EXPERT APPROACH FOR E-BUSINESS SOFTWARE DEVELPOMENT

Sylvia Ilieva[1], Eliza Stefanova [2]

[1] – *Department of IT, FMI, Sofia University, 5 J. Bouchier str. P.B. 48, Sofia 1164, BULGARIA ,+359 2 71 71 27, + 359 2 656 157, sylvia@acad.bg*

[2] – *Department of IT, FMI, Sofia University, 5 J. Bouchier str. P.B. 48, Sofia 1164, BULGARIA , +359 26256511, + 359 2 656 157, eliza@fmi.uni-sofia.bg*

1. Introduction

All Small and Medium Enterprises (SMEs) developing e-commerce and e-business software have similar business objectives as less time to the market, better quality, and cheaper cost. Current projects are very large and complex and it becomes difficult to handle them. There are different private methods and techniques developed to overcome this problem. The latest studies on the well-known software development methodologies and their applicability to e-projects reveal that they do not fit very well the priorities, the abilities and the corporate culture of SMEs developing e-business or e-commerce applications. Traditional methodologies assume that if the programmers tried hard enough, they could anticipate the complete set of requirements early and reduce cost by eliminating change, which can be reasonable in stable environments, but it is not the case at all in e-projects. In the fast changing environment of e-projects change cannot be eliminated; rather than eliminate rework, the new strategy is to reduce its cost, while retaining quality.

There are a number of reports showing increased productivity and software quality by applying Extreme Programming (XP) principles [1]. However, even projects that have adopted several or all XP practices meet project management problems, which are related to estimating and planning the project, both in terms of time and costs. To overcome this obstacle European project *e*XPERT (IST-2001-34488) has the objective to define agile and integrated approach guiding development of e-projects. The approach, named also eXPERT, focuses on combining XP and Personal Software Process (PSP) practices [2].

The main aims and logics of eXPERT project and underlying software methodologies are presented in this paper. The paper is organized as follows: Section 2 presents the objectives of eXPERT project. In section 3 we describe the basic software engineering practices, which will be used – XP and PSP. Section 4 discusses issues in the process of combination of XP and PSP principles. Section 5 concludes the paper.

2. eXPERT objectives

The eXPERT project aims to raise the skills, efficiency and quality of SMEs in the e-project development industry, whilst also teaching them how to develop and modify software at a low and manageable cost, regardless of the development phase. Actively improving the quality of SME software will raise the standards (benchmarking) for young start up companies, and by example (dissemination of Best Practice Cases and Generic Model) other young companies will be able to follow suit and to realise the benefits of radical change.

The first aim of the project is to define a light approach for software development – named eXPERT – combining the principles of XP and PSP, which will address the goals and the specifics of e-projects (e-business and e-commerce software projects). The project will teach developers in different SMEs how to work effectively in an atmosphere of often changing requirements; to construct the simplest system that has business value; to accurately estimate development work to avoid schedule slips, overcosts; to work effectively with the customers; to maintain software code in top condition. The project consortium establishes a network of Centers of Expertise (CE) in Spain, Germany and Bulgaria, and User companies associated to each CE. Seven pilot projects will be run at the User companies experimenting the application of the defined approach to e-project development.

The results aimed to be achieved by the end of the project by the SMEs performing the experiments are as follows:

- Increase productivity by approximately 20% due to the design of simpler, easy to understand systems and avoiding rework
- Reduce defect rates by 30%.
- Reduce their project overruns by about 15% due to avoiding schedule slips.
- Increase their competitiveness by producing higher quality products in shorter time and better prices

Baseline projects described by each one of the User companies will be used to measure the progress of the pilots.

The data collected during these experiments will be carefully analyzed. The lessons learnt from them, together with a final benchmarking study and a detailed description of the methods for project assessment and success measuring will be included in a Best Practice Toolkit. Based on the experience gained from these pilots and the analysis of the obstacles they met, the achieved results and the added values for the experimenting SMEs, a Best Practice toolkit will encompass:

- a description of the eXPERT approach and guidelines for its application by SMEs doing e-business or e-commerce;
- for each experiment: its settlement, aims, achievements, tools used, analysis of the results, lessons learnt;
- Final report about the pilot projects

The Best Practice Toolkit will be provided for public access to SMEs all around Europe interested in adopting and benefiting from the *e*XPERT approach. The dissemination activities aim at providing a wide range of SMEs abilities to meet the increasing flexibility, quality and time-to-market demands with highly skilled and communicative people.

3. eXPERT basics

The combination of XP and PSP principles will result in a novel software development approach which is suitable for e-business and e-commerce applications, where technology and needs change often and time-to-market is critical. In this section both XP and PSP approaches are presented shortly.

3.1. XP

Of all the lightweight methodologies, eXtreme Programming (XP) [1] tends to be best accepted by the e-project developers due to the simplicity of its rules and practices, its flexibility to changes in a project run, refactoring orientation and collaboration ideology. The twelve XP practices [3] are:

The Planning Process, sometimes called the Planning Game.

The XP planning process allows the XP "customer" to define the business value of desired features, and uses cost estimates provided by the programmers, to choose what needs to be done and what needs to be deferred. The effect of XP's planning process is that it is easy to steer the project to success.

Short Releases.

XP teams put a simple system into production early, and update it frequently on a very short cycle.

Metaphor.

XP teams use a common "system of names" and a common system description that guides development and communication.

Simple Design.

A program built with XP should be the simplest program that meets the current requirements. There is not much building "for the future". Instead, the focus is on providing business value. Of course it is necessary to ensure that you have a good design, and in XP this is brought about through "refactoring", discussed below.

Test first.

XP teams focus on validation of the software at all times. Programmers develop software by writing tests first, then software that fulfills the requirements reflected in the tests. Customers provide acceptance tests that enable them to be certain that the features they need are provided.

Refactoring.

XP teams improve the design of the system throughout the entire development. This is done by keeping the software clean: without duplication, with high communication, simple, yet complete.

Pair Programming.

XP programmers write all production code in pairs, two programmers working together at one machine. Pair programming has been shown by many experiments to produce better software at similar or lower cost than programmers working alone.

There are two roles in pair programming: the driver and the navigator. Both are active roles. The driver writes code. The driver needs to pay attention to what navigator is or is not saying. The navigator helps directly. He does not just admire work of the driver - pair programming is not “pair watching”. At least all the time navigator follows the activity of the driver. But this is not enough, he needs to be active: ask questions, raises objection, suggest alternatives – without distracting the driver. The navigator makes code review constantly – following coding standards, simplifying design (if code is not understandable for the second person in the pair, it is not simple), checking for errors, but giving the driver time to see his own mistakes and to correct them. If navigator see that driver get stuck, it is better to ask to drive.

Collective Ownership.

All the code belongs to all the programmers. This lets the team go at full speed, because when something needs changing, it can be changed without delay.

Continuous Integration.

XP teams integrate and build the software system multiple times per day. This keeps all the programmers on the same page, and enables very rapid progress. Perhaps surprisingly, integrating more frequently tends to eliminate integration problems that plague teams who integrate less often.

40-hour Week.

The idea of this practice is that long period of intensive work kill performance. It is not so important to have exact number of hours but to have in mind that principle. Tired programmers make more mistakes. XP teams do not work excessive overtime, keeping themselves fresh, healthy, and effective. Overtime is not the answer of the project schedule delay problem. It often is symptom of other problem (if you try to solve basic problem you do not need to have overtime).

On-site Customer.

An XP project is steered by a dedicated individual who is empowered to determine requirements, set priorities, and answer questions as the programmers have them. The effect of being there is that communication improves, with less hard-copy documentation - often one of the most expensive parts of a software project.

Coding Standard.

For a team to work effectively in pairs, and to share ownership of all the code, all the programmers need to write the code in the same way, with rules that make sure the code communicates clearly. The goal of coding standards is not to have an exhaustive list of rules, but to provide guidelines that will make sure that your code communicates clearly. Without coding standards it is harder to apply other XP practices, e.g. refactoring and simple design, as well as switching pairs and go fast – because in order to change the system it is important to have clear communication of code

XP practices are intended to use with small teams (2-12 people) which is the case with most of the e-project development. Although the XP practices seem to be very simple they require strong discipline (both individual and team) in order to achieve good results.

There are a number of reports about XP usage which show that they have failed due to developers' reluctance or incapability to apply the practices in a disciplined and professional manner. The incapability aspects are mainly related to making wrong estimations of individual work and failing to create a correct plan of the tasks that have to be performed. In this respect complementing XP with the Personal Software Process (PSP) is a good way for resolving the problem.

3.2. PSP

The *Personal Software Process (PSP)* [2] is a self-improvement process designed to help software developers to control, manage and improve their way of work. It is a structured framework of forms, guidelines, and procedures for developing small and big software projects.

The PSP progression is shown in Fig. 1. The first level in PSP is PSPO: The baseline process. It is to establish a baseline that includes some basic measurements and a reporting

format. PSP0 is enhanced to PSP0.1 by adding coding standard, size measurement and the process improvement proposal. PSP1 adds planning steps to PSP0 which include test report and size and resource estimation. In PSP1.1 task and schedule planning are introduced. PSP2 is the Personal Quality Management Process and PSP3 is Cyclic Personal Process.

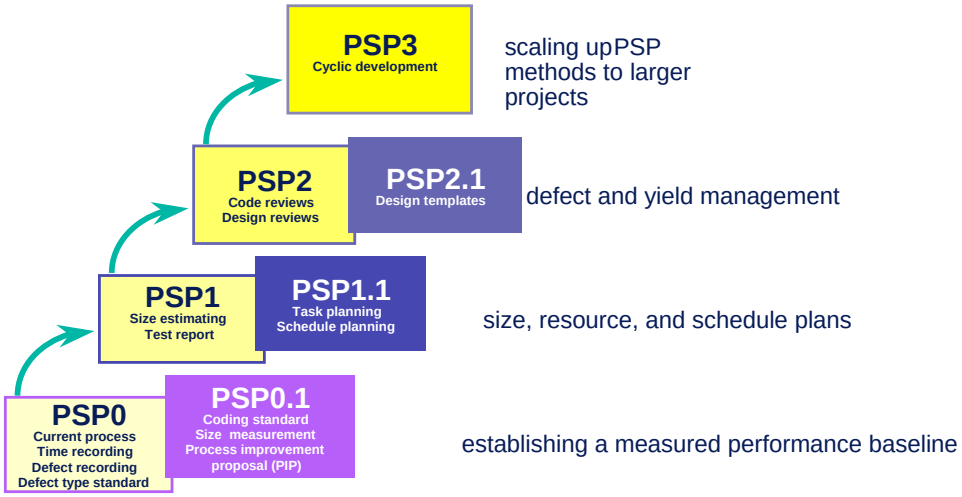

Fig.1 The PSP Evolution

Software quality and personal performance are of great importance for PSP. Software quality depends on software engineering quality and on software process. In order to do a software engineering job in the right way, engineers must plan their work before committing to or starting on a job, and they must use a defined process to plan the work. In order to have quality products, engineers must plan, measure, and track product quality, and they must focus on quality from the beginning of a job. To understand their personal performance, engineers must measure the time that they spend on each job step, the defects that they inject and remove, and the sizes of the products they produce. For example in PSP the *time recording log* is used for time measuring. In this log, engineers write the time they started working on a task, the time when they stopped the task, and any interruption time. The quality measures are related to recording data on every defect found in every phase. These data are kept in a *defect-recording log*.

4. eXPERT approach – combination of XP and PSP

The experience with XP application proves that it could be used as a base for solving the problems of small teams developing projects with often changing requirements and high quality demands. Therefore it is better to base the eXPERT approach activities on XP practices.

There are a number of reports about XP experiments showing that engineers usually have problems with correct estimation of time and cost. We find that complementing XP with the Personal Software Process (PSP) is a good way for resolving the problem.

The PSP helps developers as individuals to understand and improve their personal performance. It educates them to estimate and plan their work and to do this before committing to start doing the job. PSP, similar to XP, focuses on building quality products and tracking quality from the initial development phase, instead of checking it right before delivery. A complementary feature of PSP to XP is that PSP provides scripts that support each engineering activity and facilitate its correct completion. Another PSP feature that pairs the XP objectives is that it assists the software engineers in improving their performance.

Some of the difficulties in combining XP and PSP result from the fact that in relation to documentation they seem as two extremes. From one side XP does not assume creation of useless documents, and from other side PSP requires every activity to be documented. This imposes the usage of not all PSP principles, but only those, which are most important for reaching the goal and with additional modifications. Those modified principles could be used mainly for measuring defects and effectiveness of activities in order the problems to be identified in time and to be eliminated in the future.

Some PSP patterns could be used in data gathering but adapted in a way appropriate for XP application, in particular to have in mind two important features - pair programming and that processes of code design, testing and writing are strongly related and even running in parallel.

5. Conclusion

The paper is first in series of articles presenting eXPERT approach. It is a light method for software development applicable to small teams developing e-projects characterised with often changing requirements, tight schedules, and high quality demands. The underlying software methodologies XP and PSP are presented in this paper. Also the main aims of eXPERT project are described. The characteristics and process-based architecture of the eXPERT approach will be presented in next papers.

Acknowledgement

This work was the result of the IST-2001-34488 project, called “eXPERT“ that was funded by EC 5 Framework Programme.

References

[1] K. Beck. “The XP Series: Extreme Programming Explained”, Addison Wesley, 2000.
[2] W. Humphrey. “A Discipline for Software Engineering”, Addison Wesley, 1995.
[3] http://www.extremeprogramming.org