ISSN: 2226-7522(Print) and 2305-3327 (Online)
Science, Technology and Arts Research Journal
April-June 2012, 1(2):59-64
www.starjournal.org

Review Article

Amalgamation of Personal Software Process in Software Development Practice

Abdul Kadir Khan

College of Computing and Informatics, Haramaya University, Haramaya, Ethiopia

Abstract

Today, concern for quality has become an international movement. Even though most industrial organizations have now adopted modern quality principles, the software community has continued to rely on testing as the principal quality management method. Different decades have different trends in software engineering. The Personal Software Process (PSP) is an evolutionary series of personal software engineering techniques that an engineer learns and practices. A software process is nothing without the individual programmer. PSP a data driven process customized to teaching individuals about their programming styles, helping software engineers further develop their skills in developing quality software. Apart from discussing about PSP as a framework of techniques to help engineers and their organizations to improve their performance while simultaneously increasing product quality, in this paper, the Personal Software Process definition, principles, design, advantages and opportunities are explained focusing on the incorporation of PSP concepts in software development practice.

Article Information

Article History:

Received : 30-04-2012
Revised : 24-06-2012
Accepted : 26-06-2012

Keywords:

Personal Software Process
Software Development
Engineers
PSP Concepts

***Corresponding Author:**

Abdul Kadir Khan

E-mail: qadirforu@gmail.com

INTRODUCTION

Modern quality principles have now been adopted by most of the industrial organizations, the software community has continued to rely on testing as the principal quality management method. Researchers and industrial leaders began to realize that software process, plans and methodologies for producing software, could help to produce accurate project deadlines, help keep software projects on budget and teach programmers to be more productive and knowledgeable of their own programming styles. For software, the first major step in the direction pioneered by Deming and Juran was taken by Michael Fagan when in 1976 he introduced software inspections (Fagan, 1976 and 1986). By using inspections, organizations have substantially improved software quality. Another significant step in software quality improvement was taken with the initial introduction of the Capability Maturity Model (CMM) for software in 1987 (Humphrey, 1989; Paulk, 1995). The CMM's principal focus was on the management system and the support and assistance provided to the development engineers. The CMM has had a substantial positive effect on the performance of software organizations. A further significant step in software quality improvement was taken with the Personal Software Process (PSP) (Humphrey, 1995).

In recent years there has been a great deal of emphasis on the assessment and improvement of the software development process. Engineering is a set of disciplines seeking solutions for complicated problems and systems that could not be done by individuals. The aim of engineering is to repetitively produce complicated artifacts in an efficient way. As we know that the term *software engineering* first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at the time. Software professionals have been arguing the term "software engineering" and its implication for three decades since Frits Bauer invented it in 1968.

The Personal Software process (PSP) was created by Watts Humphrey to address the need for individual software engineers to acquire a disciplined and effective approach to writing programs. The philosophy behind the PSP is that an organization's ability to build large-scale software systems dependent upon the ability of its individual software engineers to develop high quality small scale programs in a disciplined, effective manner. The PSP is designed to help engineers organize and plan their work, track their performance, manage software defects, and analyze and improve their personal process. The Personal Software Process (PSP) has been taught at a number of universities with impressive results. It is also of interest to industry as a means for training their software engineers. The PSP is intended to improve the personal practices of software engineers through the evolutionary introduction of good software engineering practices. This paper focuses on the amalgamation of PSP in software development practice.

Definition of PSP

"The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines" (Fritz Bauer, 1992).

Software engineering process is defined as the system of all tasks and the supporting tools, standards, methods, and practices involved in the production and evolution of a software product throughout the software life cycle. A process, in its most basic form, is a sequence of steps required to do a job. A software process defines methods which allow the programmer separate routine tasks from complex tasks and establish the criteria for starting and finishing each process step making the efficient use of his time. Therefore process is a term used to describe the people, methods, and tools used to produce software products. Hence Software is intangible and not subject to the same physical constraints as hardware and many manufacturing products, defining the software process can be difficult therefore process driven software development implies that organizational process is adapted to meet project and product quality goals.

The Personal Software Process (PSP) is a structured software development process that is intended to help software engineers understand and improve their performance, by using a "disciplined, data driven procedure". The PSP was created by Watts Humphrey to apply the underlying principles of the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to the software development practices of a single developer. The PSP is similar to the Capability Maturity Model (CMM), except that it focuses on the personal process. It claims to give software engineers the process skills necessary to work on a Team Software Process (TSP) team. The PSP concentrates on the work practices of the individual engineers. The principle behind the PSP is to produce quality software systems; every engineer who works on the system must do quality work. This means that almost everyone associated with software development must know how to do disciplined engineering work. When engineers use the disciplined approach to software development included by PSP, they learn to become more competent engineers producing quality software products on schedule.

The PSP is designed to help software professionals consistently use sound engineering practices. It shows them how to plan and track their work, use a defined and measured process, establish measurable goals, and track performance against these goals. The PSP shows engineers how to manage quality from the beginning of the job, how to analyze the results of each job, and how to use the results to improve the process for the next project.

The Principles of the PSP

Right way and the right approach is to be followed to do a software engineering job, engineers must plan their work before committing to or starting on a job, and they must use a defined process to plan the work. To understand their personal performance, they 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. To consistently produce quality products, engineers must plan, measure, and track product quality, and they must focus on quality from the beginning of a job. Finally, they must analyze the results of each job and use these findings to improve their personal processes. Software design is characterized as a set of practices and implementation techniques that allow the construction of marketable software systems that provide form and function satisfying to users.

The PSP Design is based on the Following Planning and Quality Principles:

- Every engineer is different; to be most effective, engineers must plan their work and they must base their plans on their own personal data.

- To consistently improve their performance, engineers must personally use well defined and measured processes.
- To produce quality products, engineers must feel personally responsible for the quality of their products. Superior products are not produced by mistake; engineers must strive to do quality work.
- It costs less to find and fix defects earlier in a process than later.
- It is more efficient to prevent defects than to find and fix them.
- The right way is always the fastest and cheapest way to do a job.

The PSP Structure and Planning

Structure

The PSP is a personal process that can be adapted to suit the needs of the individual developer. It is not specific to any programming or design methodology. Software engineering methods can be considered to vary from predictive through adaptive. The PSP is a predictive methodology PSP training follows an evolutionary improvement approach: an engineer learning to integrate the PSP into his or her process begins at the first level - PSP0 (Introduces process discipline and measurement, estimating and planning) and progresses in process maturity to the final level PSP2.1 (Introduces quality management and design).

The structure of the PSP process starting with a requirements statement, the first step in the PSP process is planning. There is a planning script that guides this work and a plan summary for recording the planning data. While the engineers are following the script to do the work, they record their time and defect data on the time and defect logs. At the end of the job, during the postmortem phase (PM), they summarize the time and defect data from the logs, measure the program size, and enter these data in the plan summary form. When done, they deliver the finished product along with the completed plan summary form.

Process Discipline and Measurement PSP0, PSP0.1: The main purpose of PSP0, the baseline process, is to provide a general structural framework for writing first PSP programs and for gathering statistical data on this software. PSP0 has 3 phases: Planning, Development (design, coding, compile and test) and a Post mortem. A baseline is established of current process measuring: time spent on programming, faults injected/removed, size of a program. In a post mortem, the engineer ensures that all the data for the projects has been properly recorded and analyzed. Once enough statistical data is gathered, typically after a few programs have been written or after a few projects, programmers can analyze their own performance data to manage and improve their individual personal process (Figure 1).

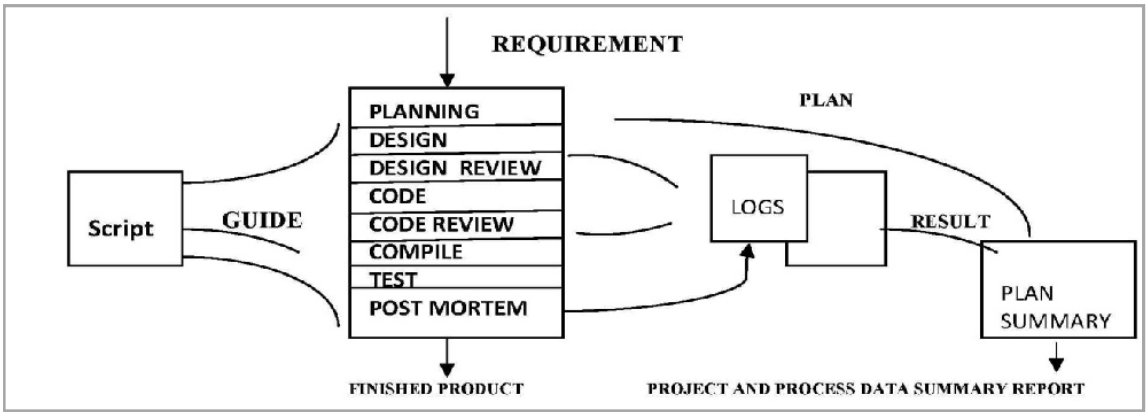

Figure 1: PSP Process Flow.

PSP0.1 advances the process by adding a coding standard, a size measurement and the development of a personal process improvement plan (PIP). In the PIP, the engineer records ideas for improving his own process.

Introduces Estimating and Planning PSP1, PSP1.1: When developing very large software systems, it is very important to have a plan to keep projects on schedule and programmers on task. Based upon the baseline data collected in PSP0 and PSP0.1, the engineer estimates the size of a new program, historical data is assumed

to be a good proxy because comparing future parts with historical parts can help the programmer to better understand the planned product and make more accurate judgments about its size and prepare a test report (PSP1). Data accumulated from previous projects is used to estimate the total time. Whereas estimation bias is one root cause of estimation errors, a programmer may always estimate a certain percentage than the actual amount of work performed. Secondly, estimation accuracy will fluctuate around a mean by a certain standard deviation therefore to gain more accurate size estimations the programmer should eliminate this estimation bias. Subsequently each new project will record the actual time spent. This information is used for task and schedule planning and estimation.

Introduces Quality Management and Design PSP2, PSP2.1: PSP2 adds two new phases: design review and code review. At this stage, the organization's engineers are able to use historical data to estimate a project's size, and management is able to plan accordingly. Now the focus is on defect prevention to decrease development time and produce higher quality software products (Defect prevention and removal are the focus at the PSP2). Engineers learn to evaluate and improve their process by measuring how long tasks take and the number of defects they inject and remove in each phase of development. Engineers construct and use checklists for design and code reviews. Reviewing designs before implementation allows the programmer to see and incorporate potential design improvements which can considerably save time and effort. The main goal of a code review is to be sure all of the details are correct. The objectives of these different types of reviews are the same, however to discover and to fix as many defects as possible PSP2.1 introduces design specification and analysis techniques. (PSP3 is a legacy level that has been superseded by TSP). The Team Software Process (TSP) was developed to address the commitment, control, quality and teamwork problems faced by most software development teams. By working together on a TSP, the programmers are able to give management and the customer a good idea of how much effort would be involved in producing and develop the desired software product.

Planning

The PSP uses the Proxy Based Estimation (PROBE) method to improve a developer's estimating skills for more accurate project planning. Logging time, defect, and size data is an essential part of planning and tracking PSP projects, as historical data is used to improve estimating accuracy. The PSP also uses statistical techniques, such as correlation, linear regression, and standard deviation, to translate data into useful information for improving estimating, planning and quality. These statistical formulas are calculated by the PSP tool. As shown in figure 2, planning process contains following steps:

Requirements: Engineers plan by defining the work that needs to be done in detail manner. If all they have is a one sentence requirements statement, then that statement must be the basis for the plan.

Conceptual Design: Since the planning phase is too early to produce a complete product design, engineers produce what is called a conceptual design. Later, during the design phase, the engineers examine design alternatives and produce a complete product design. Therefore to make an estimate and a plan, engineers first define how the product is to be designed and built.

Estimate Product Size and Resources: The correlation of program size with development time is good for engineering teams and organizations but for individual engineers, the correlation is generally quite high. Therefore, the PSP starts with engineers estimating the sizes of the products they will personally develop. Then, based on their personal size and productivity data, the engineers estimate the time required to do the work.

In the PSP, these size and resource estimates are made with the PROBE method. PROBE uses proxies or objects as the basis for estimating the likely size of a product. With PROBE, engineers first determine the objects required to build the product described by the conceptual design. Then type and number of methods for each object is determined. They refer to historical data on the sizes of similar objects they have previously developed and use linear regression to determine the likely overall size of the finished product.

PSP Planning

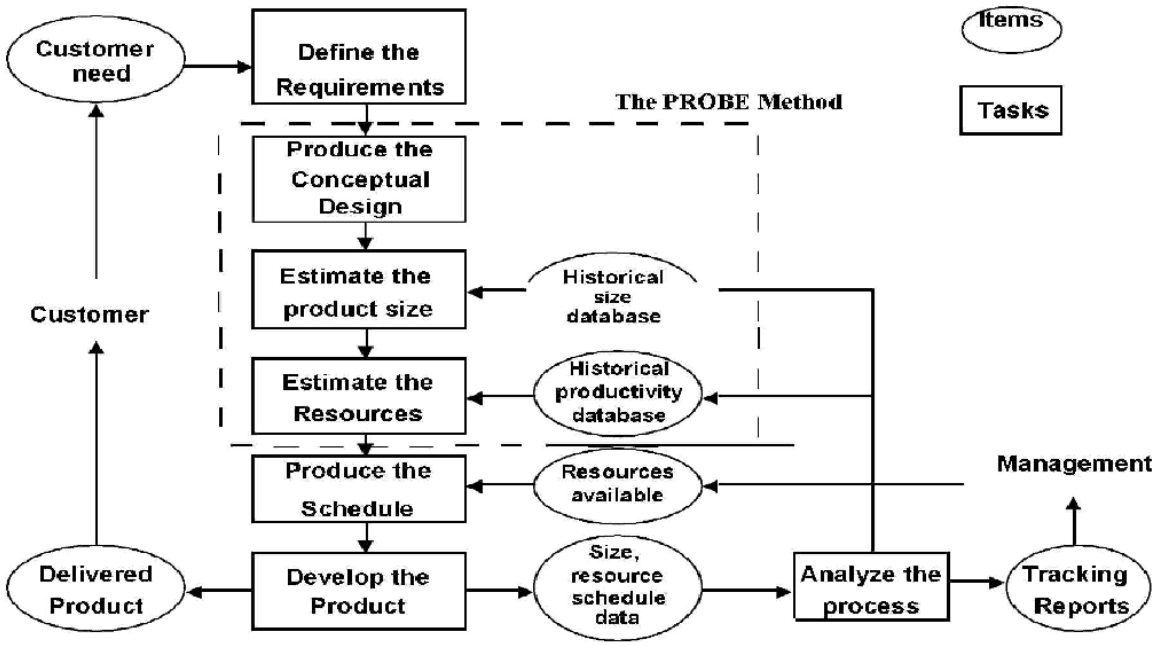

Figure 2: Project Planning Process.

Resource Estimating with PROBE: The PROBE method also uses linear regression to estimate development resources. Again, this estimate is based on estimated size versus actual effort data from at least three prior projects. The data must demonstrate a reasonable correlation between program size and development time. Once they have estimated the total time for the job, engineers use their historical data to estimate the time needed for each phase of the job.

Using these percentages as a guide, engineers allocate their estimated total development time to the planning, design, design review, code, code review, compile, unit test, and postmortem phases. When done, they have an estimate for the size of the program, the total development time, and the time required for each development phase.

Produce the Schedule: The Engineers spread the task time over the available scheduled hours to produce the planned time for completing each task once they know the time required for each process step, they estimate the time they will spend on the job each day or week. For larger projects, the PSP also introduces the earned value method for scheduling and tracking the work.

Develop the Product: In the step called develop the product, the engineers do the actual programming work and use the data from this process to make future plans.

Analyze the Process: After completing a job, the engineers do a postmortem analysis of the work. In the postmortem, they update the project plan summary with actual data, calculate any required quality or other performance data and review how well they performed against the plan. As a final planning step, the engineers update their historical size and productivity databases. At this time, they also examine any process improvement proposals (PIPs) and make process adjustments. They also review the defects found in compiling and testing, and update their personal review checklists to help them find and fix similar defects in the future.

PSP-Advantages, Challenges & Opportunities

Advantages

Context: The context is given by the definition of the PSP as described by Humphrey (1995). We may want to change the proposed PSP slightly, but basically the context is provided, and hence we do not have to define the context and

describe it very carefully to allow for others to understand our study from the context perspective.

Replication: The context also forms the basis for replication. Experiments and case studies can be conducted at several places using the PSP. Thus, the PSP may be one way to ease replication. The PSP provides a stable process and the process description is generally available.

Measures: This is also closely related to the PSP. Measures are collected as an integrated part of the PSP, and it is fairly easy to add measures of specific interest for an empirical study. Thus, the PSP provides a good starting point for collecting measures to use for hypothesis testing of model building.

Challenges

Scaling: The PSP implements activities performed in large scale project, hence scaling down, for example, planning and estimation to the individual level. The major challenge in using the PSP as context is the ability to scale the observation to other environments, and in particular to large scale software development. On the one hand, it is difficult to scale individual results to large project, on the other hand the PSP is supposed to act as a down-scaled project.

Validity: The validity of the observations and findings is crucial. The actual validity for different studies must be addressed separately, as the ability is highly dependent on the study and what we intend to generalize.

Opportunities

The PSP provides opportunities for experiential studies. We may study the use of different techniques and methods, or investigate the relationships between different attributes. The main limitation of using the PSP as a basis for experiential studies is that we cannot use it to study group activities. It is possible to experiment with using different reading techniques on an individual basis, but we are unable to study the use of inspections and group meetings.

CONCLUSIONS

In the future, software engineering groups will increasingly be required to deliver quality products on time and for their planned costs. Engineers will have to learn how to measure the quality of their work and how to use these measures to produce essentially defect free work. The aim of PSP is to provide software engineers with disciplined methods for improving personal software development processes. It is designed in such that it shall provide the disciplined practices software professionals will need in the future. It will help the software engineers to improve their estimating and planning skills and make commitments they can keep apart from managing the quality of their projects.

The PSP is designed for use with any programming language or design methodology and it can be used for most aspects of software work, including writing requirements, running tests, defining processes, and repairing defects. When engineers use the PSP, the recommended process goal is to produce zero defect products on schedule and within planned costs. When used with the Team Software Process (TSP), the PSP has been effective in helping engineers achieve this objective. The Personal Software Process, or PSP, is a flexible historical data-driven process tailored to teaching individuals about their own unique programming styles and even helps software engineers further develop their skills in writing quality software with few defects.

REFERENCES

Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall.

Fagan, M. (1976). Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal* 15:3.

Fagan, M. (1986). Advances in Software Inspections. IEEE Transactions on Software Engineering, SE-12:7.

Humphrey, W. (1989). Managing the Software Process. Reading, MA: Addison-Wesley.

Humphrey, W. (1998). The Software Quality Index, Software Quality Professional.

Humphrey, W. (2002). Winning with Software: An Executive Strategy. Addison-Wesley, Boston.

Humphrey, W. (2005). PSP: A Self-Improvement Process for Software Engineers. Addison-Wesley, Upper Saddle River, NJ.

Paulk, M., Curtis, B., Chrissis, M. B. (1995). Capability Maturity Model for Software, Version 1.1 Pittsburgh, Pa. Software Engineering Institute, Carnegie Mellon University.

Pressman, R. (1992). Software Engineering: A Practitioner's Approach. NewYork: McGraw-Hill.