SCRUM-PSP: Embracing Process Agility and Discipline

Guoping Rong, Dong Shao
Software Institute
Nanjing University
Nanjing, P.R.China
ronggp@gmail.com

He Zhang
National ICT Australia
University of New South Wales
he.zhang@nicta.com.au

***Abstract*—With the research and debates on software process, the mainstream software processes can be grouped into two categories, the plan-driven (disciplined) processes and the agile processes. In terms of the classification, personal software process (PSP) is a typical plan-driven process while SCRUM is an agile-style instance. Although they are distinct from each other per se, our research found that PSP and SCRUM may also complement each other when SCRUM provides an agile process management framework, and PSP provides the skills and disciplines that a qualified team member needs to estimate, plan and manage his/her job. This paper proposes an integrated process model, SCRUM-PSP, which combines the strengths of each. We also verified that this integrated process by adopting it into a real project environment where typical agile processes are favored, i.e. change-prone requirements, rapid development, fast delivery, etc. As a result, manageability and predictability which traditional plan-driven processes usually benefit can also be achieved. The work described in this paper is a worthy attempt to embrace both process agility and discipline.**

Keywords- PSP SCRUM Integration

I. INTRODUCTION

Personal Software Process (PSP) and SCRUM are usually considered opposite to each other. PSP is a typical plan-driven process which emphasizes discipline, whereas SCRUM is a typical agile process which encourages agility.

PSP is a software process for individual software engineers and was developed by Watts. S. Humphrey [5]. PSP includes well-defined steps, forms, standards and scripts. It provides a measurement and analysis framework at individual level to characterize and manage personal work. The key metrics in PSP are time, size and defects. PSP also defines the process improvement framework, which describes the detailed practices at different maturity levels and guides the self-improvement of individual developer. A typical PSP process includes phases like planning, design, code, compile, test and postmortem. A script is ready for each phase. PSP is also a continuously evolving software process, which suggests seven main levels. Each level introduces new practices based on its prior levels. These seven levels comprise the individual software process improvement framework, and provide a roadmap towards a mature software engineer.

Proposed by Takeuchi and Nonaka in 1987, SCRUM is an iterative and incremental software development approach to describe an efficient and flexible product development process. It was refined, supplemented and introduced into software industry by Ken Schawaber and Jeff Sutherland, and became to take shape [1, 2].

The key feature of SCRUM lies in its iterative development strategy. In each iteration, called a Sprint, the team reviews the latest product requirements, selects the technology used, and develops a consistent development strategy based on the evaluation of both the team capability and product requirements. In SCRUM, a Sprint typically ranges from 2 to 4 weeks. There are defined goals and stable requirements in one Sprint, which means all requirements changes will be put aside until next Sprint. There is a Sprint plan meeting at the beginning of a Sprint, during which the team chooses and prioritizes the feature list to be developed in this Sprint. A simple estimation is performed in the plan meeting to derive the plan and balance the workload. A SCRUM daily meeting is held to evaluate the progress and issues. In SCRUM, a lot of work is done to prepare the demonstration in the sprint review meeting, in which the management gets to know the status of the product. The team must hand over the planned deliverables at the end of the sprint. The sprint retrospective meeting at the end of the sprint provides a self-improving opportunity for the team, in which the performance of the team and individuals are summarized.

Like other agile processes, the performance of SCRUM depends largely on the capability of involved team members. In [3, 4], Turk et al. made a detailed description about the prerequisites and limitations of SCRUM and other agile processes. As Turk et al. pointed out, the team members should have enough experiences and skills to define and improve process in order to implement SCRUM. This implies, the SCRUM team should consist of smart, capable, and experienced staff who are able to support process evolution effectively. As a management process framework, however, few engineering practices are addressed by SCRUM. As a result, it is common to combine SCRUM with other agile practice-oriented process models, such as XP (Extreme Programming) and LD (Lean Development). For example, some typical XP practices such as pair-programming and test-driven development are adopted by SCRUM, because both SCRUM and XP are similar in philosophy and values.

However, modern software development requires not only adaptability which is better inspired by agile processes, but also predictability which is better supported by plan-driven processes. With a thorough review of PSP and SCRUM, we found that they are not conflicting to each other.

PSP is an individual level process, focusing on the improvement of software engineer's process capability, while SCRUM is a team level process framework, concentrating on team cooperation and adaptability to project environment. In this paper, we propose an integrated software process named SCRUM-PSP, which systematically combines the strengths of SCRUM and PSP. Compared to traditional SCRUM, the SCRUM-PSP provides more concrete practices that guide and support the development team to achieve better manageability, more reasonable estimation, and quality in control.

The rest of the paper is structured as follows. Section 2 provides a brief introduction to the related work of this paper. Section 3 explains the iterative process life-cycle and typical iteration of SCRUM-PSP in detail. Section 4 describes the application of SCRUM-PSP in a real project. Some process data were collected and analyzed to verify the performance. Section 5 further discusses several limitations on SCRUM-PSP model at this stage. The paper is concluded in Section 6 with the suggestions on future continuous research.

II. Related Work

A technical note from SEI [6] has verified that there are hardly any conflicts between agile and plan-driven processes. Experience and research in XP and other processes [6,7,8,9,10] have shown that agile and plan-driven processes can supplement each other. Barry Boehm and Richard Turner's work [11] demonstrates that there exists supplementation as well as need by modern software engineering.

While all the effort above addressed the feasibility to combine different processes, there still remain important issues of implementation of the combination, e.g., 1) *how to set up a team?* 2) *how to select candidate processes to be integrated?* and 3) *how to integrate the processes?*

1) How to set up the team? Alistair Cockburn and Jim Highsmith emphasize several critical people factors for agile methods: amicability, talent, skill, and communication[17]. Skills of design, coding and testing are easily recognized for agile methods, but skills of estimating, planning, quality management might be neglected or even misunderstood by some agile method advocators. As we discussed above, PSP focuses on increasing self-management and self-improvement skills for software engineers, and is suitable to be used to recruit and set up a qualified team. Suphak Suwanya and Werasak Kurutach[18] propose a software process improvement model, in which PSP is used to train developers to build discipline and SCRUM is used to manage software projects. But we believe PSP offers more than a training tool.

2) How to select candidate processes to be integrated? The desired new process should provide both the adaptability and predictability which meets the needs of modern software development. Besides, we also expect the integrated process provides more concrete practice guidance for the team. Hence we select PSP and SCRUM as the candidate processes. PSP provides not only detailed practice to guide software development but also discipline and quantitative management to achieve predictability for individual software engineer. While SCRUM is an process "wrapper" for both agile and non-agile practices[6].

3) How to integrate the processes? PSP should work well with SCRUM, but detailed instructions must be provided to the process performers. In the integrated process proposed, PSP can be used not only to manage personal work but also to improve the process capability of team members when a Sprint and SCRUM is used to construct team. Detailed steps and instructions are also provided to support development teams. Our specific methods used to integrate PSP and SCRUM are depicted in the following sections.

III. SCRUM-PSP

SCRUM-PSP is designed as two layers. The *lifecycle* layer describes the main process framework. There are several iterations turning customer requirements into final products. Figure 1 depicts a typical SCRUM-PSP with multiple iterations. The *iteration* layer describes specific steps within iteration. Usually, there are five phases in one iteration, namely Launch (L)/Re-Launch (RL), Plan (P), Requirement & Design (R&D), Construction(C) and iteration Postmortem (PM), as depicted in Figure 2.

A. SCRUM-PSP Lifecycle

SCRUM-PSP is an iterative process in the first place. The whole development work is divided into several iterations. At the early stage of the first iteration, the team discusses the development strategy. Typical development strategy includes priority of requirements, number of iterations, cycle time and phases in each iteration. Usually, the cycle time is fixed for about one month, but it's up to the team's choices to make the duration flexible. Under either situation, the plan is reliable and practical due to introducing PSP methods in estimation and planning activities based on the historical data.

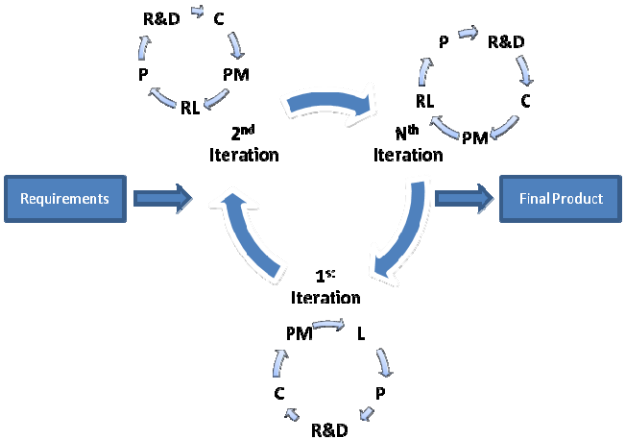


Fig. 1 Lifecycle of SCRUM-PSP

B. SCRUM-PSP Iteration

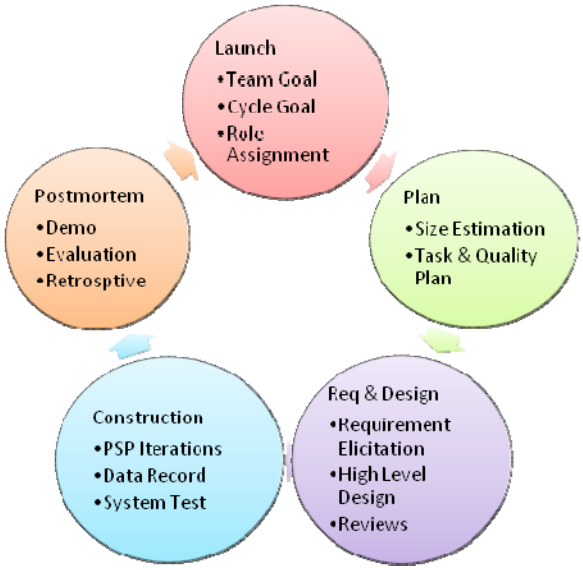
1) Launch. An iteration starts with a launch phase. During launch, the team is established by identifying the team goals and the iteration goals. The team members build relationship, are assigned roles and reach consensus about all the goals. The launches in iterations other than the first one are called a re-launch. Although team building is not a required task during re-launch, new team goals and iteration goals usually still need to identified and throughly discussed to reach consensus among team members.

2) Plan. The development strategy and plan is set up in the plan phase. To be specific, there are several main tasks in the plan phase including size estimation, task and quality plan development, risk assessment and strategy selection and other necessary plans development.

- Size Estimation. Differing from formal designs, the conceptual design aims at helping the team clarify the scope of the project, so as to predict the project size effectively. The strategy is quite simular to developing a WBS (Work Breakdown Structure), breaking the work to a level that historical data is available or the size is small enough so that it can be estimated with confidence. The requirements are understood roughly at that time. In our real project cases, we found that usually, with detailed conceptual design, the estimation does not differ much from the final actual value of the project size. Therefore, the project schedule plan which is based on the empirical estimation could be practical and realistic.

- Task and Quality Plan. Task plan is developed based on size estimation results and team resource estimation results. The rationale behind is that all the software engineers should have enough resource to complete the allocated tasks regardless of the project deadline. Then, each engineer can determine the schedule plan according to his/her resource level(hours each week). If the schedule plan does not match the project deadline, alternative plans should be developed and approved by stakeholder.

The quality plan identifies the quality index that will be tracked based on quality goals. In SCRUM-PSP, it is recommended to track core PSP quality indices such as *phase yield*, *review rate*, *PQI* and *A/FR*[5]. *Phase yield* is calculated as the defects removed during a phase (Review, Compile, UT, etc.) as a percentage of those present at the start of the phase plus those injected during that phase. *Review rate* can be used as guideline to help development team conduct reviews and inspection. In [13], Chris F. Kemerer and Mark C. Paulk suggested that the code inspection rate should be lower than 200 LOC per hour or 4 pages per hour for documentation. It has been justified in [13,14,15] that there is a high correlation between these quality indices and the quality of final product. *PQI* is the product of five values, namely *design quality*, *design review quality*, *code review quality*, *code quality* and *program quality*.



✓ *Design quality* requires the time spent on design phase should be longer than the time spent on coding phase.

✓ *Design review quality* requires that the time spent on design review phase should be longer than half of the time spent on design phase.

✓ *Code review quality* requires that the time spent on code review phase should be longer than half of the time spent on coding phase.

✓ *Code quality* requires that the defect density of complie phase (if exists) should be no more than 10 defects/KLOC.

✓ *Program quality* requires that the defect density of unit test phase should be no more than 5 defects/KLOC. A product value closer to 1.0 indicates a better quality of the process. *A/FR* is calculated as the appraisal cost divided by failure cost. Usually the appraisal cost means the time spent on reviews and inspections and the failure cost means the time spent on compile and unit test. A value more than 2.0 is suggest for *A/FR*.

- Risk assessment and strategy selection. The team identifies and evaluates the risks of the project. The risk is evaluated by possibility and impact. All risks need to be tracked except those of extreme low possibility and impact. Besides, risk assessment helps the selection of development strategy, which is explained in detail by Barry Boehm and Richard Turner with balanced strategy selection [11]. The team will choose the most suitable strategy according to the risks that the team will face in future.

- Other necessary plans development. Apart from schudule plan, several other plans are developed at this step as well. Some typical plans include configuration plan, quality plan, data collection plan and project monitoring plan.

✓ Configuration plan. Configuration management is essential for team software development. A good

configuration plan can guide the change control of configuration repository which leads to integrity and consistency of the work products.

✓ Data collection plan. To support objective data based decision, several basic measures are suggested by SCRUM-PSP, the time spent in each phase, the size of the program and the defects injected and removed in all the phases. Data collection plan defines the procedure to collect and store these data. Project team uses this plan to guide the data collection during the development process.

✓ Project monitoring plan. SCRUM-PSP uses daily meeting and earned value to track the progress of the project. Earned value is caculated as the percentage of total plan time that each task represents [5]. A 0-100 rule is applied to caculate the cumulative earned value, which means that 0 value is earned for partially completed task unless the task is totally completed. Using some tools, SCRUM-PSP can monitor the project status and predict complete dates in a real-time manner and take corrective actions as early as possible. The project monitoring plan defines the activities mentioned here.

3) Requirements and design. The requirements Elicitation and high-level design is performed in this phase.

● Requirements Elicitation. Requirements development in SCRUM-PSP is relatively simple. Based on the results of conceptual design, the team discusses the feature list to be developed in the current iteration, usually the product representative answers the questions to clarify the requirements. A software requirement specification is then developed based on the discussion and reviewed by both the product representative and the team. During one iteration, requirement changes are recorded but not implemented straightway.

● High-level design. High-level design deals with the overall architecture of the system, establishing a complete, correct and extensible foundation for the product. A good architecture improves the efficiency and quality of development and testing, adapts to requirement changes and reduces the risks of rework. In fact, requirement and design go not sequentially but simultaneously in SCRUM-PSP. It is suggested to postpone decisions, which means to keep a few alternative plans and make decisions with more detailed information about the requirements when some technical obstacles come across.

There is no regulation on specific design methods, but some suggested criteria for a complete and consist design which can be taken as a guideline of design are available. The criteria is summarized in Table 1

Such a completeness standard can be applied in a wide variety of design levels, in order to establish a consistent and reviewable design specification.

● Reviews. Both requirement specification and high-level design specification should be reviewed by all the team members. During the reviews in this phase, to remove the defects is not the only purpose, but to form common vision of the product.

Table 1. The criteria for a complete design [5].

	Dynamic	Static
External	Operational Specification	Functional Specification
Internal	State Specification	Logic Specification

4) Construction. The main purpose of construction phase is to construct the software system according to requirements specification and design specification. SCRUM-PSP requires a PSP2.1 process for each module constructed by individual software engineers. At the end of construction, integration test and system test are optional tasks for a certain iteration, the team will decide their test in terms of the strategy selected.

● PSP Iteration. A PSP2.1 process consists of eight phases, namely planning, detailed design, detailed design review, code, code review, compile, unit test and personal postmortem.

✓ Planning: To produce a detailed plan for developing the program defined by the module requirements. Probe [5] method is used to help the software engineer do more accurate and reasonable estimation. Differ from the team plan phase in figure 2, personal historical data is used in module-level size estimation and time estimation. Besides, linear regression is also used to make the estimation more reasonable.

✓ Detailed Design: To produce a detailed design specification for the program defined by the module requirements.

✓ Detailed Design Review: To review a detailed design developed during the design phases by individual software engineer. It's suggested to use a customized checklist to improve the efficiency of review.

✓ Code: To transform the design specification into programming language statements.

✓ Code Review: To review the code developed during the coding phases by individual software engineer. It's suggested to use a customized checklist to improve the efficiency of review.

✓ Compile: To translate the programming language statements into executable code. Most syntax defects will be removed during this phase. This phase is an optional phase determined by the development environment.

✓ Unit Test: To verify that the executable code satisfies the requirements.

✓ Personal Postmortem: To summarize and analyze the project process data. The data includes both plan value and actual value of size, quality and time. Work products from personal postmortem lay foundation for the iteration postmortem.

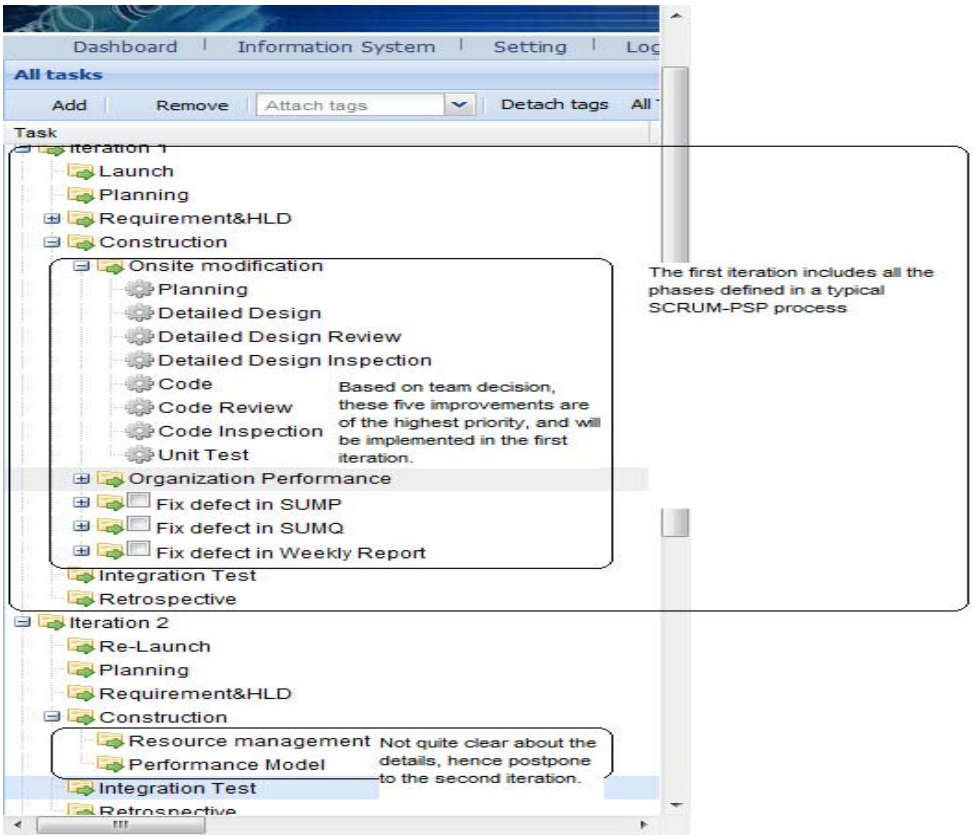


Fig. 3 Plan Structure of the Upgrading Project

Besides, to achieve high yield and A/FR, we add two inspections in detailed design and code. During the inspection, more than two developers inspect the same work product.

✓ Detailed Design Inspection: To inspect detailed design documents with team after the work product is reviewed by the owner. This activity must be conducted before entry to the code phase.

✓ Code Inspection: To inspect code with team after it is reviewd by the owner. Although code inpection can be put either before unit test or after unit test, we suggestion developers do the code inspection before the unit test.

5) Iteration Postmortem: The main tasks in postmortem phase are iteration product demonstration and process retrospect.

● Iteration product demonstration. This is a very important activity to verify iteration products. In a typical demonstration, senior management, customer representatives and other colleagues are invited. The team should introduce iteration goals, team performance and work products achieved in this iteration. A workable system must be demonstrated in this activity. Through real system demonstration, team work results can be learned and acknowledged by the management and the customer representative, the team motivation gets improved.

● Evaluation. During the demonstration, senior management and customer representatives will evulate the current software product. The team collects feedback and suggestions, hence the team can understand better about the expections from stakeholders. Besides, since senior managers are invited to attend the demonstration, the team will take the commitments much more seriously than usually.

● Process retrospect. After iteration product demonstration, iteration process retrospect meeting is then held by the entire project team. In a typical retrospect meeting, team leader leads the team discussion of the status of the project, the deviation of estimation, the quality status, the current risks and other issues. Based on historical process data, the whole project team identifies improvement opportunities. During these practices, full participation of the team members is highly encouraged.

C. Application of SCRUM-PSP

SCRUM-PSP includes process elements from both SCRUM and PSP communities, and aims to enhance SCRUM with more concrete practices and improved

manageability and predictability. Like all other methods, one size does not fit all. Using the method raised by Barry Boehm, Richard Turner and Cockburn [11, 12], we believe SCRUM-PSP is more suitable for projects with the following characteristics:

1) Project characteristic. SCRUM-PSP is suitable for small to medium sized projects, in which complete requirements cannot be clearly defined in the early stage of project and with high risk of change; meanwhile fixed delivery date is determined by several factors such as marketing, customer constraints and budget etc. SCRUM-PSP focuses on self-directed team, by which high performance (short iteration time, high productivity, high product quality, quick response to requirement changes and predictive delivery time) can be expected.

2) Management characteristic. SCRUM-PSP requires a product stakeholder who acts as the customer representative. This commitment improves the efficiency and effectiveness of requirements elicitation. SCRUM-PSP also requires a team coach to facilitate team-building and team-working. Similar to most coaches in a sports team, the coach in SCRUM-PSP process acts as a mentor to all the team members. He/She should have lots of experience on process management and improvement. In addition, any organization adopts SCRUM-PSP need to form a management culture of empirical decision based on the historical data. The project needs to be free in selecting development strategies and processes, however, iterative and incremental strategies are highly recommended. Besides, a powerful and intuitive tool is a necessary support to data collection, planing and progress tracking.

3) Technology characteristic. SCRUM-PSP does not specify technology to acquire requirements and design solutions, nor does SCRUM-PSP require specific technology to implement and integrate the product. SCRUM-PSP accepts both informal and formal requirements specifications. The project team works with product stakeholders to determine the priority of the requirements.

4) Staff characteristic. SCRUM-PSP requires all the team members receive a complete training on PSP before they can participate in the project. All the team members are knowledgeable in process improvements and evolution. They are also fimilar with proces measurement and used to make decision based on historical data. Besides, all the team members should have no technology obstacles to a certain project. Due to fixed deadline, formal training on technology is usually not provided during the whole lifecycle of SCRUM-PSP. In case where technology is unfaimilar to some of the team members, certain resources for training should be considered before launching the project. To facilitate communication, all the team members are suggested to be co-located.

IV. CASE STUDY: A SCRUM-PSP PILOT PROJECT

A. Project background

To verify whether the idea behind SCRUM-PSP is feasible and useful, we applied SCRUM-PSP in one real project. The main purpose of this project is to upgrade a current web based application in use with more features. We used this web based application in the past two years to provide the student teams a project management supporting tool. Mainly, the tool can be used in planning, earned value tracking, quality status tracking and collection of process data. When several improvements have been identified by both students and faculty, we decided to upgrade this system before the start of a new semester. Typical improvements include project objectives such as to fix some defects, to provide a mechanism to manage resource and arrange schedule, to offer an onsite version of the system and a mechanism to gather process data from different projects, and further to establish an organizational process performance baseline. Although some improvements are explicit, several others such as resource management and performance model still remain unclear.

B. Early stage

There are several facts of this project, in which we adopted SCRUM-PSP as the development process: (1) Some requirements cannot be decided at the early stage and have a high possibility to change; (2) As the students would use this web application in one of the courses since the beginning of the coming semester, the deadline of this project is not negotiable; (3) All the five team members received a full training on PSP; (4) The development team and the requirements providers are co-located, which facilitates the communication and discussion; (5) The existing system provides good basis to support continuous and incremental integration.

At the early stage during the launch, the project team worked together with product stakeholder to discuss the requirements. Directed by a coach, the five team members reached consensus on team goals and team roles. Then they began to develop plans for the project. According to project context, the team decided to divide the project into three iterations. Based on acknowledged criteria, the team selected five modules to develop in the first iteration, in which each developer needed to establish a detailed plan for his/her module. Then a schedule plan containing three iterations is established and maintained (as shown in Fig. 3). In this SCRUM-PSP process based schedule plan, each iteration

WeeklyData	Plan	Actual	Plan / Actual	Plan - Actual
Hours for this week	55.0	60.02	0.92	-5.02
Hours to date	156.48	165.0	0.95	-8.52
Earned value for this week	15.00	12.8	1.17	2.2
Earned value to date	40.00	36.48	1.09	3.52
To-date hours for tasks completed	156.48	165.0	0.95	-8.52
To-date average hours per week	52.16	55	0.95	-2.84
EV per completed task hour to date	0.26	0.22		

Fig.4 Weekly Report for the Third Week

contains all the phases defined in a typical SCRUM-PSP process. For construction phase, a complete PSP2.1 process is used. For the first iteration, a detailed plan is established based on the latest agreed requirements. For the second and the third iteration, only time box is defined. As the development advances, we got more and more clear about the project requirements and received more data gathered from prior iterations, which enabled more accurate estimation about the size and time of the new iterations. As a result, we managed to finish all the improvements and deployed the upgraded system before the start of the new semester.

C. *Mid stage*

To facilitate project management and data collection, we used a web based supporting tool. This tool provides a set of features such as estimation and planning (cf. Fig. 3), time log recording, defect log recording, weekly reporting (cf. Fig. 4), plan summary, process quality index summary (cf. Fig. 5 through Fig. 7), etc. To reduce difficulty, most features are provided with similar style to the supporting tool [16] the team members used when they were receiving PSP training. Feedback from the practitioners indicates that this tool plays a vital role to the success of SCRUM-PSP process project.

Weekly report is auto-generated by the supporting tool based on the process data each team member recorded when they were developing the system. The report provides rich information for all the team members. Fig. 4 illustrates a weekly report for the third week of the pilot project, which shows process data for both plan value and actual value. From the process data, the project team can draw several conclusions as the following:

1) For the earned value to date, the actual value is 36.48 which is less than planned value 40.00. This indicates that the project team is a little behind the schedule.

2) For the hours to date, the actual value is 165.0, which is bigger than the plan value 156.48. This indicates that the team spent more time than planned.

3) For the to-date hours for tasks completed, the actual value is 165.0, which is larger than the planned value 156.48. This indicates that the team might a little under-estimate the size and effort needed. EV per completed task hour to data value also indicates similar conclusion.

4) From all the conclusions above, the project team may have some progress issues. They spent more time than planned while gained less earned value. Although there was another possibility that the project team put some time into some unfinished tasks which impact the earned value for them.

With a discussion on the weekly meeting, the project team determined the root cause of the issue and resolved it by modifying the original estimation on size and effort needed. In addition, they also managed to improve resource level to nearly 12 hours per week for each developer to meet the new situation.

D. *Project summary*

After three iterations, the project team managed to deploy the system before the deadline. All agreed improvements to the original system have been implemented. The process data were collected and summarized in Table 2. The team achieved a less than 5% deviation in both size and time estimation. Compared with an average time estimation deviation 63% in the industry [19], we considered this performance can be regarded as evidence of good manageability and predictability of the example project. Besides, we used PQI to control the process quality. This helped to assess the quality of the components. PQI is a very important quality assurance method in SCRUM-PSP. Fig. 4, 5 and 6 illustrate the actual PQI data during these three iterations. We can see an improvement (the percentage of shade area becomes more and more close to 100%) in these figures which usually means improved quality. The project result summary (Table 2) shows the decrease of the defect density during integration testing through these three iterations.

Table 2 Process Data of the Upgrading Project

Iteration	Iteration 1		Iteration 2		Iteration 3		Total	
	P*	A*	P	A	P	A	P	A
Size(LOC for JAVA)	3250	3641	4300	4107	1150	994	8700	8742
Time(Hour)	392.5	431.2	428.3	439.7	113.83	97.3	934.66	968.2
Productivity	8.28	8.44	10.04	9.34	10.10	10.22	9.31	9.03
Defects in Integration test	68.0	57	64.4	56	16.2	10	148.6	123
Defect Density (pKLOC)	20.9	15.7	14.98	13.64	14.1	10.06	17.8	14.07

*P: Plan Value; A: Actual Value

V. DISCUSSION

Although it's not a brand new idea to integrate different processes, to combine two processes with conflicting values is a challenge. PSP is an individual level process and SCRUM is a team level process. This complementing relation reduces the conflicts between the two processes in integration. However, there still remain several possible limitations, among which the project scale, the developers' skill, consideration of data collection and usage and reconciliation between agile and plan-driven are worthy to discuss.

1) The project scale factor. We believe the scale plays an important role to the success of the pilot project. The project scale is relatively small in terms of either the team size (5 students) or the duration of the project (4.5 months). While most software issues appear when the project scaling up, more research and experiments need to be conducted to make SCRUM-PSP more scalable to deal with projects with more team members and longer duration.

2) The developers' skill factor. The five students in this pilot project are of the top level students in our school. They have plenty experiences on both technology used in the project and PSP, which help them adapt to SCRUM-PSP rapidly and effectively. Whereas, how to apply SCRUM-PSP to those inexperienced software engineers and to achieve the similar benefits still needs further research and experiments.

3) The consideration of data collection. SCRUM-PSP highly relies on process data to help estimating the size and time, tracking the progress and the quality status. However, there is no agreement among all team members on collecting process data. Some argue that measuring software projects is of no use and is not so "agile"[20]. Our strategy is to lower the effort of and the obstacle to software engineers to measure their own process with supporting tools. When software project teams expect to achieve not only adaptability, but also predictability, data collection is necessary.

Besides, more rational metrics should be proposed and applied to SCRUM-PSP to guide the decision making. In the case study, we found that high PQI did not lead to high quality in the final product. The reason is that although students were good at development, they lacked skills to do effective verifications such as reviews and unit test. Hence they did not identify and remove most defects before the entry to the integration phase. PQI works well in PSP assignments, where the problem is quite simple, but for real projects, when problem is much more complicated and difficult, simply measuring by PQI only is not enough.

4) Reconciliation between agile and plan-driven. Serveal features of SCRUM-PSP make us believe the essence of SCRUM such as empiricism, emergence self-organization, prioritization and timeboxing still remain[21].

a) Empiricism requires continuous project monitoring that allows the team and management make data based decisions in real time. In SCRUM-PSP a supporting tool can be used to collect process data such as phase time, product size and defects. The plan value and actual value based on and derived from the data can support team's decisions.

b) Emergence implies that all solutions to all problems will become clear as we work. SCRUM-PSP does not require all the requirements clear enough to start the work, nor does it require perfect high level design. All decisions are based on currently known evidence of the project context. When the context changes, the team will make new decisions.

c) Self-organization requires the teams are empowered to make the important decisions necessary to make the project success. In SCRUM-PSP, team goals, team roles, development strategy and processes are selected and defined by the team. The team will also conduct weekly meetings where they have a chance to make common decisions. SCRUM-PSP coach will work with the team. He/she will not make decisions on behalf of the team but help to establish and maintain self-directed project team.

d) Prioritization means that some features are more important than others. In SCRUM-PSP, the team need to establish criteria to determine the importance and priority of requirements.

e) Timeboxing means one week or serveral weeks will be taken as a time box, usually fixed, during which the team will try to solve portion of the whole problem to establish a basis for the whole system and gain experience on the project. SCRUM-PSP also provide similar feature by the concept of iteration. Although the duration will not always be fixed up to the team's decision.

At the individual level, all the team members use PSP to plan and track their work. With plenty of process data, each team member is aware of the status of their own tasks. This forms a solid basis for the whole team to know the status of the project. Hence necessary corrective actions will be conducted when significant deviations were identified to ensure the success of the whole project.

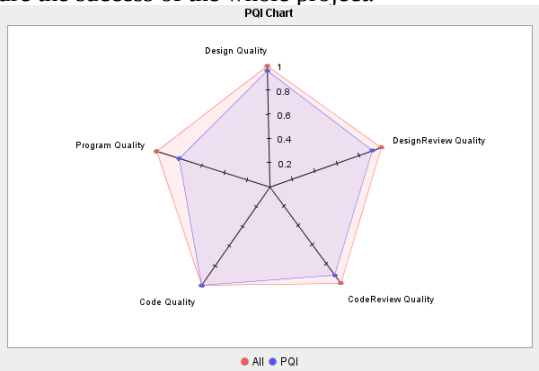


Fig. 5 PQI in Iteration 1

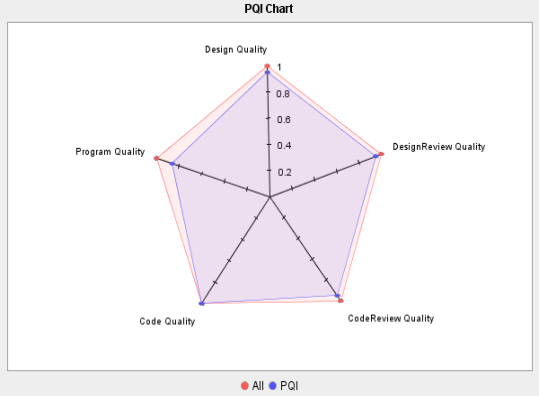


Fig.6 PQI in Iteration 2

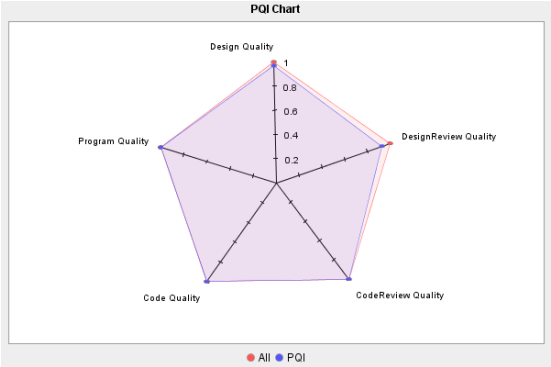


Fig.7 PQI in Iteration 3

VI. CONCLUSIONS

The agile community and plan-driven community should not be taken as contradictory sides. Modern software projects, in which project teams will face more challenges than ever (variable requirements, fixed deadline etc.), require adaptability as well as predictability. In the research behind this paper, we add PSP process elements into typical SCRUM process framework, in order to design an integrated process that includes both agile features and plan-driven features. Our approach (SCRUM-PSP) justifies that PSP enhances SCRUM with concrete practices which will provide more useful and effective guidance to software developers. Besides, the manageability and predictability that are well supported by PSP will benefit individual software engineers with better planning and commitment. When every software engineer's work can be predicted, the team's work becomes predictable. Meanwhile, agile features such as empiricism, emergence self-organization, prioritization and timeboxing are also well supported by SCRUM-PSP.

Our work is a worthy attempt to combine different processes to meet the needs of modern software projects. There still exist several interesting issues which need future research, for example

a) How to apply PSP to other known processes? As we know, PSP is a personal process, which can not only improve individual software engineer's skill, but also manage personal work in software projects. Software development is intellectual undertaking, which needs self-directed development teams. PSP provides the skills for all the developers to form self-directed team. In this sense, PSP is able to support other agile style processes such as XP, DSDM, Crystal, RUP, etc. However, more focused research and experiments need to be conducted in the future.

b) How to combine best practices from various methods and processes? Most published development methods or processes contain indentified best practices in a certain area. To combine these best practices to meet various software project context is challenging. Criteria and guidelines should be established to leverage software project team's better decisions and more efficient work.

REFERENCES

[1] K. Schawaber, M. Beedle, Agile Software Development with SCRUM, NJ: Prentice Hall, 2001

[2] Entry SCRUM in Wiki website. http://en.wikipedia.org/wiki/Scrum_(development)

[3] Turk, Dan; France, Robert; &Rumpe, Bernhard. (2002). "Limitations of Agile Software Processes." Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, p. 43-46, May 26-29, 2002, Alghero, Sardinia, ITALY.

[4] Turk, D., France, R., Rumpe, B. "Agile SoftwareProcesses: Principles, Assumptions and Limitations." Technical Report. Colorado State University, 2002.

[5] Watts S. Humphrey PSP: A Self-Improvement Process for Software Engineers Addison-Wesley, 2005

[6] Hillel Glazer, Jeff Dalton, David Anderson, David J. Mike Konrad,Sandy Shrum, "CMMI® or Agile:Why Not Embrace Both" TECHNICAL NOTE, CMU/SEI-2008-TN-003.

[7] He Huang, Peiji Tao, Xianming, Liu, Qiang Cui "Research and Practice of Reducing and Merging XP with Heavy Soft ware Developing Process" Journal of Computer Engineering and Applications 2003.22

[8] Hui Li, Peiji Tao, Wenfeng Li " Combining XP and RUP to develop small projects" Computer Engineering & Design 2005

[9] LanCao; Mohan, K; PengXu; Ramesh, B."How extreme does extreme programming have to be? Adapting XP practices to large-scale projects"Proceedings of the 37th Annual Hawaii International Conference onSystem Sciences, 2004.

[10] J Wäyrynen, M Bodén, G Boström "Security Engineering and eXtreme Programming: an Impossible marriage?"Extreme Programming and Agile Methods - XP/Agile Universe 2004Springer Berlin / Heidelberg

[11] Barry Boehm Richard Turner Balancing agility and discipline: a guide for the perplexed Addison-Wesley, 2004

[12] Cockburn, A. Agile Software Development.Bostom:Addison-Wesley 2002

[13] Chris F. Kemerer, Mark C. Paulk, "The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data", IEEE transactions on software engineering, 2009 Vol 35 no.4

[14] A.F. Ackerman, L.S. Buchwald, and F.H. Lewski, "SoftwareInspections: An Effective Verification Process," IEEE Software,vol. 6, no. 3, pp. 31-36, May/June 1989.

[15] R.L. Glass, "Inspections—Some Surprising Findings," Comm.ACM, vol. 42, no. 4, pp. 17-19, Apr. 1999

[16] SEI Supporting Tool web site, http://www.sei.cmu.edu/tsp/tools/studypsp-form.cfm

[17] A. Cockburn and J. Highsmith, “Agile Software Development: The People Factor,” Computer, Nov.2001, pp. 131-133.

[18] Suphak Suwanya and Werasak Kurutach “Applying Agility Framework in Small and Medium Enterprises” ASEA 2009, CCIS 59, pp. 102–110, 2009 Springer-Verlag Berlin Heidelberg 2009

[19] Extreme Chaos, The Standish Group International, 2001

[20] Tom Demarco “Software Engineering: An Idea Whose Time Has Come and Gone?” IEEE Software, July/August 2009, pp95-96

[21] Tobias Mayer “The Essence of SCRUM” http://agilethinking.net/essence-of-scrum.html