Category: Software & Systems Design

Extreme Programming for Web Applications

Pankaj Kamthan
Concordia University, Canada

INTRODUCTION

The engineering environment of Web Applications is in a constant state of technological and social flux. These applications face challenges posed by new implementation languages, variations in user agents, demands for new services, and user classes from different cultural backgrounds, age groups, and capabilities.

We require a methodical approach towards the development *life cycle* and maintenance of Web Applications that can adequately respond to this constantly changing environment. In this article, we propose the use of an *agile methodology* (Highsmith, 2002), namely Extreme Programming (XP) (Beck & Andres, 2005), for a systematic development of Web Applications. In general, agile methodologies have show to be cost-effective for projects with certain types of uncertainties (Liu, Kong, & Chen, 2006) and, according to surveys (Khan & Balbo, 2005), been successfully applied to Web Applications.

The organization of the article is as follows. We first outline the background necessary for the discussion that pursues and state our position. This is followed by a discussion of the applicability and feasibility of XP practices as they pertain to Web Applications. Then the shortcomings of XP towards Web Applications are highlighted, and suggestions for improvement are presented. Next, challenges and directions for future research are outlined. Finally, concluding remarks are given.

BACKGROUND

Over the last decade, Web Applications have become increasingly large and complex as they respond to the expectation of sophisticated services. The need for managing increasing size and complexity of Web Applications and the necessity of a planned development was realized in the late 1990s (Coda, Ghezzi, Vigna, & Garzotto, 1998; Powell, Jones, & Cutts, 1998). This led to the introduction of the notion of Web Engineering (Ginige & Murugesan, 2001), which subsequently has been treated comprehensively (Kappel, Pröll, Reich, & Retschitzegger, 2006).

In the last few years, a number of methods for realizing Web Applications have been proposed including, but not limited to, Web Site Design Method (WSDM) (De Troyer & Leune, 1998), Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe & Rossi, 1998), and Web Object-Oriented Model (WOOM) (Coda, et al., 1998). However, the focus in these approaches has been on specific aspects of Web Applications (like modeling, designing, or implementing) rather than on the *process*.

We adopt the most broadly-used and well-tested *agile methodology*, namely XP, for the development of Web Applications. There are several differences between traditional software and Web Applications (including aspects of delivery, legality, privacy, security, and *usability*) that makes the realization of XP challenging.

XP is a test-driven, "lightweight" methodology designed for small teams that emphasizes customer satisfaction and promotes teamwork. XP was created to tackle uncertainties in development environment, and in doing so, put more emphasis on the social (people) component (engineer, customer, and end-user). The XP practices are set up to mitigate project risks (dynamically changing requirements, new system due by a specific time line, and so on) and increase the likelihood of success. The use of XP has been suggested for a "rapid application development" of Web Applications (Maurer & Martel, 2002; Wallace, Raggett, & Aufgang, 2002), however, a detailed analysis has not been carried out.

It is not the purpose of this article to evaluate the merit of XP on its own or with respect to other *agile methodologies*; such assessments have been carried out elsewhere (Mnkandla & Dwolatzky, 2004).

ENGINEERING WEB APPLICATIONS USING EXTREME PROGRAMMING

In this section, we discuss in detail how the *twelve practices* put forth by XP manifest themselves in the development of Web Applications (Table 1).

We note that some of these practices such as *Testing*, *Refactoring*, or *Pair Programming* are not native to XP and were discovered in other contexts previously. In this sense, by aggregating them in a coherent manner, XP bases itself on "best" practices. These practices are also not necessarily mutually exclusive and we point out the relationships among them where necessary. We also draw attention to the obstacles in the realization of these practices that pose challenges to the deployment of XP for Web Applications.

Table 1. XP practices corresponding to process workflows in a Web application

Process Workflow	XP Practices
Planning	40-Hour Week, The Planning Game (Project Velocity)
Analysis (Domain Modeling, Requirements)	On-Site Customer, The Planning Game (User Stories)
Design	Metaphor Guide (Natural Naming), Simple Design, Refactoring
Implementation	Collective Ownership, On-Site Customer, Metaphor Guide, Coding Standards, Pair Programming, Continuous Integration
Verification and Validation	On-Site Customer, Testing (Unit Tests, Acceptance Tests)
Delivery	Small Releases

1. *The Planning Game.* The purpose of *The Planning Game* is to determine the scope of the project and future releases by combining business priorities and technical estimates. For that, it first solicits input from the "customer" to define the business value of desired features and uses cost estimates provided by the programmers. This input documented in form of *user stories* (Alexander & Maiden, 2004). A *user story* is a user experience informally expressed in a few lines with a Web Application such as navigating or using a search engine. The estimation is limited to the assessment of *project velocity*, a tangible metric that determines the pace at which the team can produce deliverables. The plan is iterative: it is prone to modifications based on the current reality. For example, a new user story may lead to revisitation and evolution of the current plan.
2. *Small Releases.* The idea behind *Small Releases* is to have a simple system (an evolutionary prototype) into production early, and then via short cycles, iteratively, and/or incrementally, reach the final system. To have a concrete proof-of-concept up-and-running can be used to solicit feedback for future versions and can help convince customers and managers of the viability of the project. This is useful for Web Applications that are highly interactive such as those making broad use of fill-out-forms. However, there is cost associated with prototypes and, therefore, their number should be kept under control.
3. *Metaphor Guide.* The use of metaphors (Boyd, 1999) is prevalent in all aspects of software development. A *Metaphor Guide* is an effort to streamline and standardize efforts for naming software objects and is available for team-wide use. The main concerns in naming are of sensitivity to the domain under consideration (that is, use of terminology of the application area) and familiarity (as user background is assumed to be non-technical). *Natural naming* (Keller, 1990) is a technique initially used in source code contexts that encourages the use of names that consist of one or more full words of the natural language for program elements in preference to acronyms or abbreviations. Indeed, natural naming strengthens the link between the underlying conceptual entity and its given name. For example, DeviceProfile is a combination of two real-world metaphors placed into a natural naming scheme.
4. *Simple Design.* The motivation behind a *Simple Design* is that in XP's view, requirements are *not* complete when the design commences. This is inline with the reality of Web Applications which have to respond to the market pressures and the competition that are beyond their control, or other unavoidable circumstances such as variations in implementation technology. Therefore, the design is minimal based on *current* (not future) requirements. It aims for simplicity, and to ensure "good" design, its quality (specifically, structural complexity) is improved by frequent revisitations; that is, *Refactoring*. The interfaces of Web Applications are particularly amenable for simple design as they are likely to change often during the process. Design patterns (Van Duyne, Landay, & Hong, 2003) are forms of reusable knowledge based on past experience and expertise that can lead to simplified design.
5. *Testing.* There is a strong emphasis in XP on validation and verification of the software at all times. By being test-driven, there is transition from one phase to another only if the tests succeed. The tests range from *unit tests* (using tools such as HTMLUnit, HTTPUnit, XMLUnit, XSLTUnit, and JUnit) written by programmers to acceptance tests involving customers