# Using Extreme Programming for Web Applications Development

Valentina M. Milićević

*Abstract*--**Web applications are becoming very complex and composite, therefore it's necessary to adopt a methodology or a model of proceeding, which will guide designers through process of application development with hope that the risk of abortion of the project will be reduced, to simplify composite cloth and generally to ameliorate finite result. Extreme programming model of development, composed from good concepts, can be adopting for Web applications, also for the others software's systems.**

*Keywords*—**Extreme programming, Web application, the model of software development, Cascade model**

## I. Introduction

Web applications are becoming more and more popular. This is in part due to the rapid deployment of the tools and technologies for developing them. The process of developing of the Web application is often simple: a developer built application, in browser he adduced how the application looks like, then he offer the application to publicity. In the industry of software most of professionals will be agree that those informal methods are good only for small projects on which works only one programmer and for which is expecting that in the future will not demand circumstantially observance. Because of that, there are many problems of Web application, which realized by informal methods. The programs, which made without any plan, comprise the complicate program's logic, which is very difficult to preserve.

Developers made mistake when they develop Web applications, because they attend on the selection of the tools the most, whilst they does not attend on the process of the development of Web application a lot. Current development environments make it so easy to produce simple web applications that they have the unfortunate side effect of encouraging the developer to develop and evolve applications in the absence of serious analysis and design. Web applications are no longer simple. They have evolved into complex and mission-critical systems. Today, building a feature-rich Web application requires a team of developers and a strong development process. Therefore, developers should adopt a model of proceeding, which describe the different phases of development, in order to avoid leastwise those primary difficulties at developing application. Then, the developer of the project can accomplish each step precisely and he can be sure that he accomplishes every step on the proper way, using defined line of march all the time and documentation. Ideal model of proceeding for building Web application enable to developer to perceive complexity of application, to decrease risk of failure of the project, to struggle with certainly modifications during developing of the project and to deliver application very fast.

In recent years, Extreme Programming (XP), which will be describing in this paper, has been advocated as an appropriate programming method for the high-speed, volatile world of Internet and Web software development. XP can be characterized as a "lightweight" or "agile" methodology. Also, XP represent discipline of software development based on simplicity, high-band-width communication, feedback, and courage. This model maintains adequate method for very fast Web applications development, which represent very important factor. If developers use this model of development software, then customer will receive desirable application for beforehand deadline. Using this model, it can be decrease risk of the project abortion.

## II. Extreme Programming Model Description

Extreme Programming (XP) represents important approach in software development, which contains the collection of principles and activities, which procure the designing of secure and quality software [1,2,3]. This model is based on fast development and delivery of application to customer, planning and constant testing. XP look attention on development project with risk. If customer requires software delivery until deadline, the risk will grow. XP has goal to alleviate risk and aggrandize probability of project issu. This application development model has designed for small programmer teams.

An XP approach emphasizes customer involvement and testing. Customer from the beginning participates with XP developers' team in developing of project.

In XP, the basic partition rule is on customer and programmer. Customer and programmer are in the same team, but they have to retrieve different decisions. This division of labor helps keep the whole team on track by making the consequences of decisions visible. The costumer has to see what he gets, while the programmer has to define price of building application. This show up in who gets to make which decisions. The customer gets to decide: scope (what the

V. M. Milićević, is with the Faculty of Electronic Engineering, University of Niš, Beogradska 14, 18000 Niš, Yugoslavia (telephone: 381-18-529-524, e-mail: valentina@elfak.ni.ac.yu).

system must do); priority (what is the most important); composition of releases (what must be in a release to be useful) and dates of releases (when the release is needed). The programmers get to decide: estimated time to add a feature; technical consequences (programmers explain the consequences of technical choices, but the customer makes the decision); process (how the team will work) and detailed schedule (within an iteration).

The XP project development has three phases:

1. Release planning phase, where the customer writes stories, the programmers estimate them, and the customer chooses the order in which stories will be developed;

2. Iteration phase, where the customer writes tests and answers questions, while the programmers program;

3. Release phase, where the Programmers install the software, and the customer approves the result.

Figure **Fig. 1.** shows the process of XP model [2]. During the release planning phase, and all along the way after that, the developers will seek an effective metaphor that helps guide their solutions. The team adopts system metaphor. This helps orient developers when they are trying to understand the functionality at the highest levels. Members of the team define the names for the objects of problem and relationships between them. The metaphor may change over time as developers learn more about the system, and as they get inspired in their understanding of it. A release is a version of a system with enough new features that it can be delivered to users outside the development group. The goal of release planning is to help the customer identify the features of the software they want, to give the programmers a chance to explore the technology and make estimates, and to provide a sense of the overall schedule for everybody.

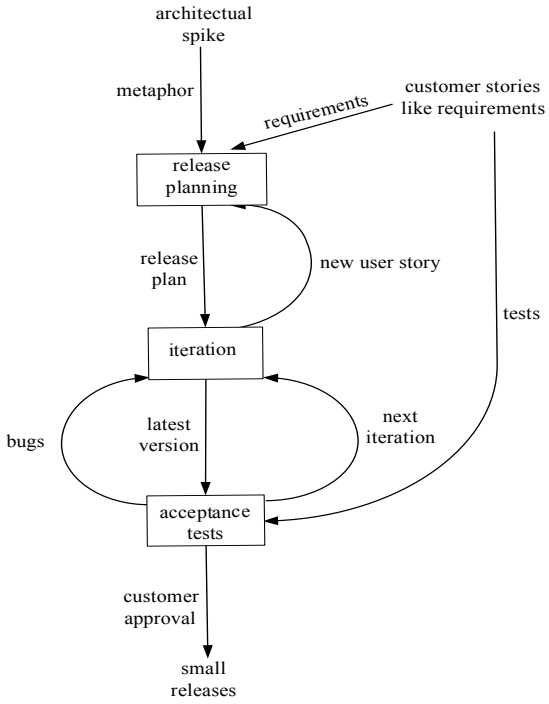


Fig. 1. Phases of the software devolopment by XP model

The release planning has two phases: exploration phase (fig. 2) and planning phase.

The customer writes a story, which is simplier then classical writting of documents with specification of requirements. When customer write a story, he has not to give any technical detail. Basis on story, the programmer has to define deadline of the release. When the programmer starts to implement a story, he will contact the customer to get more details of the story. The customer has to writting a story carefully, because the story must be testable. The customer has to specify the tests (later on), so they should have in mind some mechanism by which to test it. In the case that the story is too big, the customer has to split a story. In the case that the programmer does not know how to implement a story, he has to spike a story. The team has the opportunity to do spikes: quick throw-away explorations into the nature of a potential solution. Based on the stories and the spikes, developers decide on an approach to the system's structure. The developers can past on the planning phase after the story builds. Exploring will be done when customer cover the whole demands with stories.

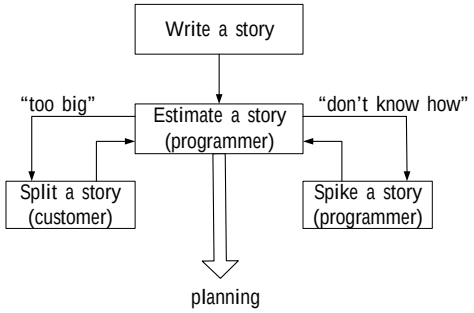


Fig. 2**.** Exploration phase

In the planning phase, developers plan releases of some versions of application. Customer need to sort stories by value, from most to least valuable, or at least labeled high, medium, or low. After that, programmers classify stories by risk (optional, like high, medium, or low) and declare the velocity. Declare the velocity, that is, how many story points the customer should expect per fixed-length iteration. After declaring the velocity, the customer has to choose scope and stories for the next release. To judge how long the development effort should take, it has to divide the total story point estimates by the velocity. For the first release, the stories must exercise the whole system end-to-end, even if at a minimal level. At this way, developers procure release plan of the system and, after that, they can traverse on the iteration phase.

The goal of iteration planning is to take the stories a team plans to implement in this iteration (the stories currently most valuable to the customer), break those stories into smaller tasks, and assign programmers to work on the tasks. Iterations are of a fixed length. Iterations are time-boxed: if the team cannot get everything done, they will drop features rather than slip the deadline of the iteration. At the end of each iteration, developers should expect to see the system ready to deliver,

running the acceptance tests for the stories they have chosen. On the first day of each iteration, the team will decide which stories to focus on. The iteration plan will identify what tasks will be done, and who will do them. The team accomplished some number of story points in the previous iteration. The stories do not have to be in the order they were in the release plan; the customer can request them in whatever order they like. In fact, the customer may introduce new stories if they are willing to give the team time to estimate them. In iteration it can be implement just those stories which were planed for that iteration. When the iteration is finished, developers have to deliver that version of system, like results of that iteration. This version of the system can be test, after iteration phase, whereby for each iteration phase the tests need to be create. The tests are creating basis on the story of customer. For each story, a test can be creating. After testing, customer analyzing results of tests. Stories, which are not show the positive results on the testing, have to be return in iteration phase on implementation over again or on the removing bugs from beforehand implementation. After realization and testing, customer has to approve delivery of application, wherewith the process of designing and realization is done.

XP can be summarized by twelve practices [1, 2, 3]. Although many other practices can be considered part of XP, but these twelve are the basic sets:

- Metaphor enables the better communication between members of development team and guides all development with simple shared stories of how the whole system works. XP encourages stories, which are the brief descriptions of system task. Metaphor expresses the evolving project vision that defines the system's scope and purpose, helps in generating of new apprehensions of the problem and system, and helps directly to the architecture of system.
- Planning Game encompasses requirements definition and project planning. Customer defines application features with stories. Programmers prioritize the stories and schedule the most important and difficult for the next iteration. Only the programmers who work on a story may estimate how much time it will take to complete. Tackling the most difficult tasks first is to reduce the overall risk associated with the project.
- Small releases containing the most valuable business requirements are used to build the system. Releases should be delivered very often so customers get to see and touch the working product on a regular basis.
- Simple Design focuses on delivering a system that meets the customer's immediate needs - no more and no less.
- Testing is continuous. Programmers write unit tests to validate correct operation of modules before they write functional code for the module under development. Customers then write system tests to demonstrate that requirements have been satisfied.
- Refactoring is the process of restructuring the system to remove duplication, simplify code and add flexibility without changing how the code operates.
- Pair programming is the practice of having two people working together on all production code. They do this as full partners, taking turns typing and watching, to provide constant design and code review. This is the most controversial aspect of XP.
- Collective ownership, which lets any programmer change any code in the system at any time, is similar to open source programming. This approach is markedly different from traditional methods in which a single developer owns a set of code. XP proponents argue that the more people who work on a piece, the fewer bugs will occur.
- Continuous integration is a day-to-day activity. Code is integrated and test after a few hours or a day at the most. Integration of one set of changes at a time simplifies the integration process and makes it obvious who is responsible for fixing the code when integration tests fail. Continual regression testing means no regressions in functionality as a result of changed requirements.
- Forty-hour workweeks are highly encouraged on XP projects. Having rested, motivated developers boost productivity.
- On-site customer is a designated person who works with the team and is available to answer questions, resolve issues and set priorities. The customer, after all, is the final arbiter of system acceptance. This customer representative remains with the development team throughout the project.
- Coding standards are a mandated requirement, not a set of guidelines. Programmers follow common rules so all code in the system looks as one person wrote it. Create code standards that work for team and consistently apply them.

## III. THE OTHER MODELS FOR WEB APPLICATION DEVELOPMENT

Model which can also be used for Web application development is Cascade model or Waterfall model, which is shown at fig. 3 [4]. The model starts with planning phase, then proceeds on the designing phase, then on realization and testing and finished with keeping phase. Mention phases represent separate steps, but proceed from one in another phase has not always been explicitly emphatic. Furthermore, sometimes there is need to repeat some steps, if the project has been changed. If designer practice this model, then he can planning everything beforehand. That is the biggest weakness of this model. The second demerit of this model is that the all phases are partly overlapped. Every phase influences as on anterior thus on subsequent phase, while some of them need to be repeated. Cascade model do not withstand big modifications. However, this model for Web application development is popular, because of it’s understanding and applicable.

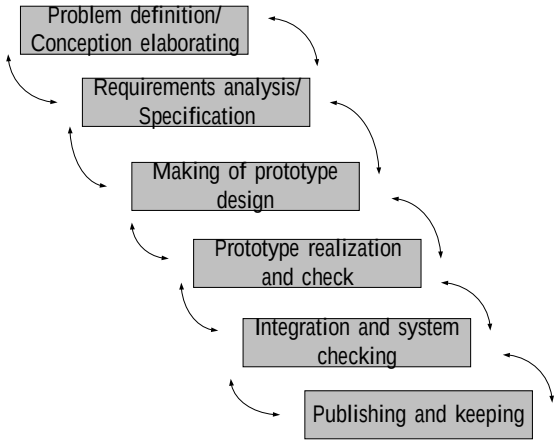


Fig. 3. Cascade Model

The important characteristic of this model is it's beforehand planning. However, because of need to pass all steps, many designers haste thought earlier phases of model and finished with repeat some steps. This model does not maintain too explorations, because of what in whole proposition can be introduced unnecessary risk. Maybe it's possible to amend designing in the case of longer moderation on the primordial phases. The model, which more times reversion on the same development phase, is Cascade Model with Swirl, which is shown at fig. 4. This model is good at projects developing, which comprehend many unknown factors.

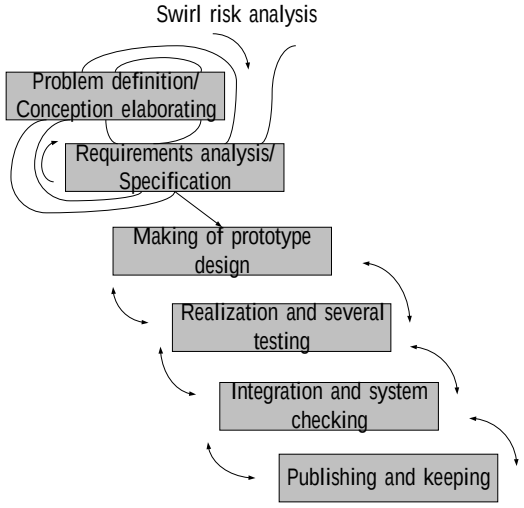


Fig. 4. Cascade Model with Swirl Risk Analysis

## IV. Extreme Programming from a Cascade Model perspective

The values of XP should be captured in any modern software project, even if the implementation may differ radically in other environments. Communication and simplicity may be stated in other terms (coordination and elegance, for example), but without them non-trivial projects face almost insurmountable odds.

To the phases problem definition, apropos conception elaborating, and requirements analysis, apropos specification, correspond release planning phase of model XP. Good feature of model XP is strong definition of the problem and analyzes of customer's requirements in release planning phase. Communication and simplicity are fundamental factors of XP model. Obscurities eliminate at the specification of requirements, because the customers are always accessible to the programmers. At this way, that remove the possibility that the customers get application like they does not want, what is not represent in Cascade model.

Function of XP model is to increase flexibility of the strategy of the project. XP looks attention on developing of the projects with certainly risk. Using this model, it can be decrease risk of the project abortion and increase possibility of project success, what is not case in Cascade model. Cascade model does not maintain too explorations, because of what the whole project can be introduced to unnecessary risk.

Substantially deference between these two models is in the prototyping phase. Phases making of prototype design and realization and check have not their double in XP model, because the XP do not maintain prototype making. Model XP maintain parts of system creating based on customers stories, which will be pooled and integrated in system (iteration phase).

Integration and system checking of the Cascade model correspond iteration phase of the model XP in which customer writes tests and where the checking is occur. After every story implementation in XP, there is need to integrate story in system and to check system and implemented story.

As a system becomes larger, some XP practices become more difficult to implement. As projects becoming larger, emphasizing a good architectural "philosophy" becomes increasingly critical to project success. Multi-discipline teams are also problematic since XP is aimed at software-only projects.

Basic objection on using model XP at the developing Web applications is that XP implies management and organization infrastructure, until more focuses on the processes of software engineering and technical work.

## V. Conclusion

Using traditional approach at the developing Web applications, often missing some features and the quality is not as customers expecting for. Deficiencies of traditional model of software development are solving in XP model.

XP has to use in any project based on Web, and in any modern software project. Most of XP consists of good practices that should be thoughtfully considered for any environment.

## References

[1] William C. Wake, "Extreme Programming Explored", 2000.
[2] James D. Wells, "Extreme Programming: A gentle introduction", www.ExtremeProgramming.org, 2001.
[3] Gordon Benett, "Extreme Programming (XP) Holds Promise For Intranet Development ", intranetjournal.com/articles/200110/gb_10_10_01a.html
[4] Thomas A. Powell, "Web design: The Complete Reference", The McGraw-Hill Companies, 2000.