A Study on the Application of Six Sigma Tools to PSP/TSP for Process Improvement

Youngkyu Park, Hyuncheol Park, Hojin Choi, Jongmoon Baik
Information and Communications University
119, Munji-ro, Yuseong-gu, Deajeon, 305-714, Korea
{youngkyupark,hcparker,hjchoi,jbaik }@icu.ac.kr

Abstract

The advent of software process models such as CMM/CMMI has helped software engineers understand principles and approaches of software process improvement. There, however, has been difficulty increasing productivity from applying these models since "how" is not within the scope of the CMM/CMMI. For this reason, SEI introduced PSP/TSP, but those still lack analysis tools and systematic process control techniques for analyzing metrics collected in PSP/TSP. Six Sigma, on the other hand, provides the quantitative analysis tools necessary to control process performance. Deploying PSP/TSP in conjunction with Six Sigma, therefore, can enable software engineers to analyze PSP data, and to systematically improve process performance. Continuing with this rationale, we map Six Sigma tools to each PSP process in order to show what Six Sigma techniques can be applied to PSP data and suggest Six Sigma practical use guideline to support process improvement activity at the individual and team level.

1. Introduction

Software development process is intangible and human-intensive. Several development processes such as Waterfall, Prototyping, Spiral, and Iterative were developed to make obscure software process visible and structured. Watts S. Humphrey in SEI (Software Engineering Institute) at CMU (Carnegie Mellon University) suggested a process maturity model framework arguing that measuring process and managing software process under control is a basis for continuous process improvement [1]. It has eventually evolved into CMM/CMMI (Capability Maturity Model/ Capability Maturity Model Integration) [2,3]. Those models helped software engineers understand principles and approaches of software process improvement. There, however, has been difficulty increasing productivity from applying the models to software organizations since "how" is not within the scope of the CMM/CMMI. In other words, software engineers do not know how to implement those due to the lack of procedures and means to improve their processes at organization level. To show individual developers and their teams how to apply CMM/CMMI principles at the individual and team level, PSP/TSP (Personal Software Process/ Team Software Process) were developed by SEI.

There are only a few analysis techniques provided by PSP/TSP. However, Six Sigma has many other available tools that have not been suggested or incorporated in PSP/TSP. Six Sigma statistical analysis and decision-making tools can be employed for data analysis, continuous improvement, and process control in PSP/TSP. PSP metrics framework provides the infrastructure that can be used as a basis for Six Sigma approach. Deploying PSP/TSP in conjunction with Six Sigma, therefore, can provide the quantitative analysis capabilities to identify high leverage activities, evaluate the effectiveness of process changes, quantify cost and benefits, and control process performance. In short, despite the fact that PSP/TSP and Six Sigma can be utilized independently of each other, there is a definite and significant synergy between them. Continuing with this rationale, we map available Six Sigma tools to each PSP process. This paper also describes the Six Sigma practical use guideline to support process improvement activity at the individual and team level.

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

The rest of this paper is organized as follows. In Section 2, a brief introduction to PSP, TSP, and Six Sigma is provided. In Section 3, we provide the mapping table that shows what kind of Six Sigma tools can be used in each PSP process and explain a specific example. In Section 4, we conclude the paper and give suggestions for future works.

2. Introduction to PSP, TSP, and Six Sigma

2.1. PSP

PSP helps individual developers improve their performance by bringing discipline to the way that they develop software. It clearly shows developers how to manage the quality of their products, how to make a sound plan, and how to make commitments. It also offers them the data to justify their plans. They can evaluate their work and suggest improvement direction by analyzing and reviewing development time, defects, and size data.

PSP consists of a series of seven personal processes that provides various scripts, measures, forms, templates, checklists, and standards. These personal processes are categorized as follows [4]:

- PSP0, PSP0.1 – Gather current data
- PSP1, PSP1.1 – Estimate using historical data
- PSP2, PSP2.1 – Manage quality
- PSP3 – Apply PSP2 cyclically

PSP0 provides a consistent fundamental for measuring progress and a defined foundation on which to improve [5]. PSP0.1 extends PSP0 by appending elements such as coding standard, size measurement, and PIP (Process Improvement Proposal) [4].

PSP1 focuses on personal project management. PSP1 adds software size estimating and test reporting to PSP. PSP1.1 appends task planning and schedule planning to PSP1.

PSP2 extends PSP1 by introducing review techniques such as code review and design review [5]. These review techniques help software engineers find defects as early as possible. PSP2.1 appends design templates to PSP2 so that software engineers establish design completeness criteria and examine design verification and consistency techniques.

PSP3 introduces methods for individuals to use when they are developing larger-scale software. In scaling the PSP2 up to larger projects, the approach is to subdivide the personal process of developing larger programs into PSP2-sized pieces [5].

2.2 TSP

The primary goal of TSP is to create a team environment for establishing and maintaining a self-directed team, and supporting disciplined individual work as a base of PSP framework [6]. Self-directed team means that the team manages itself, plans and tracks their work, manages the quality of their work, and works proactively to meet team goals.

TSP has two principal components: team-building and team-working [7]. Team-building is a process that defines roles for each team member and sets up teamwork through TSP launch and periodical re-launch. Team-working is a process that deals with engineering processes and practices utilized by the team.

TSP, in short, provides engineers and managers with a way that establishes and manages their team to produce the high quality software on schedule and budget.

2.3 Six Sigma

Six Sigma is a quality improvement approach to enhancing organization's performance by using statistical analytic techniques [8]. Six Sigma aims to eliminate the variability and defects which interfere with customer satisfaction and cost reduction. Six Sigma has been being embodied in the management strategy for quality improvement to quantitatively evaluate organization's processes and to reduce process variability [9].

Six Sigma is defined at three levels [10]:

- Philosophy: Being more profitable, Six Sigma can be used for improving customer satisfaction by reducing and eliminating defects.
- Metrics: As a metric, Six Sigma equals to 3.4 defects per million opportunities (DPMO). Additionally Six Sigma includes several metrics such as Defect rate (parts per million), Sigma Level, Defects per Unit (DPU), and Yield [11].
- Improvement Framework: Six Sigma owns various toolkits and structured problem solving roadmaps such as DMAIC (Define, Measure, Analyze, Improve, Control) and DMADV (Define, Measure, Analyze, Design, Verify).

Six Sigma, in other words, is defined as the scientific quality improvement approach that measures

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

the performance of the organization's processes and analyzes the cause of the defects by using Six Sigma roadmaps and toolkits, then eliminates the defects and pursues continuous, measurable, and controllable improvement of the organization's processes so as to accomplish Six Sigma level. By eliminating defects and process variability, Six Sigma achieves cost reduction and customer satisfaction.

3. Integrating PSP and Six Sigma

There are two approaches to process improvement. One is top-down approach which is a way to improve software process from organization to individual level. The other is bottom-up approach which is a way to improve from individual to organization level. According to bottom-up approach, Figure 1 shows that Six Sigma can support software process improvement activities in PSP/TSP. In addition, by utilizing data collected and analyzed in PSP/TSP using Six Sigma tools, software organizations can make use of these data to achieve process improvement at organizational level. Despite the fact that PSP provides data which can be analyzed by Six Sigma, there are few cases that software organizations actually apply Six Sigma to PSP to help improve software process at individual level by providing in-process feedback. Therefore, we provide the mapping table that shows what kind of Six Sigma tools can be used in each PSP process to help analyze metrics collected in PSP/TSP.

PSP helps developers plan, manage, and improve their works by showing them how to use defined process, trace time, record defects, estimate program size, make a plan, trace earned value, and so on. Six Sigma, on the other hand, provides PSP with various tools for detecting special causes of variation, evaluating the impact of process changes, and improving process performance.

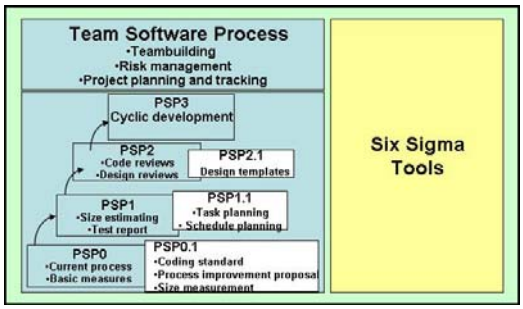

Figure 1. Relationship between PSP/TSP and Six Sigma

Before providing the mapping table, locating features of PSP that can be thought of as Six Sigma enabler can help understand relationships between PSP/TSP and Six Sigma. Key features of PSP that are pertinent to Six Sigma include:

- Software engineers are required to estimate the size of software products to be produced, and record actual size upon completion of the software products.
- Software engineers are required to record time for each development process activity.
- Software engineers are required to record and track defects, and hence they can predict and evaluate yield at each step in the process.

Software developers can follow prescribed methods, represented as levels PSP0 through PSP3 as shown in Figure 1. With each process, they are introduced to various advanced software engineering methods step by step. Once software developers get to know PSP, most of them end up working on teams that use the TSP. All of the PSP metrics and quality assurance activities are embedded into TSP. In addition, TSP establishes defined process foundation and generates useful data that can be analyzed by Six Sigma tools. In this way, Six Sigma provides ways to analyze collected data in PSP/TSP which ultimately lead to individual and team performance improvement, and hence provide a basis for applying Six Sigma at the project and organization level.

In the next subsection, we provide mapping table that maps Six Sigma tools to each PSP process and then we provide an example that shows Six Sigma tool usage in detail.

3.1 Mapping Six Sigma tools to PSP

Mapping Six Sigma tools to each PSP process helps software engineers understand how to use Six Sigma analysis techniques in conjunction with PSP data. As shown in Table 1, Six Sigma statistical analysis and decision-making tools are mapped to each PSP process. This table provides information on what kinds of Six Sigma tools can be used in which PSP process. It also describes the purpose of the tools at the most right column. As PSP process moves from lower to higher level, more Six Sigma tools can be applied because more data are collected.

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

Table 1. Mapping table between Six Sigma tools and PSP processes

Process	Phase	Six Sigma tools	Purpose
PSP0	Planning	Correlation analysis, regression analysis	To estimate development time of new program
	Development	Dotplot, pareto analysis	To analyze the percentage of defects by defect type
	Postmortem	Correlation analysis, regression analysis	To estimate size and development time of new program
PSP0.1	Planning	Correlation analysis, regression analysis	To estimate *Added and Modified* value for new program
	Development	Multiple regression analysis	To analyze *Added*, *Reused*, *Modified* value distribution of current program and to estimate LOC for new program
	Postmortem	The same tools as in postmortem phase in PSP0	
PSP1	Planning	PROBE(PROxy-Based Estimating)	To estimate new program size and development time
	Development	The same tools as development phase in PSP0.1	
	Postmortem	2-sample t-test	To determine whether estimations has been correct by comparing between *Plan Size/Hour* values and *To Date Size/Hour* values
…			

3.2 An example on application of Six Sigma tools to PSP

This section provides a 2-sample t-test example that can be used in postmortem phase in PSP1 process. It shows specific steps of applying Six Sigma tools to particular PSP process. One of the primary jobs in postmortem phase in PSP1 is to calculate *Plan*, *Actual* and *To Date Size/Hour* values in *Summary* section which is done by dividing *Added and Modified* in *Program Size* section by *Total* planned, actual, and to-date in *Time in Phase* section in PSP1 project plan summary form. By analyzing difference between *Plan* and *To Date* values on prior projects, estimation accuracy can be determined. For this example, we made use of Minitab tool to analyze the collected data [12].

Problem

PSP1 requires that individual software engineers enter *Plan*, *Actual*, and *To Date Size/Hour* values in PSP1 project plan summary form in postmortem phase. Once those values are calculated, 2-sample t-test can be applied which is used to check if two population means are equal. In this case, applying 2-sample t-test is appropriate since estimation accuracy can be determined by difference between average of *Plan Size/Hour* values and that of *To Date Size/Hour* values.

Data Collection

Plan LOC/Hour and *To Date LOC/Hour* values should be collected in order to determine estimation accuracy. The primary goals of the metric, *Size/Hour*, are to provide a convenient basis for comparing plans with historical performance and a fall-back in case the regression method does not produce a reasonable result. The data should answer the following questions.

- How many *LOC/Hour* did I plan?
- What has been my average *LOC/Hour* on prior projects?

Procedure

This procedure shows each step on how to apply 2-sample t-test to determine whether program size and effort have been correctly estimated in postmortem phase in PSP1.

[Step 1] Let's suppose that LOC (line-of-code) is used as size measure and also suppose that planned *Added and Modified* value is 169LOC, estimated development time is 380 minutes, *Added and Modified* value to date is 492LOC, and actual development time to date is 1663 minutes. *Plan* and *To Date LOC/Hour* values can be calculated as shown below.

Plan LOC/Hour value
60 * 169/380 = 26.7 LOC/Hour

To Date LOC/Hour value
60* 492/ 1663 = 17.8 LOC/Hour

[Step 2] Tabulate *Plan* and *To Date LOC/Hour* values from prior projects. Table 2 shows *Plan LOC/Hour* and *To Date LOC/Hour* values collected from 6 prior

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

projects. The values from Table 2 will be used for this example.

Table 2. *Plan LOC/Hour* values, *To Date LOC/Hour* values

Plan LOC/Hour values	*To Date LOC/Hour* values
16.4	12.3
20.2	14.3
23.4	16.2
26.7	17.8
30.3	20.4
35.2	26.7

With this configuration, we can test the hypothesis that the population means for two samples are equal using 2-sample t-test with 95% confidence level.

H_0 (Null Hypothesis) : $\mu_1 = \mu_2$
H_1 (Alternative Hypothesis) : $\mu_1 \neq \mu_2$

[Step 3] Input the data to Minitab.

[Step 4] Do normal plots to verify that the data do not deviate significantly from a normal distribution. If the data come from a normal distribution, the points will roughly follow the straight reference line. Conversely, if the data do not come from a normal distribution, the points will not follow the line. Since both P-values (0.966 for *Plan LOC/Hour* values, and 0.635 for *To Date LOC/Hour* values) are greater than 0.05 and the points follow the straight reference line, it is reasonable to assume that the data do not deviate substantially from a normal distribution. The assumption of normality is relatively satisfied. As an example, Figure 2 shows that normality test for *Plan LOC/Hour* values are satisfied.

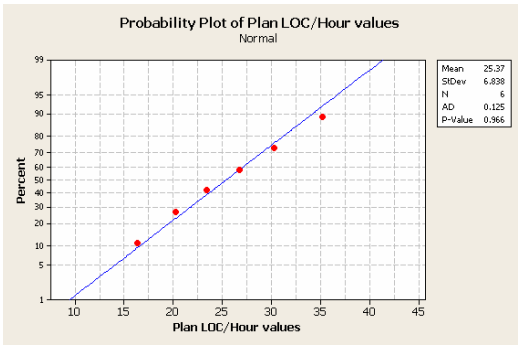

Figure 2. Probability plot of *Plan LOC/Hour* values

[Step 5] Evaluate the variances of the two distributions to see if they differ. Variance tests should be conducted since the calculations for the 2-sample t-test depend on whether the sample variances are the same or different.

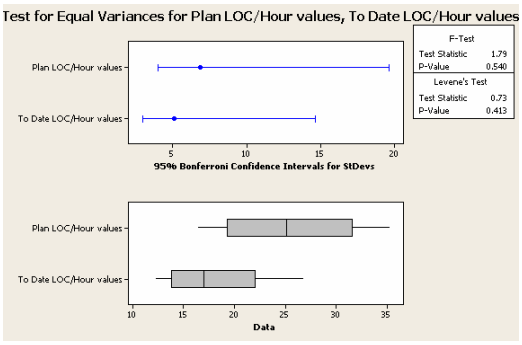

Figure 3. Variance tests for *Plan LOC/Hour* values, *To Date LOC/Hour* values

If the data are continuous and normally distributed, F-Test can be used. On the other hand, if the data are continuous, but not necessarily normally distributed, Levene's Test can be used. The present data are reasonably normal as checked in Step 4, so F-Test can be used. Null hypothesis that the variances are equal can not be rejected at the 0.05 α-level since P-value (0.54) is greater than 0.05 as shown in Figure 3. The result suggests that the variances are equal.

[Step 6] Conduct 2-sample t-test to determine whether estimations are correct. Because the variance test indicated that the population variances are equal, check *Assume equal variances* check box when conducting 2-sample t-test in Minitab. Create boxplot to help visualize the data. Test the hypothesis using summary results and box plot from 2-sample t-test.

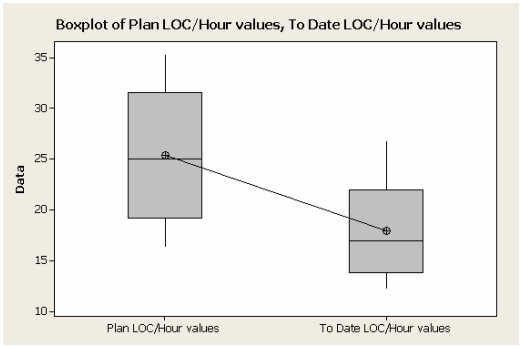

Figure 4. Boxplot of *Plan* LOC/Hour values, *To Date* LOC/Hour values

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

Figure 4 illustrate two facts. One is *Plan LOC/Hour* values appear to be larger than those of *To Date LOC/Hour*. The other is there is more variability in *Plan LOC/Hour* values than in *To Date LOC/Hour*.

Two-Sample T-Test and CI
Plan LOC/Hour values, *To Date LOC/Hour* values

	Plan LOC/Hour	*To Date LOC/Hour*
N	6	6
Mean	25.37	17.95
StDev	6.84	5.12
SE Mean	2.8	2.1

Difference = μ_u (Plan LOC/Hour values) - μ_u (To Date LOC/Hour values)
Estimate for difference: 7.41667
95% CI for difference: (-0.35172, 15.18505)
T-Test of difference = 0 (vs not =): T-Value = 2.13 **P-Value = 0.059** DF = 10

Figure 5. 2-sample t-test summary result

The result gives summary statistics for the two samples, the estimated difference in means, a 95% confidence interval for the difference, a t-test of the difference, the observed value of t, the P value and the degrees of freedom.

[Step 7] Interpret the result.

As shown in Figure 5, null hypothesis can not be rejected at the 0.05 α-level since P-value (0.059) is greater than 0.05. The conclusion is that estimations have been correct when comparing the planned values and actual values. If P value is lesser than 0.05, it means estimations have been incorrect. It also means that there have been problems in estimations. Therefore, recalibration or different estimation techniques can be used to resolve this incorrect estimation problem. Process performance can be improved and controlled with this procedure in PSP.

4. Conclusion and Future Works

Six Sigma is a measurement-driven approach to continuous process improvement that focuses on reduction of variation, consistency, and high product quality. Therefore, in terms of software process, applying Six Sigma which emphasizes on data measurement to PSP/TSP can accelerate software process improvement by identifying high leverage activities, quantifying cost and benefits, evaluating the effectiveness of process changes, and controlling process performance. Based on this, we provided the mapping table that shows Six Sigma tools which can be used in PSP/TSP. We also provided an example of 2-sample t-test to show how a Six Sigma tool can be utilized in PSP/TSP. However, performance increase at the individual and team level can be attained by identifying and analyzing issues using Six Sigma tools that could happen at team level, and that can provides a basis for applying Six Sigma at the project and organization level. Thus, we are currently doing research on how to improve individual and team performance from TSP angle using Six Sigma tools. For future works, we are going to extend the use of Six Sigma tools and methodologies to organization and business level to create more values and better customer satisfaction.

5. Acknowledgement

The research described in this paper was supported by University IT Research Center Project of the Ministry of Information and Communications, Korea.

6. References

[1] W. S. Humphrey, Managing the Software Process, Addison- Wesley, Reading, MA, 1989.
[2] Mark C. Paulk, B. Curtis, M. B. Chrissis and et al., "Capability Maturity model for Software", Software Engineering institute, CMU/SEI-91-TR-24, DTIC number ADA240603, August 1991.
[3] CMMI, "Capability Maturity Model Integration", Software Engineering Institute, Carnegie Mellon University, 2002.
[4] Watts S. Humphrey, "The Personal Software Process (PSP)", Software Engineering Institute, CMU/SEI-2000-TR-022, ESC-TR-2000-022, 2000
[5] Watts S. Humphrey, "The personal process in software engineering", October 1994, pp 69 – 77
[6] Noopur Davis, Julia Mullaney,"The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results", Software Engineering Institute, CMU/SEI-2003-TR-014, ESC-TR-2003-014, 2003
[7] Watts S. Humphrey, "The Team Software Process (TSP)", Software Engineering Institute, CMU/SEI-2000-TR-023, ESC-TR-2000-023, 2000
[8] Thomas Pyzdek, The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels, Revised and Expanded Edition, McGraw-Hill, 2003
[9] Pete, Larry Holpp, What is Six Sigma?, McGraw-Hill, 2002
[10] Jeannine Siviy, M. Lynn Penn, and Erin Harper, "Relationships Between CMMI® and Six Sigma", Software Engineering Institute, CMU/SEI-2005-TN-005, 2005
[11] David L. Hallowell, "Six Sigma Software Metrics, Part 3",http://software.isixsigma.com/library/content/c031015a.asp
[12] http://www.minitab.com/