The Adoption of Agile Software Methodology with Team Software Process (TSPI) Practices in the Software Engineering Undergraduate Course

[1]Nurfauza Jali , [2]Azman Bujang Masli, [3]Cheah Wai Shiang, [4]Yanti Rosmunie Bujang , [5]Abdul Rahman Mat and [6]Norazian Mohd Hamdan

Faculty of Computer Science & Information Technology
Universiti Malaysia Sarawak
94300, Kota Samarahan, Sarawak, Malaysia
Email : [1]jnurfauza@unimas.my, [2]azman@unimas.my, [3]wscheah@unimas.my, [4]yanti@unimas.my, [5]rahman@unimas.my, [6]mhnorazian@unimas.my

Abstract— *In computer science, software engineering courses expose the undergraduate students to both the technical and methodological aspects of software development. The traditional software development methods and techniques represent a huge proportion of the courses and hence contribute an essential part of software engineering students' development process. This plan-driven development is dependent on a set of predefined phases and ongoing documentation which found to be problematic; such as time-consuming, slipped requirements and complicated processes. The main aim of this paper is to study and review the adoption of Agile Software Methodology and Team Software Process (TSPi) practices in the undergraduate course focus on software development. The framework and course plan will be designed to apply and observe the implementation. Furthermore, this study will help to gather the teams' viewpoint regarding the importance of Agile and TSPi practices in handling small projects with real clients.*

Keywords: Software Engineering, Team Software Process, TSPi, Agile, Scrum, Software Estimation, Software Planning.

1 Introduction

In computer science, especially in software engineering courses, the success and failure of students' project depend on the collaboration between the members of the project team. The software engineering students must not only be taught on the theory and technical aspects of the software development discipline, but also the effectiveness of teamwork and social capabilities. The TMP3413 Software Engineering Laboratories (SELab) course at the Faculty of Computer Science and Information Technology (FCSIT), UNIMAS offers to the third year students of Software Engineering programme students who have grasped and self-equipped with the fundamental of various programming languages, software modelling and database management system (DBMS). These skills and knowledge help the students to develop a different kind of systems.

The main aim of this paper is to study and review the adoption of Agile Software Methodology and Team Software Process practices in the undergraduate course focus on software development. The framework and course plan will be designed in order to apply and observe the implementation in future.

This paper organised as follows. The next section introduces the background study and overview of the Agile software methodology and Team Software Project, while Section III discusses the problem statements and construct the research questions based on the study. Section IV discuss the methodology used. Section V summarised the initial research outcomes and drawn up the contribution and future works of the study.

2 Background Study

2.1 Team Software Process (TSPi)

Team Software Project (TSPi) is a pint-down academic version of the Team Software Process (TSP) that is a huge scale, cutting-edge, mechanical quality, a coordinated structure that aides improvement groups in delivering amazing programming concentrated frameworks (Chick et al., 2009) (Humphrey, 2000). TSPi method provides a series of operational processes to the software engineers which can help them to achieve more efficiently and effectively software development and improve the quality and productivity of their software development project. The general concept of TSPi is an iterative and evolving development process, and the phases are shown in Fig. 1.

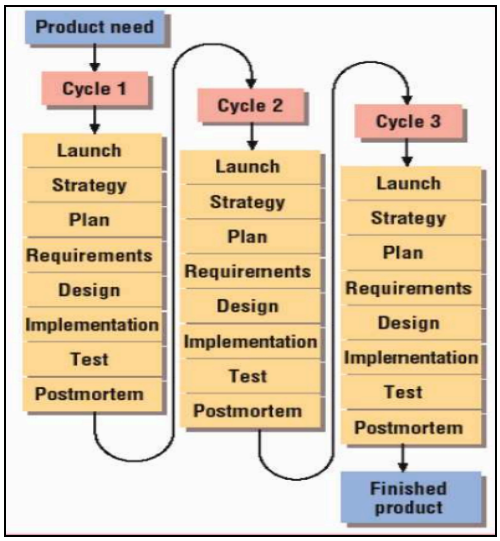

Figure 1: TSPi Process Cycle (Over, 2000)

In the launch phase, the team identify and reassess the objectives, role allocation, customer needs, and team goal. After that, strategy phase required the team to build a conceptual design for the product and set up the development strategy for the cycle and reuse plan. After that, planning phase required the team to estimate the size of SRS, SDS, and code. Task's weekly schedule and a quality plan are required to be done during this phase. Next is requirements phase, which required the team to interview a customer, define and examine the requirements, and test plan creation.

In the design phase, the team required to define and examine the high-level design. The implementation phase is the phase that creates and reassess the design of the modules and units and converts it into the code, reassess the code, compilation, testing, and analysis of the modules and units. In the test phase, the system will be build, integrated, and tested. User documentations also created in this phase. Lastly, post-mortem phase included the activities such as analyse the post-mortem, produce cycle report and evaluation for peer and team (Cannon, Hilburn, & Diaz-Herrera, 2002)(Chick et al., 2009).

2.2 Agile Software Development

In the Agile approach, the process is based on the iterative and incremental development, the phases of the software development life cycle are revisited thoroughly until the users or clients are satisfied with the solutions. Agile software methodology is widely used in the software industries.

According to a survey done by Version One Inc. (Com, 2015), from 2012 – 2014, Agile was spreading globally where the teams practising agile jumped from 35% to 80%. Many software companies who adopted Agile have made a significant improvement in teams productivity, product and services quality, reasonable improvement in cost and stakeholder satisfaction (Moniruzzaman & Hossain, 2013). Moreover, several institutions have adopted this methodology in their software engineering courses (Reichlmayr, 2003)(Abdulwahab, Abdalla, Galadanci, Algudah, & Murtala, 2015). The major success factors of agile practice include in few of the following Agile Manifesto (Fowler & Highsmith, 2001)(Qureshi, Alshamat, & Sabir, 2014)(Fowler & Highsmith, 2001):

- Customer involvement at an early stage.
- The client's satisfaction is the highest priority throughout the project timeline.
- Team-based and collaborative development.
- Adaptation to requirement changes.
- Iterative and incremental development.

Scrum is the most popular agile methodologies applied in the industries and institutions (PricewaterhouseCoopers, 2014)(Despa, 2014) aside from feature-driven development (FDD), lean development, and Adaptive Software Development. Scrum as defined by Ken Schwaber, the founder of Scrum methodology;

"Scrum is a framework for developing complex product and systems. It is grounded in empirical process and control theory. Scrum employs an iterative and incremental approach to optimise predictability and control risk."

Scrum is an iterative and incremental agile software development framework for managing software project and product or application development. This approach required more than one workforce to operate. In 2014, 56% of Agile practitioners used Scrum methods and practices (Com, 2015). Scrum methods and practices are suitable for the software development project with rapid changing and emergent of user requirements, increase of teams productivity and increased the project visibility [8][10].The most common metrics to track Agile projects are velocity, iteration burndown, release burndown and planned vs. actual stories per iteration (Com, 2015).

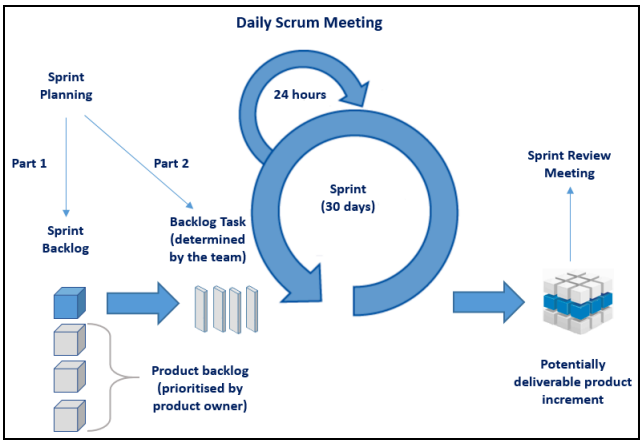

Figure 2: Scrum framework

A Scrum team which consists of a product owner, development team and scrum master is commonly structured in a project with Scrum approach. Product Owner represents the stakeholder, development Team responsible for building and Scrum Master responsible for scrum process. Scrum framework works in an iterative and incremental manner where the software projects are developed by cross-functional teams. As illustrated in Fig. 2.0, the Sprint represents the cycle of development work. These iterations are not more than four weeks each and must be continued without pause. At the beginning of each Sprint, a Team (consist of seven members) select product backlogs (customer requirements) from a prioritised list prepared by the Product Owner. The Team then will determine a backlog task which will be delivered by the end of the Sprint. No new item will be added to the Sprint activity. Every day, the Team together with the Scrum Master will sit in a meeting to inspect team's progress, and plan for the next step to complete the remaining work. At the end of the Sprint, the Team will review the Sprint with the Product Owner and demonstrate what has been developed. Scrum emphasises on a working product which indicates fully 'Done' at the end of the Sprint. This means a system is integrated, tested, documented and potentially deliverable product (Deemer, Benefield, Larman, & Vodde, 2012).

Quite some studies have reported on using Agile as the software development methodology used in the software engineering course. Problems and issue have been observed during the adoption of this practice in the classroom. In the beginning, students were having difficulty to understand the concepts and principles of Agile development (Zorzo, De Ponte, & Lucrédio, 2013)[1(Werner et al., 2012). They face problem to deal with the dynamic environment, technical challenges, personal interaction and inter-team relationship. A comparison study between Unified Process (UP) and Extreme Programming(XP) were also conducted by Steward & Agah (Stewart & Agah, 2012) in teaching software engineering on developing video games. The overall findings indicate that the application of software development methodologies did not have a major impact on the overall structure of software engineering course. However, the quality of codes tested and defects free code generation in XP is much better than UP. The students were having difficulty in the testing phase in XP although the test-driven development was enforced.

2.3 Comparison between Agile Software Development and Team Software Process (TSPi)

The major similarities between TSPi and Agile Project Management are apparent whereby they focus heavily on reducing the costs and increasing efficiency. The differences are rooted in the concept it holds. TSPi allows the team to be self-directed where in Agile, for using the Scrum, there needs to be a Scrum master. TSPi relies on its members to be highly trained and capable while in Agile, the Scrum master will bear the responsibility of handling the team members and also to act as an intermediate between the customers and management outside of the team. Table 1 and 2 brief the similarities and contracts of Agile/Scrum and TSPi.

Table 1: Similarities between Agile/Scrum and TSPi

Agile / Scrum	TSP
Start by Sprint Planning meeting	Start directly with Launch / Relaunch
Sprint is visible, usable, and deliverable and have a goal	Launches or relaunches have one or more team goals
Identifiable list of tasks and requirements of the project	Generate task lists with assignments for short team long term by phase with different granularity
Planned for early sprints by identifying high technical risks	Risk identification with mitigation and contingency planning in launches and relaunches
Scrum master in Agile	Planning Manager, Team leader, TSP Coach, Process Manager, and Timekeeper
Define and discuss risks and issues also remove impediments	Its mitigation and contingencies, monitoring risks and issues

Table 2: Contrast between Agile/Scrum and TSPi

Agile / Scrum	TSP
Sprint duration 1 to 4 weeks and project duration couple of months to infinite time.	Phase duration 1 – 3 months, and project duration depends on the phase duration.
Excluded formal planning.	Including formal planning, must need the plan for real commitment.
Excluded formal roles.	Roles are defined in the launch time and distribute tasks by role wise.
Sprint deadlines are always fixed. So functionality can change but cannot change the date in the "Time-Boxing" perspective.	All committed is kept, but changes of deadline possible. If any changes needed some more time.

3 Problem Statement and Research Questions

In the current practice and implementation of TMP3413 Software Engineering Laboratory (SELab) course, software development projects are adapted the heavyweight practice of Team Software Process (TSPi) which require the project team to prepare quite a number comprehensive artefacts (such as the SRS1, SDS2, STP3 and TSPi forms). Moreover, the students need to follow rigid TSPi process scripts and provide detailed plans, roles, responsibility, workflows and work product descriptions for the system proposed. These artefacts will guide the organisation and use measurements to improve the team as a whole continually.

[1] SRS: Software Requirement Specification
[2] SDS: Software Design Specification
[3] STP: Software Test Plan

This plan-based project course makes use of the modified waterfall model approach and the Unified Process of the software development life-cycle activities. The drawbacks of this approach will limit the team's opportunity to understand and experience the process improvement with the ability to handle the requirement changes. Therefore, the research study will adopt Agile methodology (Scrum) in the current SELab syllabus to promote an evolutionary approach to the software development using short release cycles and engaging the teams with a real system's client and the end-user.

The primary objective of this research is to adopt Scrum methods or apply some of its practices to complete their projects successfully following the user' requirements within a short schedule. The specific objectives of this research include:

- To analyse the development of team' abilities in adopting Scrum concepts (e.g., estimation of user stories, release and iteration planning, effort estimation) for the first time.
- To gather the feedback from the teams regarding the importance of Scrum practices for a successful Scrum projects.
- To identify the limitation need to be considered when applying the findings in the industrial environment.

In the beginning, the planning and estimation tend to be less accurate but will be improved from Sprint to Sprint. The development team will begin to acquire and gather the requirement and user stories based on questions constructed from the potential client or stakeholders. These question will help them to understand the whole system. Furthermore, the clients or known as a product owner will collaborate in helping to break down the requirements into granular pieces. The development team will estimate and prioritise the user stories and reorder them on the backlog. As for the effort estimation, each team members must individually estimate how many hours they take to accomplish each task which has been assigned based on their roles in the team. These task distributions are constituted from the decomposition of the user stories. Due to time constraints, some of the requirements might not be able to be accomplished on the high demand from the real client of the industrial environment.

This research will try to answer the following research questions:

When these Agile software development methodologies are being practised:

- Will the teams able to adapt and accept Scrum concept in software development process?
- What are the practices that help the teams manage a successful Scrum project?
- What are the limitations to be considered when applying the findings in the industrial environment?

This SELab course applies Team Software Process(TSPi) framework and guidelines since 2010. The main objective of TMP3413 Software Engineering Lab course is to prepare the undergraduate students to work collaboratively and applied social software development scenarios.Therefore, the student is required to use TSPi practice (Over, 2000), which is an integrated framework that guides development in producing high-quality software-intensive systems. TSPi consists of 21 number of process scripts, 21 forms, ten role scripts and three standards which need to be recorded, reported and compiled in team's project notebook.

This method is well-organised and documented however it is quite complex and complicated to implement. The work on managing the methodology itself is more than the work of building the software product. Fig. 1.0 illustrates the TSPi product development cycles which consist of seven iterative phases in each cycle where each produces a testable version of the subset of the final project. Based on the observation and evaluation of the previous team's project assessments, the team tends to focus on documenting the required artefacts rather than delivering a quality software product. Unlike Agile, it practices are to ensure the satisfaction of both the client or the end-user and the quality of the end product (Sureshchandra & Shrinivasavadhani, 2008).

4 Methodology

The methodology for this research involved the planning and design of the course activities, implement and test the proposed design and analyse the data gathered from all the processes.

4.1 Planning and Design

During this stage, Teams will be categorised into two groups; Agile–driven methodology software development groups (AG) and Plan-driven methodology software development groups (PG). The AG practices the iterative and incremental approaches to Scrum methods. Hence, PG continuing to adopt the current Unified Process (UP) [12][13] approach and fully applies a heavyweight method of TSPi process which has been a common practice for the course. The experiment will be

conducted for two terms. In the first term, the facilitator will become the product owner (PO) and Scrum master. While in the second term, the teams will work with a real system where they will deal with real system clients like the product owner.

The development of the software product will be conducted over 14 weeks (per term) which will involve the lectures to brief both the software methodologies and the project development processes. Students will be divided into few groups which consist of six (6) to seven (7) members per team.

Table 3 Course and project schedule

Week	PG (UP + TSPi)	AG (Scrum)
1	Team formation and project selection	Team formation and project selection
2	Lecture on the introduction to UP and TSPi	Lecture on the introduction to Scrum
3	Project launch and proposal submission	Project launch and proposal submission
4	Development strategy on product functions and assess risk	Project planning – define the user stories
5	Develop product size estimation plan, task to be completed and quality plan	High-level design
6	Document the Software Requirement Specification (SRS)	Req1: user stories implementation
7	Mid-Term Break	Mid-Term Break
8	Produce a complete and precise software design	Release 1 and the reflection
9	System implementation	Req2: Project planning
10	System Implementation and Integration testing	Req2: user stories implementation
11	Document the Software Design Specification (SDS)	Release 2 / reflection
12	Generate test plan	Final user stories implementation

4.2 Implementation

The planned and design work would next experiment with the two groups of students. The communication between the teams and their clients or users will be observed via the video recording. The use of project tracking software will help to track the progress of each team.

4.3 Testing

The methods to test the compatibility of the students will next be identified, and the results will be recorded.

4.4 Analysis

Data on project planning and estimation obtained will next be analysed. While, the evaluation from the students on the course performance in delivering the learning objective and achievements.

5 Conclusion and Future Works

The contribution and future research are expected to result in finding the effectiveness of applying an Agile software methodology approach in software engineering course which focuses on the team software development project for

undergraduate studies. It aims to analyse the software development teams' abilities to adopt Scrum concepts and principles in an Agile software development environment.

Furthermore, this study will help to gather the teams' viewpoint regarding the importance of Agile and TSPi practices in handling real software projects.The result should be able to show the ability to estimate and planning of user stories into Sprints, furthermore, able to define accurate iteration plans.

This particularly essential for the students to understand better both the software methodologies approach which emphasis on "learning by doing" approach and prepare them for a future career as a software engineer. This research will undergo the duration of two academic terms to be implemented and observed, where both of the traditional and Agile software methodologies will be applied in two different batches of Software Engineering Laboratory (SELab) course students. This comparison of both methods will help to identify which method is suitable for the development of software project in an institution of educational environment and at the same time to improve the SELab syllabus and learning outcomes.

Acknowledgment

This research is sponsored by UNIMAS research grant, F08/SoTL/1471/2016.

References

Abdulwahab, L., Abdalla, A. A., Galadanci, B. S., Algudah, M., & Murtala, M. (2015). Agile Methods for Software Engineering Students Project : A Proposed Hybrid Methodology. In *Proceedings of The Third International Conference on Digital Enterprise and Information Systems* (pp. 63–69). Shenzhen, China.

Cannon, R., Hilburn, T. B., & Diaz-Herrera, J. (2002). Teaching a Software Project Course Using the Team Software Process. *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, (March), 369–370. https://doi.org/10.1145/563340.563486

Chick, T. A., Cannon, R., McHale, J., Nichols, W., Pomeroy-Huff, M., Welch, J., & Willett, A. (2009). Team Software Process (TSP) Coach Mentoring Program Guidebook. *Sei.Cmu.Edu*, (August). Retrieved from http://www.sei.cmu.edu/library/abstracts/reports/09sr009.cfm?WT.DCSext.abstractsource=BrowseStacks

Com, V. (2015). 9th Anual State of Agile Survey.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). *A Lightweight Guide to the Theory and Practice of Scrum Version 2.0*. InfoQ.

Despa, M. (2014). Comparative study on software development methodologies. *Database Systems Journal*, *5*(3), 37–56. Retrieved from http://dbjournal.ro/archive/17/17_4.pdf

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development*, *9*(August), 28–35. https://doi.org/10.1177/004057368303900411

Humphrey, W. S. (2000). *Introduction to the Team Software Process.* Reading, Massachusetts.: Addison Wesley.

Kruchten, P. (2003). *The Rational Unified Process An Introduction* (3rd ed.). Addison-Wesley Professional.

Larman, C. (2002). Applying UML and Patterns - An Introduction to OO Analysis and Design and the Unified Process, 736.

Mahni, V. (2015). Scrum in software engineering courses: An outline of the literature. *Global Journal of Engineering Education*, *17*(2), 77–83.

Moniruzzaman, A. B. M., & Hossain, D. S. A. (2013). Comparative Study on Agile software development methodologies. *arXiv Preprint arXiv:1307.3356*.

Over, J. W. (2000). Instructor 's Guide for Introduction to the Team Software Process. Retrieved from http://www.uml.org.cn/softwareprocess/pdf/tspi.pdf

PricewaterhouseCoopers. (2014). *Adopting an Agile methodology: Requirements - gathering and delivery*.

Qureshi, M. R. J., Alshamat, S. A., & Sabir, F. (2014). Significance of The Teamwork in Agile Software Engineering.

Sci.Int.(Lahore), *26*(1), 117–120.

Reichlmayr, T. (2003). The Agile Approach In An Undergraduate Software Engineering Course Project. *ASEE/IEEE Frontiers in Education*.

Stewart, J. R., & Agah, A. (2012). Teaching a software engineering course on developing video games : a Unified Process versus Extreme Programming, *10*(1), 6–12.

Sureshchandra, K., & Shrinivasavadhani, J. (2008). Moving from waterfall to agile. In *Proceedings - Agile 2008 Conference* (pp. 97–101). https://doi.org/10.1109/Agile.2008.49

Werner, L., Arcamone, D., & Ross, B. (2012). Using Scrum In A Quarter-Length Undergraduate Software Engineering Course. In *Consortium for Computing Sciences in Colleges* (pp. 140–150).

Zorzo, S. D., De Ponte, L., & Lucrédio, D. (2013). Using scrum to teach software engineering: A case study. *Proceedings - Frontiers in Education Conference, FIE*, 455–461. https://doi.org/10.1109/FIE.2013.6684866