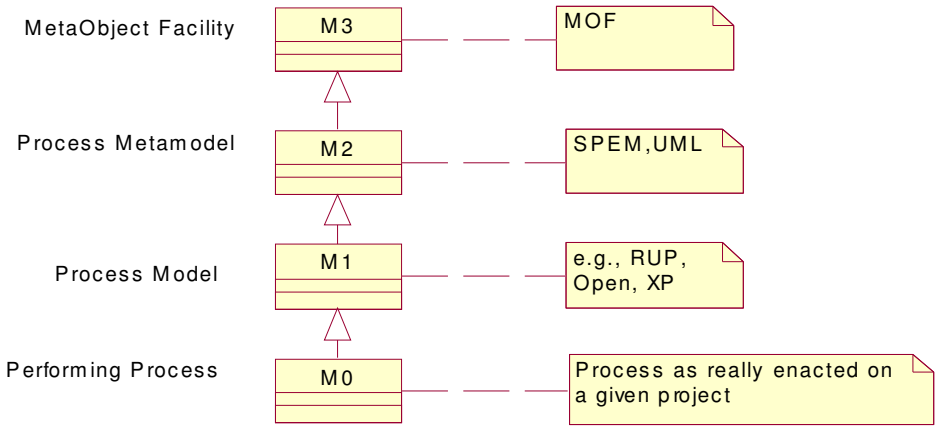


Figure 1 - Levels of modeling [5]

In Figure 1, the instantiation of the real world software process execution is at level M0. The process framework, enacted at level M0, is defined at level M1, e.g. RUP [14], OPEN [15] or XP. Level M2 defines the meta-modeling language (SPEM) that can be used to model processes at level M1. Level M3 defines a meta-model for SPEM based on MOF[1]. SPEM is a metamodeling language that extends UML with new UML stereotypes for software process modeling. The description of SPEM presented in [5] is organized in two main packages. The first package (SPEM_Foundation) is an extension of the UML 1.4 specification and contains the basic elements in which the SPEM specification is built on. The second package (SPEM_Extensions) adds the necessary abstractions and semantics required to model software processes. Both packages are shown in Figure 2.

[1] The Meta-Object Facility (MOF) is an OMG standard that defines how to represent metadata as CORBA objects. SPEM uses a subset of UML to represent its elements as a MOF meta-model.

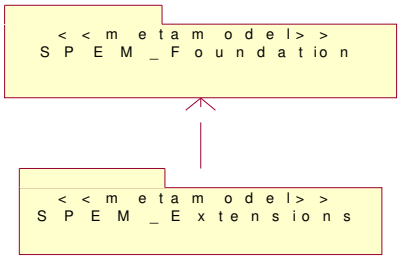


Figure 2 - SPEM Packages [5]

The packages described in Figure 2 are refined in other packages that contain all SPEM stereotypes. Figure 3 provides an illustration of all SPEM packages. For each package of Figure 3 there is a complete description of its elements, notation, relationships and semantics detailed in [5].

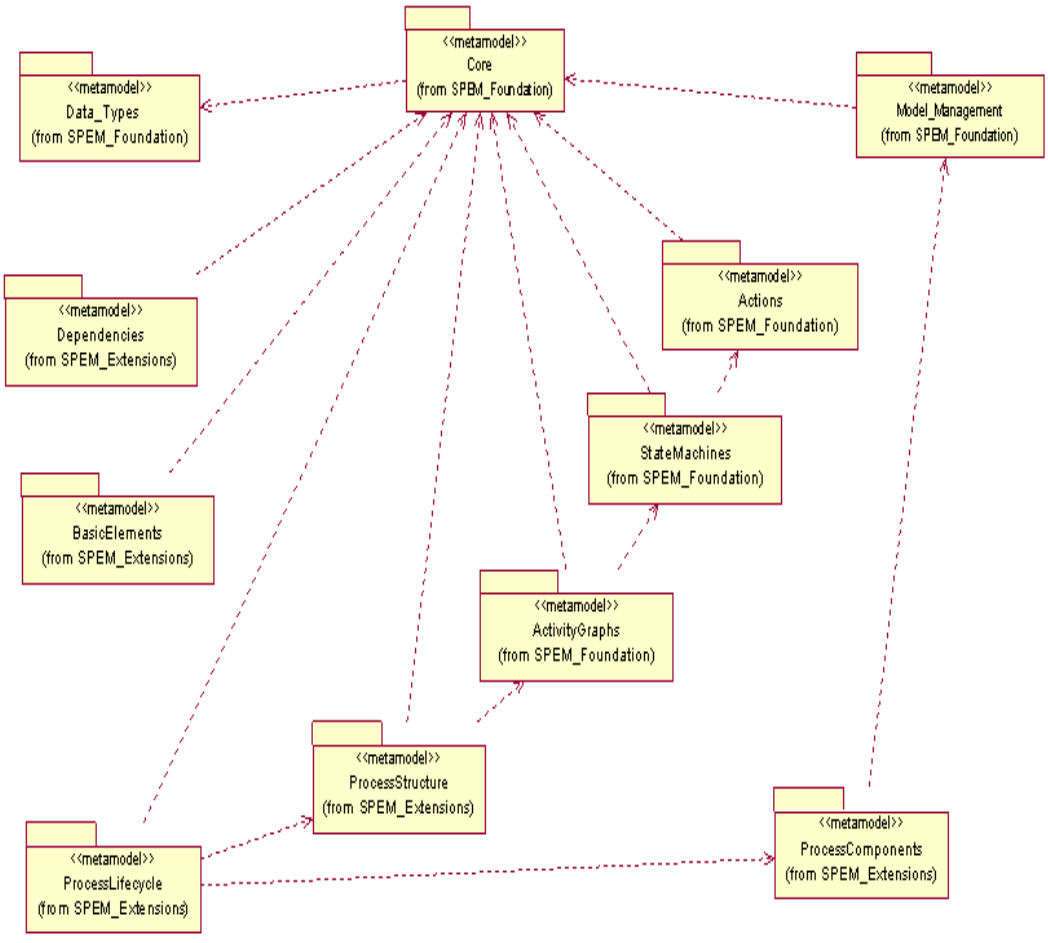
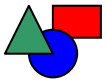


Figure 3 – Detailed SPEM Packages [5]

Elements contained in the packages shown in Figure 3 are used as modeling foundations in the development of process models. These foundation elements underpin a process model specification from two different perspectives: dynamic and static.

Table 1 provides a background on SPEM package elements used in the modeling of XWebProcess (described in the next section). The *stereotype* column relates to the SPEM element defined in one of the packages of Figure 3. The *description* column provides a brief description of the element, and the *notation* column illustrates the corresponding graphical notation.

Table 1 – SPEM stereotypes

Stereotype	Description	Notation
WorkProduct	Represents information artifacts produced or used as inputs during process activities. Typical examples of work products are: plans, source code, design models and documents.	
WorkDefinition	Defines a macro activity to accomplish a major landmark in the project. During a work definition, one or many process activities are performed employing roles responsible for the execution of each activity. A typical example of a work definition is the specification of the activities involved in creating the requirements document.	
Activity	Defines an atomic action or procedure that some role executes such as write high-level use cases.	
ProcessRole	Describes roles or responsibilities that a person can have in the process such as developer, tester, quality engineer, configuration manager and project manager.	
Guidance	Provides assistance (e.g., detailed information or best practices) to other process elements such as roles and activities. Examples of such elements are techniques, guidelines, templates and tools.	
Discipline	Defines a cohesive set of process elements such as roles, activity and work products grouped by a common goal. This definition is used by some processes like RUP and as typical examples of disciplines we can point out: planning, domain analysis, requirements, design and code.	

In the context of this work, SPEM elements and constructs were also used to model the Extreme Programming software process, facilitating the task of creating agile extensions tailored to support web application development. Agile processes often lack a model-based description of the core software process elements such as artifacts, roles, and activities, therefore limiting the understanding of how process elements relate to each other [13]. The development of a SPEM model for an agile method such as XP not only accelerated the learning of XP but it also enabled the identification of strengths and weaknesses in XP that were leveraged in the development of XWebProcess.

3. XWebProcess

Section 3.1 presents an overview of web development and XWebProcess. Section 3.2 describes the structure and dynamics of the process.

3.1. Overview

Web Applications are software systems based on standards and technologies endorsed by the World Wide Web consortium (W3C). Web applications encompass the development of web accessible user interfaces that interact with application servers deploying the core software components for the business process being enacted. Web Engineering is the application of sound software engineering approaches tailored to the web application domain [29]. Some core elements of the web application domain that have to be considered in a web engineering process are [4,16,29,30]:

- *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 specialized 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 web engineering factors 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, type 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 was 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 4.

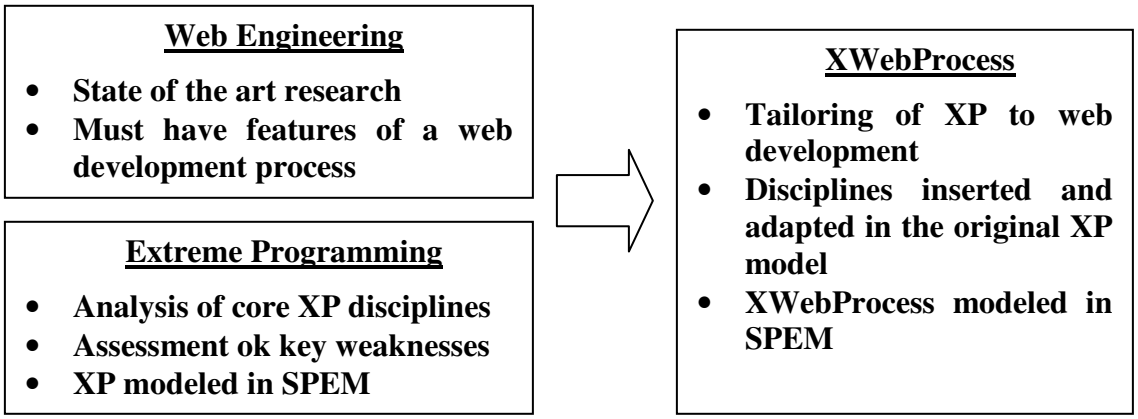


Figure 4 – XWebProcess creation steps

3.2. 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 illustrates the adaptations and tailorings performed over XP to target web application development.

The first view of the XWebProcess model uses UML´s activity diagram with the discipline stereotype defined in SPEM as shown in Figure 5. 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 [3,32]. The original XP model can be obtained by removing a set of highlighted disciplines from Figure 5 (only the inserted ones).

XWebProcess starts with an exploratory discipline encompassing experiments with technologies, architectures and system prototypes with the goal of verifying technological feasibility of the project and collecting/refining initial requirements. This discipline enables developers to become acquainted with the core elements of the project's technology stack and also to perform prototypes attached to critical aspects of the project undertaken. This discipline also helps to proactively identify and address skill gaps that the development team may encounter towards completing the project.

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 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. Inside a release, iteration cycles are performed. Within iterations, stories are implemented and tested. Every iteration cycle encompasses a set of disciplines, which are frequently enacted, including: plan iteration, design, writing of unit tests, coding, testing and integration.

During iterations, 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.

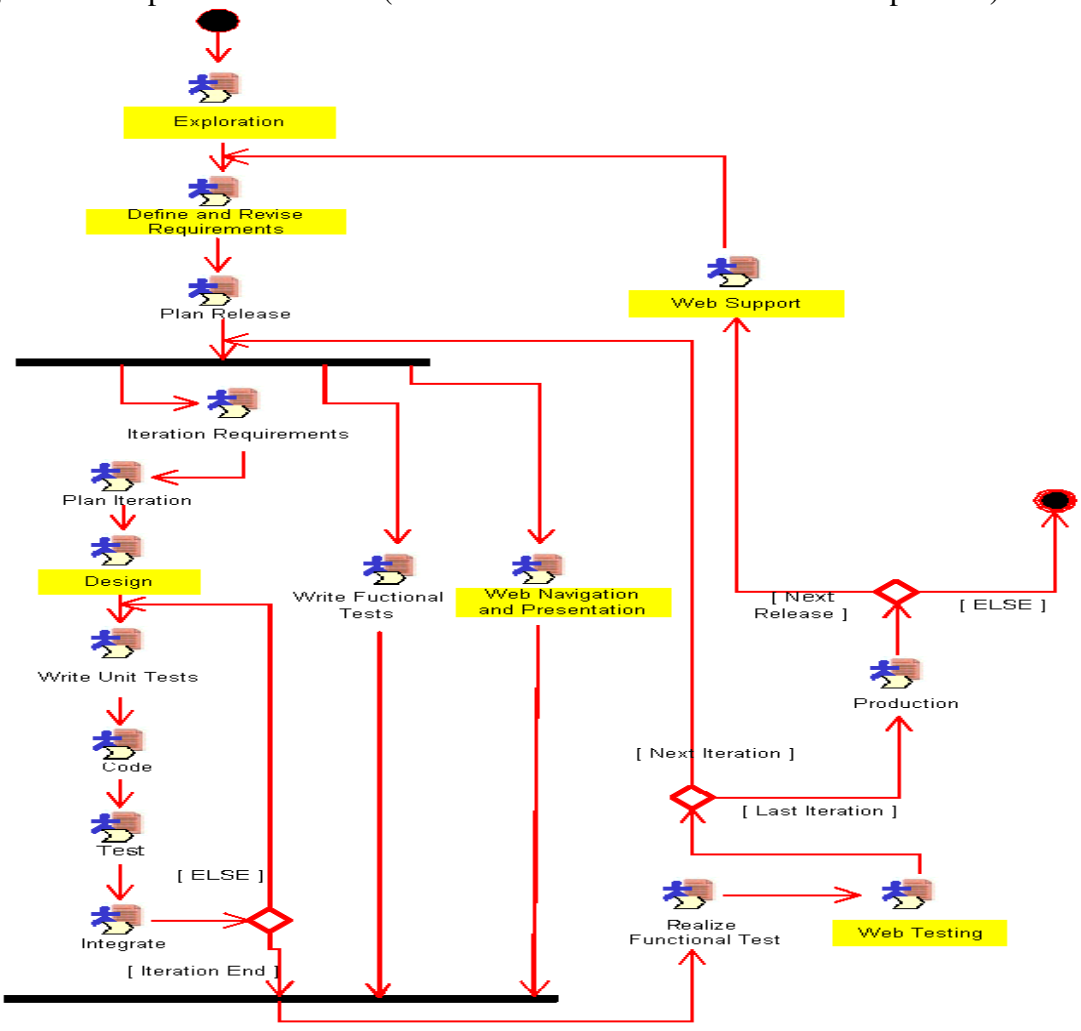


Figure 5 - Dynamic modeling of XWebProcess

At the end of an iteration cycle, it is important to verify if the functionalities implemented conform to what was specified previously. So, while functional testing is done to ensure 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 target 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 or tool. It can be supported either with .NET or J2EE platforms. The orthogonal use of web development methods such as OOHDM [18] or OOWS [19] in conjunction with XWebProcess is also possible. In this case, it is necessary to analyze the adequacy of the method with regard to its incorporation in XWebProcess and to evaluate the impact on development speed.

The highlighted disciplines in Figure 5 relate to core issues that a web development process should tackle. While some of the disciplines were adapted from the previous modeling of XP, not shown in this paper but shown in [3], others were inserted in XWebProcess. Table 2 explains the core disciplines and why they were added or modified in XP.

Table 2. Inserted and adapted disciplines

Discipline	Extension	Description
Exploration	Modified	Modified to include prototype sessions. During initial explorations is necessary 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 Review 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, since 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 it was made explicit in XWebProcess.
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 engineering, testing 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 the components in order to simplify corrections and updates.

All disciplines shown in Figure 5 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 6, 7 and 8), with modifications also highlighted, and the others (Figures 9, 10 and 11) were inserted in the original model. For Figures 6, 7 and 8 only the highlighted parts are described. The complete description of the process is available in [3].

Exploration (Figure 6)

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.

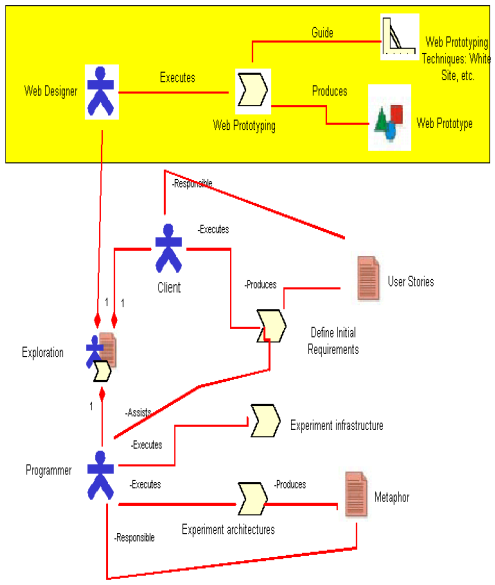


Figure 6. Exploration

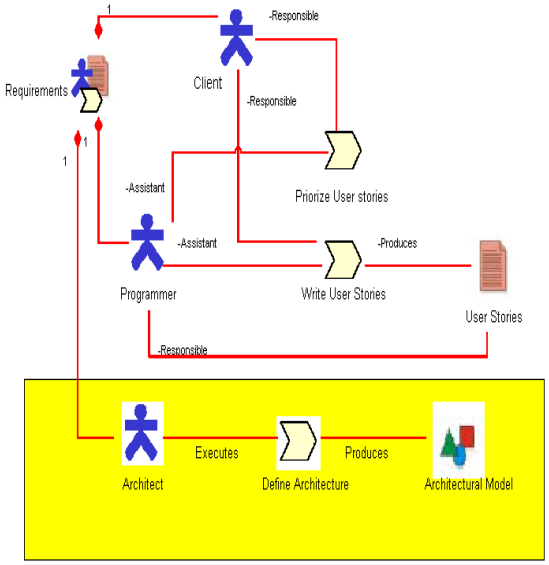


Figure 7. Requirements

Requirements (Figure 7)

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.

Analysis and Design (Figure 8)

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.

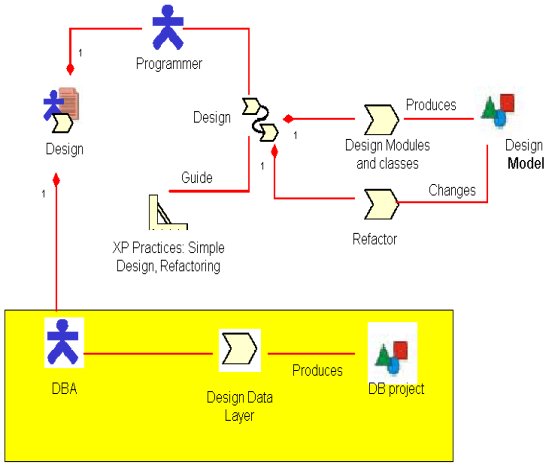


Figure 8. Analysis and Design

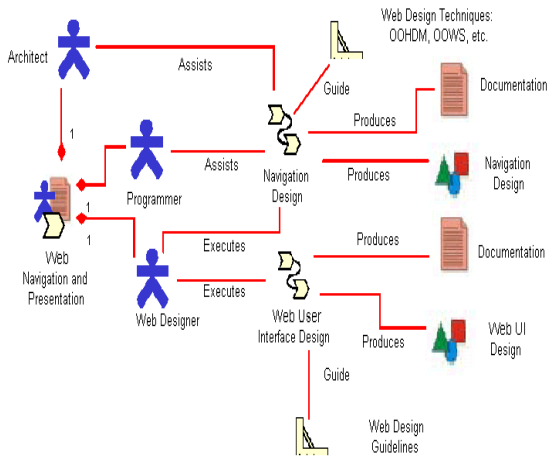


Figure 9. Web Navigation and Presentation

Web Navigation and Presentation (Figure 9)

This discipline was explicitly incorporated due to the importance of navigation and presentation in web engineering. 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 executing this task, and also responsible for creating 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.

Web Testing (Figure 10)

This discipline was incorporated to emphasize the need for 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 analyst 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.

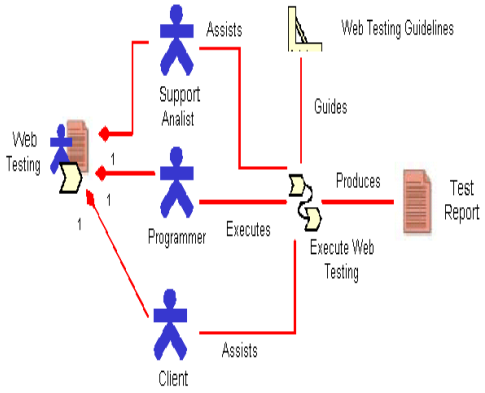


Figure 10. Web Testing

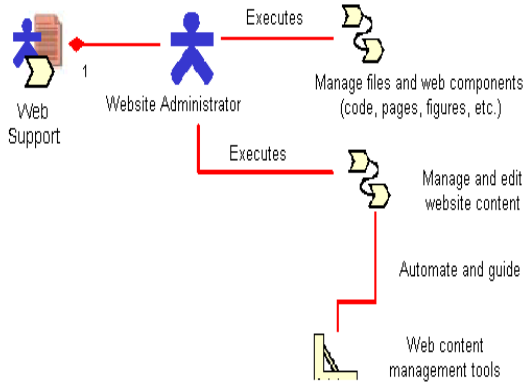


Figure 11. Web Support

Web Support (Figure 11)

The main focus of this discipline 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 former 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 assess the quantitative and qualitative implications of the disciplines incorporated in XWebProcess. To conduct the assessment, an experiment and a survey were executed to analyze XWebProcess in terms of effort and qualitative effectiveness 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.

4.1. Experiment Background

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 Universidade Federal do Pernambuco - 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) can 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.

The participating groups were introduced to each process by taking classes and reading detailed process definitions described in [3], relevant material such as [1,2,6], and presentations used during classes. An important observation is that students working in XP teams did not attend classes of XWebProcess before the experiment, what could have biased the experiment since students in XP teams could use XWebProcess extensions to perform their work.

The main goal of the experiment was to compare XP and XWebProcess in terms of effort and quality of the outcome. We wanted to verify if the adaptation done by adding elements into XP to create XWebProcess had any significant impact on the process agility, what will be shown by the quantitative data collected. Moreover, we also believed that the adaptations 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].

4.2. Experiment Planning

Object of Study: XP and XWebProcess.
Purpose: Evaluate the use of both processes in Web application development.
Quality Focus: Effort spent.
Perspective: System developers.

This section describes issues relating to the planning stage of the experiment. The planning stage addresses the setting of the experiment, describing the context and the environment where the experiment took place, personnel used and problem nature. It also states the statistical analysis framework and the key variables assessed throughout the experiment.

The experiment was conducted independently by eight groups. All groups were formed by five senior undergraduate students at Federal University of Pernambuco. Four groups used XWebProcess and the other four used XP to develop the same Web system. All groups had a previous training in XP and JUnit and the XP groups attended classes of XWebProcess only after finishing the experiment. The experiment was part of the Software Engineering course assignment and all groups had the same task, which was to develop the web bookstore system described in Section 4.1. Explicit guidelines were given to minimize communication by students from different groups. The experiment was grounded on the following guiding hypothesis:

- Null Hypothesis: NH – The effort spent to develop the system using XWebProcess is the same as using XP.
- Alternative Hypothesis: AH – The effort spent to develop the system using XWebProcess is different than XP.

The key *independent* variables analyzed throughout the experiment towards assessing the guiding hypothesis were:

- *Process Used***:** As described previously, 4 groups used XWebProcess and the other 4 used XP to develop the same system. All groups had twenty days to deliver the system and the releases were organized in two ten-day iterations.
- *Students experience in web and object-oriented development*: Most students had previous experience with web development but only a few had experience with the technologies chosen (JSP and Servlets). All students had previous experience (at least 2 years) with OO and Java and at least one year with database programming.
- *Tool Support*: All groups used the same tools to develop the web system: JBuilder as the IDE to create Java and JSP code, Microsoft Access as the DBMS and JUnit for unit test.
- *Methods and techniques used:* All groups used basically the same techniques that are provided by XP such as refactoring, pair programming and automated test. The difference was that for XWebProcess groups some techniques were used to the specific disciplines extended from XP. Table 3 presents the key differences.

The key *dependent* variable analyzed throughout the experiment towards assessing the guiding hypothesis was the *Total effort* spent to develop the example system. This was

measured in man-hour and the students were responsible to collect this data in a data spreadsheet when executing process activities.

The selection of what group would use XP or XWebProcess was randomized and the groups used the same process for both iterations. The material supplied to the subjects to perform the experiment varied according to the process used. *XP groups* were provided with the following material: A summary of the system's requirements; guidelines for using the process; XP class presentations and other descriptions of the process [1,2,3]; and the spreadsheet to collect the experiment data. *XWebProcess groups* were provided with the same material as XP plus the following: Guidelines for using the techniques with the extended disciplines.

Table 3 – Methods and Techniques Used*

Discipline	Technique Used	
	XP	**XWebProcess**
Exploration	Ad-hoc for prototyping activity	Prototyping by agile modeling, sketching in a paper or white board. The prototype was an output required for XWebProcess teams.
Define and Review Requirements	Layered architectural model for design architecture activity	Layered architectural model for design architecture activity
Design	Ad-hoc for design data layer activity	Ad-hoc for design data layer activity
Web Navigation and Presentation	Ad-hoc for whole discipline	Web navigation and presentation activity with "white site technique". The web navigation and presentation model was required as an output artifact.
Web Testing	Ad-hoc for whole discipline	Activities supported by web testing guidelines
Web Support	Ad-hoc for whole discipline	Activities supported by web support guidelines

* Details of the techniques are available in [32]

The experiment had factors that could impact the *validity of the outcome* such as unbalanced team compositions (skills, roles played, activities taken) and inter-team interference with regard to communication and learning of web technologies (JSP and Servlets). The approach taken was to minimize potential side-effects by imposing control layers surrounding the factors such as double-checking the homogeneity of teams after the random generation process and creating explicit "Chinese walls" to avoid knowledge and experience exchanges across teams. This will be further discussed in Section 4.3.

An *external validity* limitation of the approach taken in the experiment was the use of students as subjects and the limited size and complexity of the system developed in the experiment. The above choices can limit the generalization of our results towards large scale systems and team compositions formed by very experienced software developers. However, considering the limited availability of empirical studies involving agile processes our work can be considered a very important first step that can be leveraged and generalized by further investigation.

4.3. Experiment Operation

This section describes the operational phase of the experiment. The operational phase encompasses the experiment *preparation*, identifying the nature of interaction between experiment leaders and subjects and the instrumentation (training material, classes, discussions) performed. The *execution* section details how the members of the teams performed their activities and how they collected the data for analysis. Finally, the *data validation* section explains how the collected data was validated.

Preparation

Contact with experiment participants started by an inaugural class with all students. The students were exposed to the criteria for forming groups and each student was asked to answer a questionnaire with the following questions:

- What role (programmer, web designer, architect, etc.) do you prefer?
- What role you would not like at all?
- Suggest possible roles for other members of your team.

The possible roles to select were: web designer, programmer, website administrator, architect, client and tracker. By the time of the questionnaire there was not yet a decision about which groups would use XP or XWebProcess. The roles were equal for both processes even though XP does not properly define roles such as web designer and web administrator.

After defining each person's role in the teams we randomly assigned four teams with XP and four with XWebProcess. The next step taken was to schedule training for all teams as part of the instrumentation stage. The training was organized in the following manner:

- Two XP classes for all teams. Each class lasted for two hours and addressed the main features of XP. The presentation and parts of XP literature [1,2,3] were provided as reading material. Moreover, students were free to read any other XP material found on the web;
- One hands-on class on JUnit for all teams. Students implemented a simple application using JUnit following a tutorial that was created by the experiment leaders (paper authors) and explained during the practical session;
- Two XWebProcess classes exclusive to XWebProcess teams. In these classes the main features of XWebProcess were presented and some guidelines for working with the process were given. Reading material about XWebProces in [3] was also provided.

The preparation stage finished with an assessment session held between the experiment leaders where the team compositions were checked to avoid unbalances (skills, roles played, activities taken). The overall assessment was positive and the experiment was allowed to proceed to the execution stage.

Execution

The experiment was executed in two ten-day iterations and XP groups started earlier since they did not attend XWebProcess classes. Each iteration was organized to have three meetings with the experiment leaders to minimize deviations from the process.

The first meeting was to define system requirements and plan the stories to be implemented in the release. In this case the experiment leaders acted as clients and tried to minimize differences in requirements within different groups. The client of each team was responsible for asking questions to experiment leaders in case of any issues arising. This meeting happened at the beginning of the iteration.

The second meeting was held between the experiment leaders and the tracker of each team in order to have a feedback if the groups were performing their activities appropriately and also to check the current state of the data sheet. This meeting happened at the middle of the iteration.

The third meeting was held at the end of the iteration to validate the implemented requirements. A sample set of functional test cases was used with all groups. The testing for non-functional requirements was simplified only to check if the system worked with two different browsers (Internet Explorer and Netscape Navigator).

Each team was formed by 5 people organized according to the following roles: programmer, client, tracker, web designer, architect and website administrator. The responsibilities attached to each role and the rules governing their actions were as follows:

- *Programmer:* this is the main role for both processes. Each group should have three programmers and two of them could not play another role. In both processes the programmer performed the same task (implement the business and testing code in java).
- *Client:* was responsible for explaining system requirements to programmers and helping in other activities such as validating tests, writing user stories and setting priorities. In both processes clients worked the same way as described by XP guidelines.
- *Architect:* was responsible for defining the architecture of the system and the database design. In both processes the architect was responsible for defining the main structure of the system. Most systems were implemented based on a layered architecture previously known by students from other projects and detailed in [32].
- *Tracker:* acted as a project manager of the team and was also responsible for gathering the effort data and updating data sheets. Due to the simplicity of its tasks, the person playing this role had also to select another role to play.
- *Website administrator:* the responsibilities of this role included configuring IDEs and also integrating the business and web GUI code. The difference among processes, as described in Table 3, was that in XWebProcess guidelines were provided.
- *Web designer:* responsible for designing and coding the web user interface. In XWebProcess the web designer was responsible for executing the web design activity whose output was the web navigation and presentation design. In XP there was no explicit activity for web design so this role in XP was limited to coding. In both processes it was established that two persons had to play this role and one person could not play another role.

All team members received a reminder sheet of the responsibilities and rules attached to each role as part of the experiment's instrumentation.

Data Validation

During the development of the system, each group collected data using an Excel data sheet. It was explicitly instructed that all members of a group should collect 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 (at least once a day) of each group with the information gathered by the other members. An outline of the data sheet used is shown in Table 4.

Table 4 – Data Sheet Example

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)	-

The Data Sheet shows an example of how it was filled during the experiment. It shows that three team members (atfs, egml, pmb) were involved in the activity of requirements definition inside the XPDRR discipline. The code for each discipline (e.g. XPDRR) was defined based on names shown in Figure 5. The time spent to execute the activity is shown by the delta (one hour and twenty minutes) and during this activity thirty minutes were used to write the story cards. In this case, 30 minutes were spent writing the story cards and the other fifty minutes were spent interviewing the client and discussing about the system requirements. The main goal was always to get the total effort spent in one activity and this was clearly explained to students before and during the project. Therefore the most important information that Table 4 provides is the indication that 3 team members worked for one hour and twenty minutes in one specific activity.

Another important issue to point out about Table 4 is that it only shows one row of the complete data sheet that each group had to fill. The complete data sheet contains other rows for each team including data about all process activities. As mentioned in the last paragraph, students had detailed instructions on how to categorize and identify activities and how to fill in the data. Additional effort was dedicated to minimize distortions among data collected via meetings during iterations that checked if the data sheets were being properly filled. For a detailed example of a spreadsheet see [32].

4.4 Experiment Data Analysis and interpretation

Table 5 shows the results (effort in person-hour) obtained by the eight groups involved in the experiment. These data were collected during the experiment as each group performed

the activities defined in the process followed (XP or XWebProcess) and filled in the time spent as well as the number of people in the spreadsheet shown in Table 4. When the project was finished the data collected was gathered resulting in the total effort spent by each group shown in Table 5.

Table 5- 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 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: $\mu_x = \mu_y$, i.e. the expected mean values of the effort spent are the same.
AH: $\mu_x \neq \mu_y$: Reject NH if $|t| > t_{\alpha/2,f}$ where f is the degree of freedom and α is the significance level. $t_{\alpha/2,f}$ 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 presented in Table 6.

Table 6. Data Analysis

General Results		
	XP Teams	XWebProcess Teams
Mean	104.175	113.9125
Variance	169.8617	303.1819
Standard Deviation	13.0331	17.4121
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 $t_{\alpha/2,f}$ can be seen in statistical tables [25] and in this case represents the critical value for the two-tailed t-test ($t_{\alpha/2,f} = 2.446$ for $\alpha = 5\%$ and f = 6). Therefore $|t| < t_{\alpha/2,f}$ and the null hypothesis (NH) is true meaning that the effort spent in both processes is the same.

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 Table 6 (104.1 in XP and 113.9 in XWebProcess) and also by the data shown in Table 5. The higher number of process elements such as disciplines, activities and artifacts in XWebProcess demands more effort, what partially 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.

Moreover, comparing the total efforts presented in Table 5 and the data in Table 6 we can see that the effort spent by XP teams (T1, T2, T3 and T4) were similar (close values and small standard deviation compared to average) and also that the same happens to XWebProcess teams (T5, T6, T7 and T8). This helps to confirm (although it does not prove) that the groups were homogenously divided and that there were no big distortions on how the teams performed the activities and collected the data.

5 Qualitative analysis

As well as collecting quantitative data, qualitative data based on a survey was also collected during the experiment. All subjects involved in the experiment were asked to answer a survey, which helped to assess participant's views on both processes.

An important point to note is that although groups following XP did not attend classes on XWebProcess prior to the experiment, after the experiment had finished, they were also given the classes on XWebProcess in order to answer the survey. The main reason for choosing this approach was to avoid XP groups from using the extended disciplines from XWebProcess during the experiment, thus invalidating the outcome. Special care was also taken during XWebProcess classes given to XP teams to avoid the possibility of students answering survey questions based on the researchers view of XWebProcess instead of their own experience ─ to minimize this problem the main focus of classes was on the definition of XWebProcess elements such as disciplines, activities and artifacts. Therefore, the approach was to emphasize the "what" and not the "how" issues regarding XWebProcess.

The survey asked questions about the quality of both processes. The goal was to obtain feedback regarding: how easy it is to understand the processes; how well the processes are defined and structured; to what extent the SPEM modeling helped to understand and follow the process; and which of the compared processes (XWebProcess, XP) was considered more suitable for web application development. Therefore, the answers enabled to verify if the modeling done added value to developers and also if the adaptations done in XP towards developing XWebProcess (via additions and extensions of disciplines, artifacts, roles, and activities) provided a better approach to building web applications. The key questions asked in the survey were as follows:

1) *Easiness to learn:* how easy it is to understand the process definition, and how the SPEM modeling helps in your understanding of the process?

( ) Badly Defined: activities are not well defined. It is difficult to understand the organization of the activities and the roles played in the process.
( ) Reasonably Defined: activities are defined and explained. However, there is no clear definition of the sequence of activities through time and the responsibilities attached to each role.
( ) Well-Defined: activities are well-defined and explained. There is also a clear definition of the sequence of activities through time and responsibilities attached to each role.

2) *Visibility:* do process activities help to produce visible outcomes facilitating progress checking? Hint: assess the process according to the inputs and outputs of each activity (artifacts created, updated, discarded).
( ) Not visible: there is no clear view of what artifacts are created and updated in each activity and who is responsible for it.
( ) Visible: activities clearly define what are the input and output artifacts and who is responsible for them. However, the importance attached to each artifact is not clear.
( ) Highly visible: activities clearly define the input and output artifacts and who is responsible for them. In addition, there is a clear definition of the importance of each artifact.

3) *Easiness to evolve:* do you believe the process can easily evolve to adapt to changes in the organization or to provide improvements leading to better process quality? Do you think the process can be easily extended towards incorporating new elements or adapting existing elements?
( ) Difficult to adapt: due to an inappropriate definition it is difficult to understand the process and therefore tailor to changing requirements in the organization.
( ) Easy to adapt: the process is well-defined and structured simplifying its adaptation towards tackling specific projects and organizational requirements.

4) What are the main advantages and disadvantages of working with your process (XWebProcess or XP)?

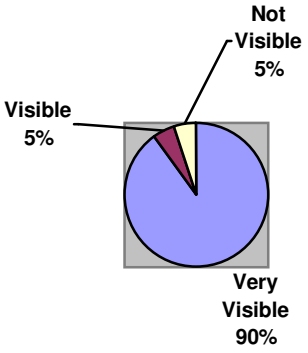
5) Compare XP and XWebProcess. Do you think the elements inserted and adapted in XWebProcess are necessary (analyze each of the six disciplines modified or inserted)? What benefits and disadvantages they provide? Which process is most suitable for Web development? Give reasons and examples to support your answer.

The survey was distributed and answered by all subjects involved in the experiment. Some core results of the survey were:

- 100% of the subjects involved mentioned that XWebProcess is more suitable for Web development than XP. All subjects reported that XWebProcess activities like: Web User Interface and Navigation Design help in producing better websites in terms of organization and appearance.
- 100% of the subjects involved reported that the use of prototyping sessions benefits requirements gathering and improves communication with clients. They also pointed out that the Web Testing discipline is very important to verify constraints like time, load and security;
- All subjects that used XWebProcess reported that despite having more activities and artifacts than XP they felt that using this process really added value given that you can remain agile whilst achieving better quality outcomes;
- Almost all (95%) of the subjects that used XP agreed that the process needs to be adapted and tailored to Web development as done in XWebProcess;
- 100% of the subjects that used both processes mentioned that the modeling in SPEM helps to give a clear understanding of the process structure and also facilitates to adapt the process to the organization and project needs.

- All subjects pointed out that they enjoyed working with the process (XP or XWebProcess) and were interested in continue using in other projects. Some of the key advantages relating to both processes were: there is less formalism and less documentation in favor of more communication and interaction among team members what made them work faster; they also reported that pair programming helped to facilitate the understanding of the system; and that automated testing provides a better way to produce code and minimize errors helping to improve quality.

Other interesting results stemming from the survey are shown next. We analyzed the data collected and divided it in two groups. Group A is formed by subjects who used XP and Group B by subjects who used XWebProcess.



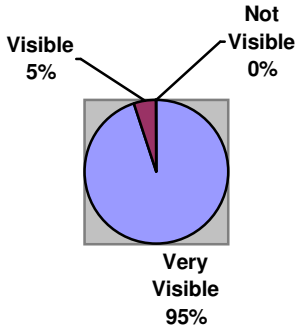


Figure 12. Visibility of XP - Group A Figure 13. Visibility of XWebProcess - Group B

Figure 12 and Figure13 show a visibility assessment for each process (question 3 in the survey). The goal of the assessment is to verify if the definition of each process enables a proper understanding about process dynamics and structure. Most subjects in groups A and B reported that both processes are easy to understand due to its definition using SPEM which facilitated following process activities as the project progresses. Moreover, subjects also reported that artifacts produced or consumed in each activity are highly visible and also the responsibilities attached to the production of artifacts. Students from group A also reported that modeling XP in SPEM helped in the understanding of the XP's structure and how XP's practices relate to each other. For more information about the survey see [3,32].

6. Summary and Conclusions

The construction of high quality software complying with severe project delivery time constraints is a major challenge faced by the software engineering community. This challenge is being addressed on several fronts (software engineering processes, techniques, principles and tools), both in academia and industry. Within the process front, agile processes such as Extreme Programming are emerging as a viable alternative to traditional development methods, helping to speed up software development via a flexible and effective process philosophy. The agile way of building software relies in intense and direct team communication, extensive client cooperation, less formalism and less documentation.

Agile methods are growing in popularity and have already shown encouraging results in several complex project developments [17].

Although important advances have been discussed in the literature, few results focus on the effort and time spent in agile processes from an experimental perspective. The results available so far have been limited to pointing out that projects developed with agile methods were delivered on time and on budget, but without analytically comparing the agile methods to identify how much more efficient they are or why they make software development faster.

This paper contributes to the agile methods empirical research field by studying two agile processes XP [1,2] and XWebProcess [3] from an experimental perspective. The first is the most popular agile process existing today and the latter is an XP-based web development process specially tailored towards reconciling quality and speed in web application development [37]. XWebProcess [3, 32, 38] seeks to retain the agile properties of Extreme Programming whilst maintaining development disciplines that have a positive impact in the quality of the delivery. The set of process disciplines incorporated to XWebProcess have been chosen after assessing the key characteristics of web application projects that demand special consideration within the process structure.

As web applications frequently represent strategic business opportunities to several companies, they also have to be built and delivered complying with severe project delivery cycles. The experiment conducted had the purpose to investigate how XWebProcess and XP would perform when compared through an experimental setting involving the development of a typical web application project (web bookstore).

The experimental results obtained are evidences of the usefulness of XWebProcess in supporting the development of high quality web applications whilst retaining agile delivery properties, therefore contributing to deliver web applications on time. The results illustrated that the effort spent in both cases were the same, reinforcing the null hypothesis underpinning the experiment. Simply stating, although XWebProcess has more elements (disciplines, artifacts and roles) tailored to promote superior software quality outcomes, the effort spent to enact the extra disciplines does not have a statistically significant impact in increasing the overall effort of the project.

In addition to the quantitative assessment performed, the qualitative survey executed obtained important feedback relating to the quality of the process model in curbing the developer's learning curve and in facilitating process structure extension given the high visibility promoted by process models specified explicitly using SPEM.

Although the experiments and surveys conducted gave important feedback on the efficiency and effectiveness of both processes (XP, XWebProcess) in supporting web application development, there are still challenging issues ahead that need to be explored in the quest for further advancing the empirical assessment of agile methods. Some of the open research issues are:

- How speed and quality assessments vary when the experiment is performed in a professional software development organization as opposed to a classroom experiment;
- How speed and quality indicators unfold on mid and large scale web application projects (e.g., more than 50000 LOCs);
- How XWebProcess and XP perform when compared to other processes that are also tailored to web application development (e.g., AWE [16] and OPEN [15]);

- How XWebProcess and XP perform when compared to "heavier" processes like RUP [14].

The frameworks and results described in this experimental study also provide an important foundation case study for tackling the above research issues, thus, empowering the agile methods community with a tried and tested roadmap for executing empirical software process assessments.

Acknowledgments

The authors wish to thank the external reviewers and the editors, prof. Julio Leite and prof. Shari Lawrence Pfleeger for their extremely valuable suggestions and thorough revision of the submitted manuscript. Americo and Alexandre wish to thank Universidade Federal de Pernambuco (UFPE) and the Brazilian Research Council (CNPq) for supporting this work. Pedro wishes to thank the Starting Lecturer Scheme of the University of Manchester for supporting his work.

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: An Agile Process for Web Application Development (In Portuguese).* 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.* XP2002 – Extreme Programming Conference, Available at: www.xp2002.org/atti/Heinecke-Noack--ConstructingAgileSoftwareProcesses.pdf
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. Kappel, G. et al. *Web Engineering – Old Wine in New Bottles?*, In proc. of 4th Intl. Conference on Web Engineering, LNCS Vol. 3140, pages 6-11, 2004.
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. *XWebProcess website*. Available at: http://www.cin.ufpe.br/~atfs/xwebprocess.
33. Rombach, Dieter. *The Dilemma of Software Development*, Quality Connection, May 2002.
34. Szyperski, Clemens. *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 2002.
35. Kleppe, A., Warmer, J., Bast, W., *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley, 2003.

36. Clements, P., and Northrop, L., *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2001.
37. Sampaio, Américo, Vasconcelos, Alexandre and Sampaio, Pedro R. Falcone, *Towards Reconciling Quality and Agility in Web Application Development*, In proc. of 1st. International Workshop on Web Quality - WQ'04, Munich, Germany, 2004 (to appear).
38. Sampaio, Américo, Vasconcelos, Alexandre and Sampaio, Pedro R. Falcone, *XWebProcess: Agile Software Development for Web Applications*, LNCS Proceedings of the International Conference on Web Engineering (ICWE 2004), pp 597-598 Munich, Germany, 2004.