A framework for partial implementation of PSP in Extreme programming

Nazir Iqbal, Mahmood ul Hassan, Abdel Rehman Osman, Mubashir Ahmad
Department of Computing & Technology IQRA University-IU Islamabad, Pakistan.
Computer Department, College of Science & Arts, Al Jouf University, Tabarjal, Saudi Arabia

Abstract

As the developed software has tendency of complexity and growing size, still major portion of it is developed by an individual and self-directed programmer. To enhance the skills of individual developer and team capabilities is the key feature towards quality and productivity.
This research gives an idea of the modification of extreme programming by inserting in Personal software process, which is easy to follow and keeps the software development process lighten. This new model will establish best development practices from XP. For quality and better project planning PSP will support the individual. Strategies, phases and twelve core practices of this model are explained in this paper.

I. Introduction

Different approaches to software have been introduced from last 25 years; today few of them have been survived. In the beginning every approach was effectively used, but with the passage of time a lot of deficiencies found in those models.
A new method called "Agile" was introduced to overcome the problems in traditional methods. They are more adoptive as compared to traditional methods. There are numerous methodologies under the umbrella of "Agile" but the most popular type of methodology is Extreme Programming. XP is a set of principles and standards to develop high quality software speedily that offer maximum advantage to the customer.
Extreme programming is suitable where customers are not sure and change their mind frequently. XP uses iterative approach to develop software. To meet changing requirements are intended to deliver working software quickly and evolve this quickly.
Developers in small scale industry are facing a lot of problems to deliver a quality product well on time. Majority of the agile methods focuses on team. These methods are needed to be customized to enhance the individual performance.
PSP is the solution to improve the performance of the organization by improving individual performance, as the performance of individual collectively reflects at the organizational level, however to learn PSP and implement it thoroughly it needs a lot of efforts, training and documentation, so the actual implementation of PSP in small scale industry is very tough.
The approach in this research is based on combination of PSP and XP. The PSP helps people to realize and enhance their individual performance. PSP train individuals in estimation and planning of their work. Like XP, PSP track quality and from the very initial development phase, focus on building the quality product.
Personal software process provides personal control, with the help of PSP a developer can produce more reliable plan, and work can be managed properly. Personal software process helps the developer to estimate and measure their work.

II. The Practices

This improved model is a software development approach designed for small scale software industry. Following are **12** core practices.

Practices	**Improved Model**
The Planning Game	To divide the task into small module user stories is a good practice, here effort on each story is estimated and then sorted according to the priority.
Small Releases	The team releases the system to the customer in iteration. Important and Small unit of functionality are often release early. The feedback of this release is critical for the system development.
Continuous Integration	Daily all the changes are integrated. Tests are after and before the integration.
Simple Design	It is always easy to work on simple design than a complex one. Requirements are always change, so just only do to fulfill today's requirement.

Testing	To verify the software Tests are written, then software is developed to pass those tests. Two types of tests in this improved model 1. The developers write test first to check their functionality, each is for class or small module. 2. Acceptance Testing is for the whole system, when acceptance is pass for any story that is considered complete
Refactoring	Keep your code simple, understandable and easy to modify. It saves time and increases quality and also produce an organize system.
Coding standards	A set of practices which works as a model to be followed.
Process Improvement Proposal	Improvement ideas and to record process problems To have improvement ideas in priority order Priority of improvement plans Any other important observation
Time Recording	For errors Review the completed Time Recording log.
Size Measurement	The size of the completed program is measured. The size of new, deleted and reused code is measured. Use size template to determine to total program size
Defect Type Standard	Team member responsible for requirement gathering
Defect Recording	To ensure that all of the defects found in each phase were recorded. Use the Project Plan Summary to verify any omitted defects

III. Strategies of the model

Incremental change: Change in increment is the main strength of this model is, as change continuously occurs. so small and incremental change is suggested.

Stand up Meetings: For reporting problems daily a short meeting is held. The members then try to find the solution to those problems.

Tracking progress: The progress of the team members is tracked with the help of other team members.

Less Documents: This improved model uses few plan and recording logs to document.

Small investment: This model is mainly for small scale software industry, the strategy is to start with few developers, later if needed more developers are inducted.

IV. Phases of the model

The life cycle of this improved model consist of five phases: exploration, planning, personal planning, iteration to release, product ionizing.

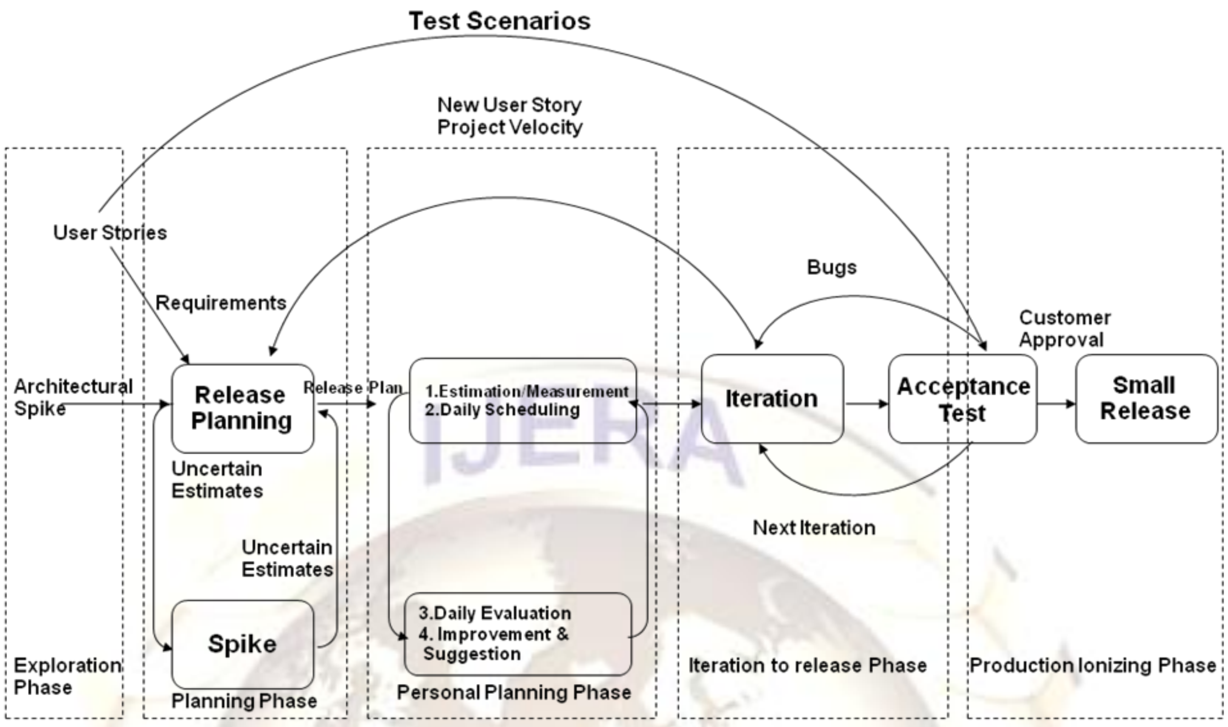
In the **exploration phase** the requirement are gathered. It is one of the challenging tasks for software developers. Requirements are gathered using story cards. In the mean time the project member familiarize themselves with tools, practices and technology that will be used in the project, by building a prototype of a system, the architectural possibilities of the system are explored. This phase takes from few weeks to months.

The main problem in this phase is the incompleteness and unclear gathering of requirements, which causes majority of the software failure

The **Planning phase** is about an agreement for the first small release and stories priority order. Here the programmer estimate time for each story and also schedule is decided. Release planning has a set of rules which permits everyone involved to make their own decisions. It is important for both technical and business people to make their decisions.

The developer on the basis of requirement priority choose a set of tasks, it may comprise of smaller tasks. The developer estimate and using his approach the cost for each task is calculated. The planning phase takes few days, while time for the first release is up to two months.

The **personal planning phase** is about estimation and measurement of the individual, here the programmer measure and estimates the size of the job. He estimates the effort required for each task and produces a schedule for himself according to the task order. To keep his work systematic the developer follow code standards and use process improvement proposal, Time and Defect recording log. In this phase the developer daily evaluates himself at the day end; with personal planning developer can provide regular updates to his mangers and customers.



A framework for partial implementation of PSP in Extreme programming

The **iteration to release** phase is of several iteration of the system, before first release. The schedule is made for a number of iterations, while each iteration is of 2 to 4 weeks long. Customer is responsible for the selection of each story. For the whole system architecture is created with the help of first iteration. At the end of every iteration functional test is run. The system is ready for production at the end of last iteration.

The **product ionizing Phase** needs some extra testing and performance of the system before it is released to customer. Decision has to be made and new changes may still be found in this phase. From 3 to 1 week the iteration need to be quick enough during this phase. For later documentation the postponed ideas are documented.

V Model at Work

The target of our model is individual and small scale software industry, although it is not formally applied in software industry, however it has been tested and applied by individual developer and has found satisfactory for the development. However a qualitative survey was conducted to realize the use of this model among software engineers. As software professionals are typically busy so few basic questions were asked. The questions are:

Q1: Was customer satisfied after usage of this model?

Q2: Does model enhance the productivity?

Q3: with practices programmer skills' get better.

Q4: Does this model improves software development cycle time?

Q5: Does the management visibility improve with practices?

100% of the professional were of positive opinion and they reiterate that to use this new model in their future suitable projects.

VI Conclusion

This paper suggests a new and improved software development model for small scale software industry. This model based on 12 core principle i.e. each six from extreme programming and personal software process. The model approach attempt between the light and heavy methodology to improves the software developer productivity and quality. This improved inducts the process in XP which targets individual performance, however it keeps the major practices of XP so that can be fit for small development team.

Our model consist of five (5) phases, they are exploration, planning, personal planning, iteration to release and product ionizing,

We do not recommend some of the practices like pair programming and on-site customer for small development team. As some of the team members do not feel comfort while working in fairs, similarly full time presence of customer and having informal request of changes makes the project costly and behind the schedule.

In order to find how the model works in reality and its effects on small software organization, one of the most exciting aspects is try to implement the model in an organization and conduct a case study. To validate and extend this model much work to be done.

References

1. "Can Extreme Programming be used by a Lone Programmer?" , Edward Akpata and Karel Riha (2004)
2. " Application and Evaluation of the Personal Software Process", by Hamdy K.Elminir, ,Eman A. Khereba, Mohamed Abu Elsoud, Ibrahim El-Hennawy,
3. PSP : "A self-improvement process for software engineers 2005"
4. W.S. Humphrey, "Introduction to Personal Software Process" , Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pa..
5. "Ravikant Agarwal , and David Umphress , "Extreme Programming for a Single Person Team?"
6. "A comparison between Agile and Traditional Software Development Methodologies", by M .A .Awad
7. " Personal Extreme Programming – An Agile Process for Autonomous Developers", by Yani Dzhurov, Iva Krasteva, and Sylvia Ilieva,
8. "Toward a Framework for Evaluating Extreme Programming", by Laurie Williams1, William Krebs2, Lucas Layman1, Annie I. Antón1, Pekka Abrahamsson3