IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.006, LIIR468
DOI: https://doi.org/10.52731/liir.v006.468

# Evaluating Instructor Role in a Personal Software Process Improvement Course Based on the Process Data

Shigeru Kusakabe [*], Masanobu Umeda [*],
Keiichi Katamine [*], Shunsuke Araki [*]

## Abstract

The Personal Software Process (PSP) is a well-designed personal software process effective for software engineers in understanding and improving their performance accompanied by training courses to establish and improve their software development processes. In this paper, we evaluate the importance of instructors' role in the PSP courses. The PSP for Engineers is one of the PSP training courses offered by the SEI, where participants learn the knowledge and skills for developing high-quality software through lectures and exercises led by instructors. The lecture materials for the course have been available under a Creative Commons license from October 2018, and participants can self-learn PSP without the guidance of instructors. However, for example, it is not always easy to collect accurate and precise process data, the basis for improvement. We expect instructor guidance plays a major role in effectively improving software processes. In this paper, we analyze the importance of PSP instructors' role based on the students' process data collected in the PSP for Engineers course at our graduate school over years.

*Keywords:* Software Process, Process Improvement, Training, Instructor.

## 1 Introduction

The Personal Software Process (PSP) [1][2] is a well-designed personal software process effective for software engineers in understanding and improving their performance, developed by Watts S. Humphrey of the Software Engineering Institute (SEI) at Carnegie Mellon University in the United States. The PSP for Engineers course offered by SEI is one of such PSP training courses, mainly aimed at experienced workers in industries. Participants can learn the knowledge and skills necessary for high-quality software development through lectures, program development exercises, and self-analysis under the guidance of PSP instructors.

Previously, lecture materials for the course were available under a partner agreement with SEI, while some were also available on the website [3] under certain conditions. However, since October 2018, the following PSP materials have been available free of charge from the SEI Digital Library [4] under a Creative Commons license [5].

- Introduction to Personal Software Process (PSP)
- Personal Software Process (PSP) for Engineers V3.2.1
- Personal Software Process (PSP) for Engineers V4

[*] Kyushu Institute of Technology, Iizuka, Japan.

- Personal Software Process (PSP) Fundamentals
- Personal Software Process (PSP) Advanced
- Personal Software Process (PSP) Instructor Training

Therefore, in a sense, we can say that learning environment settings have been established for self-study of PSP by combining lecture materials explained above with commercially available books, and so on. However, it is not necessarily easy to learn self-improvement skills by simply performing exercises based on these materials. In fact, some people have questioned the effectiveness of PSP because they believe that self-learning does not improve the skills[6].

# 2 PSP for Engineers Course Overview

## 2.1 PSP for Engineers Course Structure

The quality of software is determined by the minimum quality of the components that make it up, and the quality of each component is determined by the individuals who developed it and the quality of the process they used. Therefore, improving individual processes is essential to improving software quality.

PSP is a self-improvement process for software engineers. Figure 1 illustrates the evolution of the PSP process and its relationship to the Team Software Process (TSP) [7][8]. In PSP0 and PSP0.1, participants learn the importance of defect recording, time recording, size measurement, improvement proposals, and the discipline required to ensure their implementation. In PSP1 and PSP1.1, participants learn how to identify the components required to realize requirements, estimate size and time based on this, and plan and track progress. In PSP2 and PSP2.1, participants learn how to plan quality, review design and code, and design and verify using design templates.

The PSP for Engineers course is one of the PSP training courses provided by SEI, and is aimed primarily at people with work experience in companies. It consists of two courses (5 days each): PSP-Planning (PSP0-PSP1.1) and the subsequent PSP-Quality (PSP2-PSP2.1). After a half-day lecture, participants in each course develop small software projects of about a few hundred lines. In addition, on the final day after the lecture, participants are required to submit a self-analysis report. Participants will quantitatively analyze the process data related to the exercises they have completed, and based on that, propose improvements to their own software process and apply them to the next exercise. By continually repeating this process, participants aim to acquire the knowledge and skills necessary for high-quality software development.

## 2.2 PSP for Engineers Course Flow

In the PSP course, the instructors give lectures on the assumption that the students have learned in advance from the corresponding book chapters and assign them exercises of program development. Before starting activities in the development phases, students make their development plan according to the process (PSP0 to PSP2.1) they are learning and ask the PSP instructor to review the plan. The PSP instructor checks the contents of the plan and provides guidance and advice as necessary. If there are or become no defects in the plan, the

students are allowed to start developing the assignment program based on that plan. When development is complete, the students make their report package containing several forms in addition to the source file and test results. In the postmortem phase, students make their self-review and ask the PSP instructor to review the exercise report package. The PSP instructor checks the contents of the exercise report package and provides guidance and advice as necessary. The students revise the exercise report as necessary and request follow-up reviews until the report package becomes flawless.

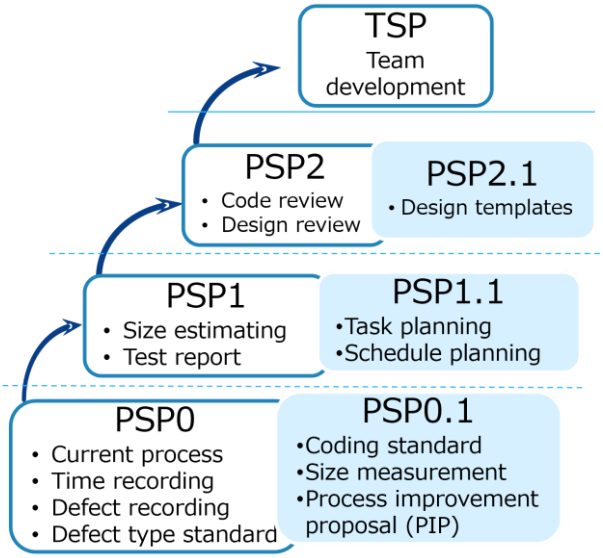



Figure 1: Evolution of PSP and its relationship to TSP

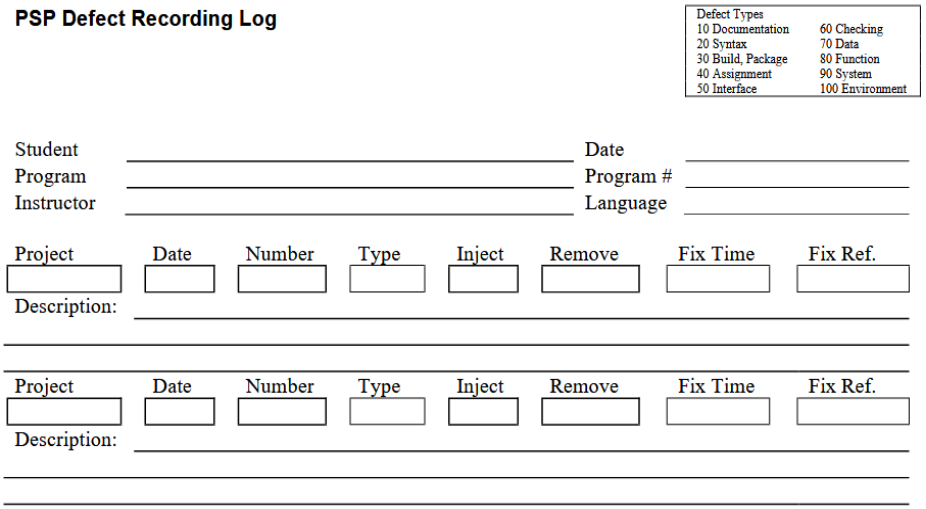
Students record process data such as defects, time, and size throughout the series of activities from the time they start the exercise assignment until their exercise report is accepted. Figure 3 and Figure 2 show PSP forms for time and defect recording. The process data recorded during exercises is basic data for proposing improvements to their software process. If the data contains imprecise or inaccurate data, such as incorrect selection of defect types as shown in Table 1 or missing time records, inappropriate process data may hinder improvements to their software process. For this reason, when reviewing exercise reports, PSP instructors may need to provide repeated guidance and advice until it becomes clear that their process data such as defects, time, size data becomes effective for improving their process.

**PSP Time Recording Log**

Student _____ Date _____
Program _____ Program # _____
Instructor _____ Language _____

| Project | Phase | Start Date and Time | Int. Time | Stop Date and Time | Delta Time | Comments |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

Figure 2: A PSP form of time recording log


PSP Defect Recording Log

Defect Types
10 Documentation 60 Checking
20 Syntax 70 Data
30 Build, Package 80 Function
40 Assignment 90 System
50 Interface 100 Environment

Student _____ Date _____
Program _____ Program # _____
Instructor _____ Language _____

Project Date Number Type Inject Remove Fix Time Fix Ref.
Description:

Project Date Number Type Inject Remove Fix Time Fix Ref.
Description:

Figure 3: A PSP form of defect recording log

In this way, PSP instructors are expected not only to convey knowledge about PSP to students through lectures, but also to encourage efficient and effective improvements to software processes by providing guidance and advice on the results of exercises.

Table 1: PSP defect types

| Defect type | Number and description |
|---|---|
| Documentation | 10: Comments, messages |
| Syntax | 20: Spelling, punctuation, typos, instruction formats |
| Build, Package | 30: Change management, library, version control |
| Assignment | 40: Declaration, duplicate names, scope, limits |
| Interface | 50: Procedure calls and references, I/O, user formats |
| Checking | 60: Error messages, inadequate checks |
| Data | 70: Structure, content |
| Function | 80 Logic, pointers, loops, recursion, computation, function defects |
| System | 90 Configuration, timing, memory |
| Environment | 100 Design, compile, test, or other support system problems |

# 3 Results of the Course at Our Graduate School

## 3.1 Overview of PSP course at Our Graduate School

Our graduate school worked with SEI since 2007 to incorporate PSP and TSP into graduate school education and to foster advanced information and communication engineers [9]. This cooperation continued until the SEI license system of PSP ended. This graduate school education

course consists of a PSP course based on the PSP for Engineers and a TSP course based on the Introductory TSP (TSPi) designed for academic organizations.

The PSP course consists of two exercise courses corresponding to PSP-Planning and PSP-Quality, and has the same content as the PSP for Engineers course. However, in order to ensure that students have enough time for exercises even if they take other courses at the same term, the time range for one day contents is extended to one week, and the entire course completes in one semester.

Until the current SEI license system was abolished, the PSP course was run by faculty members who had SEI-certified PSP instructor qualifications based on the SEI license. As a result, graduates of this course were awarded a PSP for Engineers course completion certificate, just like courses offered by SEI.

## 3.2 Software Process Improvement Results

We have been continuously evaluating the effectiveness of our training course[10]. In this section, we discuss how the software process improved by using the PSP process data of the student who took and completed PSP-Planning and PSP-Quality from 2007 to 2021.

### *3.2.1 Estimation error of size and time*

Figure 4 shows the quartile trends of the size estimation error. The horizontal axis shows the exercise number, and the vertical axis shows the size estimation error. From this figure, we can see that there is a tendency for underestimation at first, but as the course progresses, the error becomes smaller, and the estimation becomes more balanced between underestimation and overestimation.

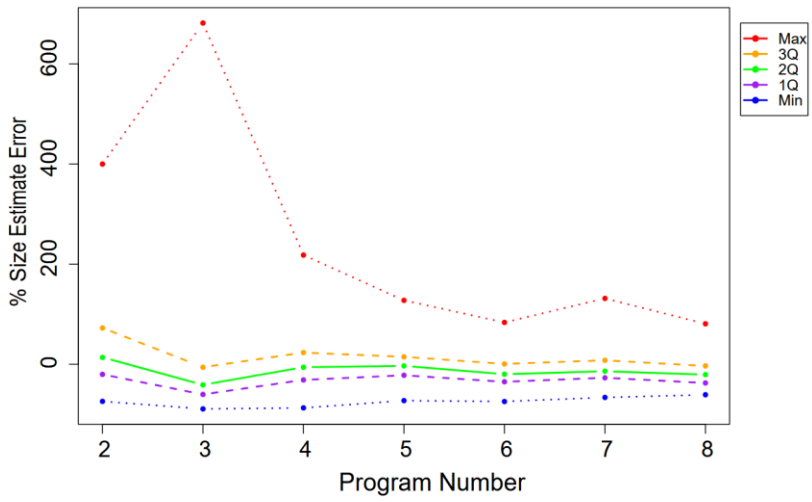



Figure 4: Trends in size estimation error.

Figure 5 shows the quartile trends of the time estimation error. The trend seems unstable. We think this is because as the more the course progresses, the more new process elements the student must master. Although it is better to have a small estimation error, when estimating an entire system consisting of multiple parts, it is more important to achieve a balance between underes-

timation and overestimation so that the estimation errors of the parts can cancel each other out.

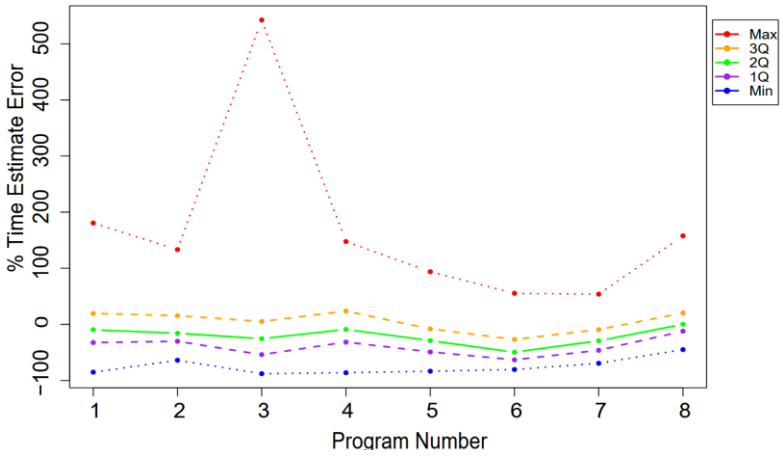



Figure 5: Trends in time estimation error

### 3.2.2 *Defect Density*

Figure 6 and Figure 7 show the quartile trends of the density of defects found and removed during compilation and testing in KLOC (number of defects per 1000 lines). The horizontal axis shows the exercise number, and the vertical axis shows the defect density. From these figures, we can see that the compilation defect density steadily decreased as the course progressed.

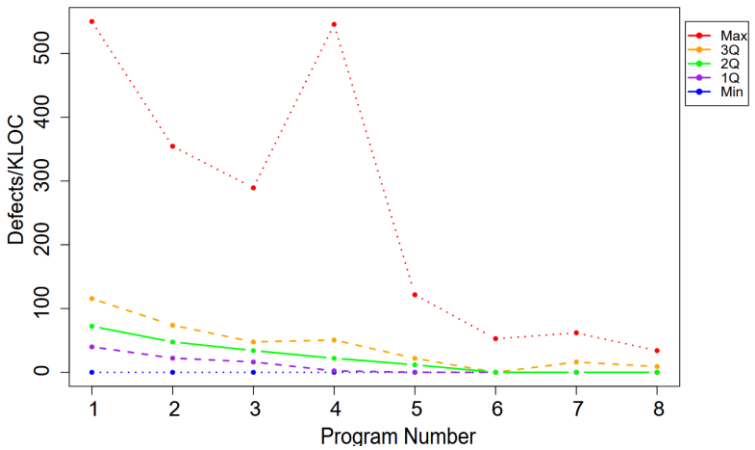



Figure 6: Trends in compilation defect density

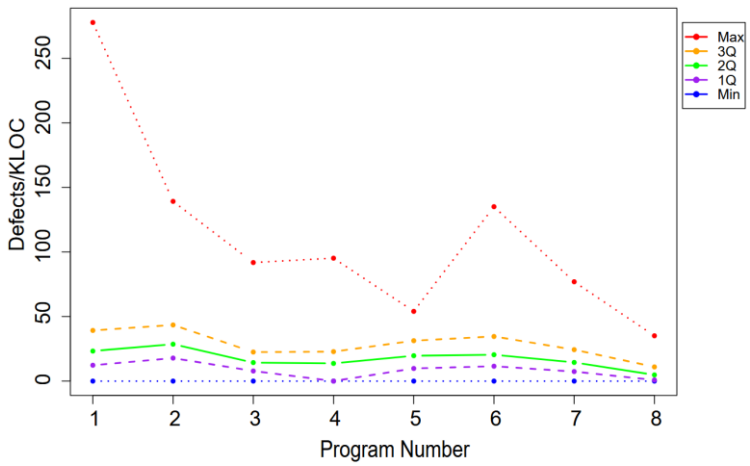



Figure 7: Trends in test defect density

In addition, the test defect density, with some exceptions, shows a decreasing trend as the course progressed, and the defect density of the third quartile in assignment 8 is lower than the defect density of the first quartile in assignment 1. In other words, many of the students who injected many defects at the beginning of the course showed an improvement that exceeded the quality that was considered excellent at the beginning of the course. This can also be confirmed by the fact that the defect removal rate before compilation (process yield) eventually exceeded 70% on average, as shown in Figure 8.

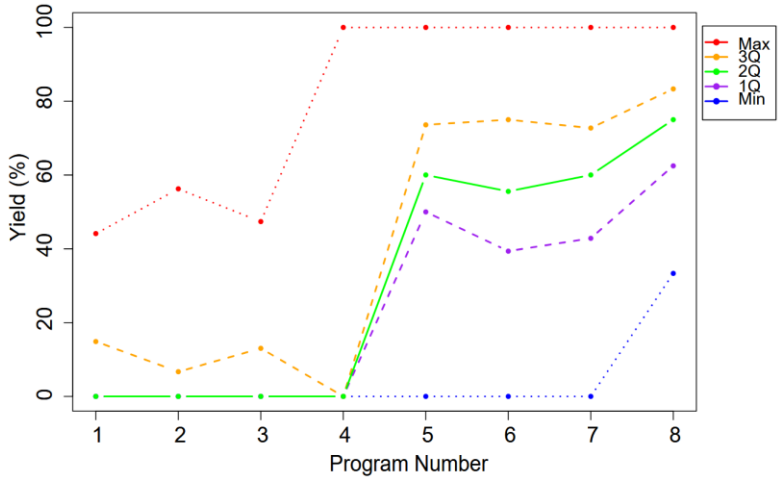



Figure 8: Trends in process yield

Figure 9 and Figure 10 show the changes in the proportion of compilation time and test time in development time, shown by quartiles. The proportion of compilation time decreased as the compilation defect density decreased. On the other hand, the proportion of testing time has not necessarily decreased in proportion to the reduction in test defect density. One reason for this is the large variability in the time consumed to remove one defect during testing.

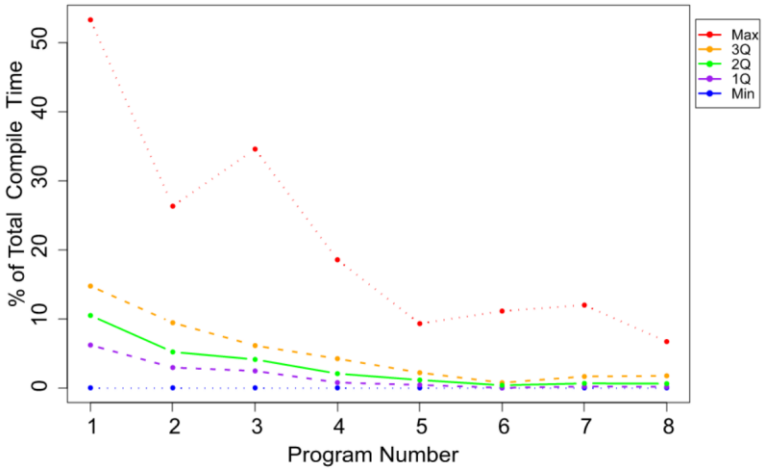



Figure 9: Trends in time ratio of compilation phase

These results are not very different from the results of the PSP for Engineers course, which was conducted for people with work experience in companies, and show that even graduate students with only experience in small-scale programming exercises can acquire the skills necessary to improve software quality. Consequently, the PSP course is an effective tool for this purpose.

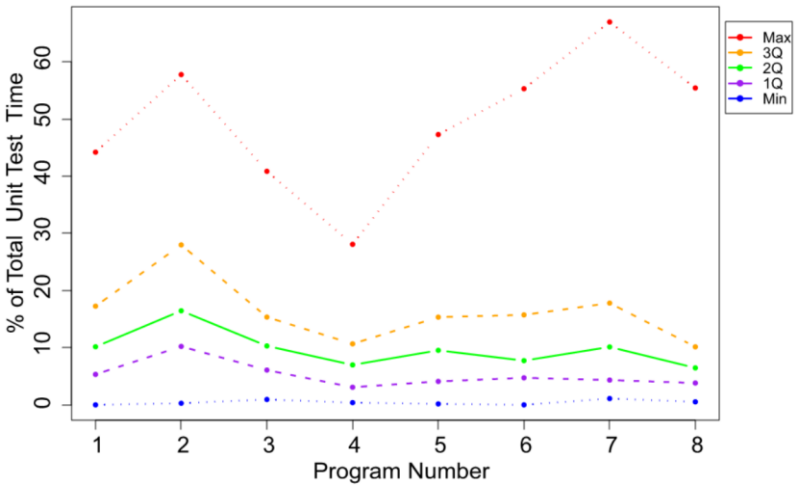



Figure 10: Trends in time ratio of test phase

# 4 Analysis of the Importance of PSP Instructors

## 4.1 Analysis Method

As mentioned previously, PSP instructors have opportunities to provide guidance and advice mainly in two situations: when reviewing plans and when reviewing exercise reports. Here, we examine the changes in the students' process data before and after the guidance and advice during the review of exercise reports. We clarify specific activities of the PSP instructors' role in improving software processes from the changes in the process data records.

The process data analyzed was from PSP course participants who submitted well-formed process data each time they requested a review of their exercise reports. This does not include participants with insufficient process data due to various reasons such as submitting the wrong file when requesting the review.

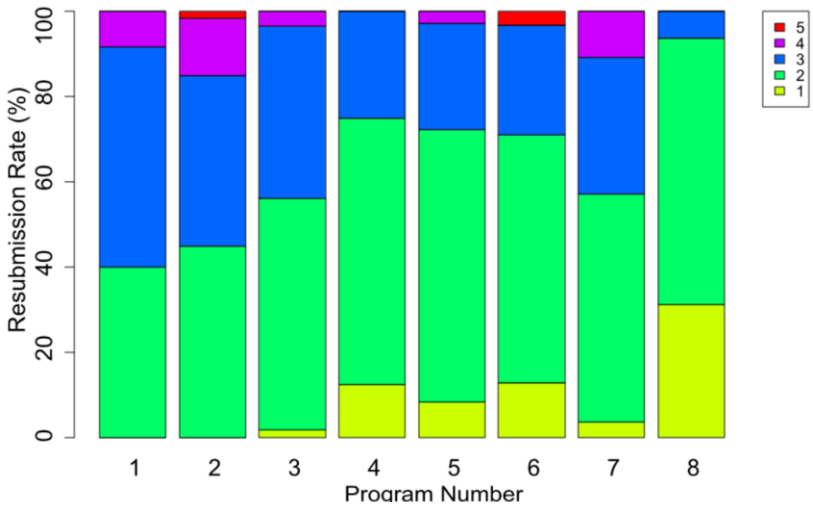



Figure 11: Changes in the number of exercise report submissions

## 4.2 Analysis Results

### *4.2.1 Number of reviews of exercise reports*

Figure 11 shows distribution of the number of reviews of exercise reports per exercise assignment. The number of reviews was a maximum of 5 times, meaning 4-time resubmission, and gradually decreased as the course progressed. There are various reasons for resubmission of exercise reports, but as described later, most of them are due to problems with the accuracy and precision of the process data, such as incorrect defect types. This confirms that the skill of properly recording process data is gradually improved as the course progresses.

### *4.2.2 Correction of defect types*

Figure 12 shows the change in the minimum, average, and maximum ratio of the defect records with defect type correction in defect records. In addition, Figure 13 shows the number of defects by defect type before correction and their cumulative ratio, and Figure 14 shows the number of defect types by type after correction compared to the defect types before correction. As shown in Figure 12, the correction of defect types was initially about 35.7% on average, but decreased as the course progressed, eventually decreasing to almost zero on average. This result indicates that the skill of determining the appropriate defect type was improved as the course progressed. Furthermore, as can be seen from Figure 13, before correction, the defect types "Syntax" were at approximately 22%, "Assignment" at approximately 19%, and "Data" at approximately 18%, with these alone accounting for approximately 60% of the total. On the other hand, as can be seen from Figure 14, most of the defects initially judged as "Syntax" were corrected as other types such as "Interface," "Assignment" and "Function." This means, for example, defects in the order of arguments or data types in functions or methods were initially mistaken as syntax errors.

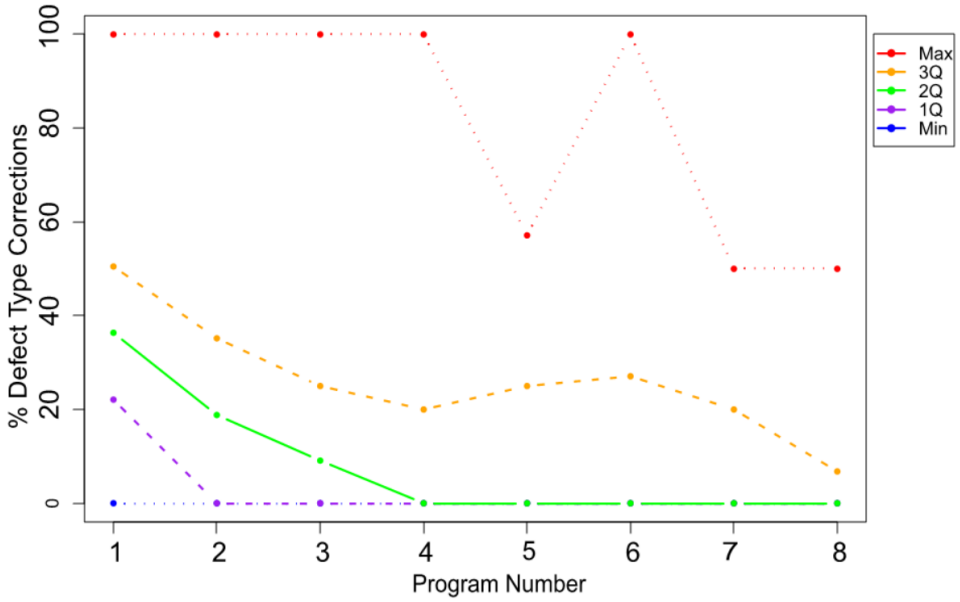



Figure 12: Changes in ratio of defect type correction

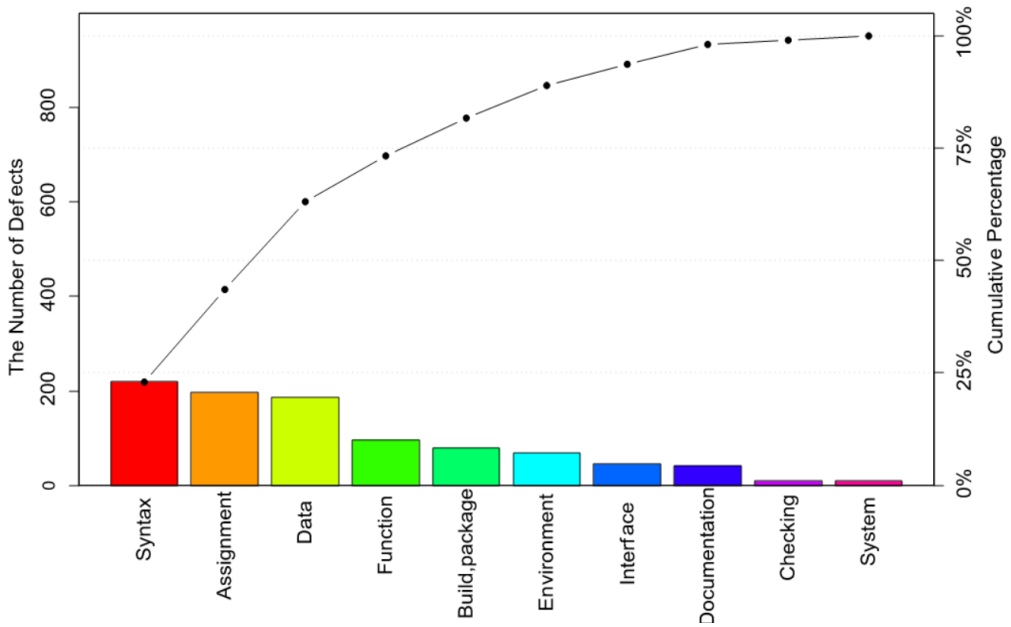



Figure 13: Percentage of defect types before instructor review

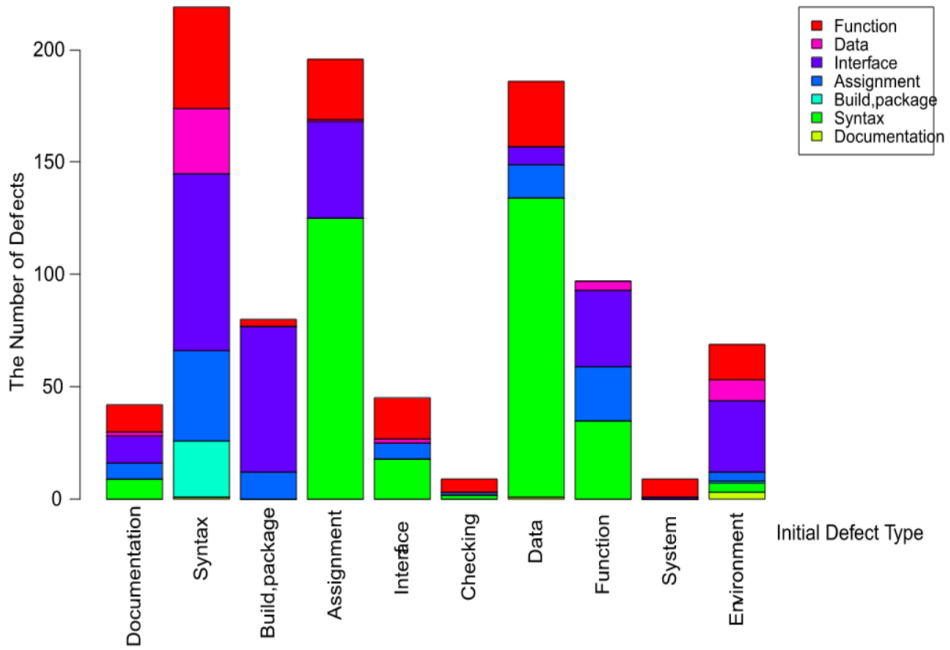



Figure 14: Defect type after correction against initial defect type.

# 5 Issues in Process Improvement through Self-Study

## 5.1 Limitations of Self-improvement through Self-study

Defect types are important clues for devising improvement related defects. Countermeasures for

preventing and removing defects vary depending on the defect type, errors in defect types are likely to be an obstacle to self-improvement. However, as analyzed in the previous section, analyzing defect type is difficult especially for beginners.

The defect injection phase is an important clue for determining in which phase countermeasures against defect injection should be taken, and the defect removal phase is an important clue for determining how defects should be removed and how defect injection should have been prevented in the first place. Countermeasures against defect injection, for example, are generally different at the time of design and at the time of coding. For this reason, as with defect types, errors in the injection and removal phases are likely to be an obstacle to self-improvement.
In this way, process data such as defect types and defect injection phases are clues for self-improvement, and if they lack accuracy and precision, they may hinder self-improvement. However, it is hard for trainees to judge the accuracy and precision of process data on their own, and that the skills to judge the validity of process data are improved and established through guidance and advice from PSP instructors. These results show the limitations of self-learning of software process improvement without the intervention of qualified one such as a PSP instructor.

### 5.2 Quality Assurance of PSP Education Courses and PSP Instructors

Literally, self-study of PSP is now possible using lecture materials that have been transferred to the Creative Commons license. Therefore, it is easier than ever to realize self-improvement of software processes. However, as mentioned above, guidance and advice from PSP instructors plays an important role in self-improvement, and it has become clear that self-improvement through self-study alone has its limitations. Therefore, efficient and effective self-improvement need PSP for Engineers courses accompanied with guidance and advice from PSP instructors.
On the other hand, with the transition to the Creative Commons license, the certification system for PSP instructors by SEI has also ended. As a result, the certified framework to educate PSP instructor has been lost. Creating the next quality assurance framework is a major challenge that the PSP/TSP community must address in the future.

## 6 Conclusion

This paper presents the analysis results of the importance of PSP instructor role in addition to the effectiveness of PSP itself in educating graduate students while PSP was originally developed for industrial software engineers. Our results indicate it is not necessarily easy to appropriately judge the accuracy and precision of process data such as defect types and defect injection phases by oneself, and clarify that the guidance and advice by the PSP instructor plays an important role in self-improvement of the software process.

In addition, there are a certain number of participants in the PSP course who get stuck in the exercises and unable to complete the course in our graduate school. The completion rate is only about 20% to 30%. It is believed that whether participants can continue the course until they have acquired new skills is closely related to their motivation. Improving educational methods of PSP instructor focusing on motivation is also an important issue for our future work [11][12][13].

## References

[1] W. S. Humphrey: A Discipline for Software Engineering, Addison-Wesley, 1995.

[2] W. S. Humphrey: A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005.

[3] Software Engineering Institute: PSP Academic Material, Carnegie Mellon University, Pittsburgh, 2011.

[4] Software Engineering Institute: Team Software Process (TSP) and Personal Software Process (PSP) Materials, https://www.sei.cmu.edu/go/tsp, 2019.

[5] Creative Commons: Creative Commons International Public License https://creativecommons.org/licenses/by/4.0/

[6] H. Kaiya et. al: How process improvement education should be in universities – based on experience in implementing the PSP method, in Proceedings of Software Symposium, p. 137-142, 2001 (In Japanese).

[7] W. S. Humphrey: Introduction to the Team Software Process, Addison-Wesley, 1999.

[8] W.S. Humphrey: TSP: Leading a Development Team, Addison-Wesley, 2005.

[9] K. Katamine, M. Umeda, M. Hashimoto, and Y. Akiyama: Changing Software Management Culture from Academic, in TSP Symposium 2011 Proceedings, pp. 12–18, 2011.

[10] M. Umeda, K. Katamine, S. Araki, M. Hashimoto, and S. Kusakabe: Is self improvement of software processes possible through self-study?, in Proceedings of Software Symposium, 2019 (In Japanese) .

[11] K. Ishibashi, M. Hashimoto, M. Umeda, K. Katamine, T. Yoshida and Y. Akiyama: A Preliminary Study on Formalization of Motivation Process in Personal Software Process Course, in the Proceedings of the 10th Joint Conference on Knowledge-Based Software Engineering, pp.128–137, 2012.

[12] M. Umeda, K. Katamine, K. Ishibashi, M. Hashimoto and T. Yoshida: Motivation Process Formalization and its Application to Education Improvement for the Personal Software Process Course, IEICE Transactions on Information and Systems, Vol.E97-D, No.5, pp.1127–1138, 2014.

[13] S. Kusakabe, M. Umeda, K. Katamine, and K. Ishibashi: Managing Personal Software Process Education Course Based on Motivation Process Model by Using System-Theoretic Method STAMP/STPA, in Proceedings of the 11th International Conference on Project Management (ProMAC2017), pp. 1066–1072, 2017.