Teaching Agile Software Development in a Software Engineering Course: An Empirical Study

Alaa Hantoli[1] and Abdel Salam Sayyad[2]
[1] Engineering and Technology Faculty,
Palestine Technical University-Kadoorie, Palestine
a.hantoli@ptuk.edu.ps
[2] Master Program in Software Engineering,
Birzeit University, Palestine
asayyad@birzeit.edu

Abstract: Agile has been successfully adopted by many software companies and it is the most popular methodology for software development in industry nowadays. However, our universities give more attention to teaching Waterfall model in related courses with a bit coverage of Agile main characteristics. In this paper, we work on the setup, execution, and results of teaching a Software Engineering course to undergraduate students with a specific focus on Agile practices, through official re-constructed lectures besides open workshops with a senior engineer from industry to follow up with students projects in parallel. This research will study its impact on students. Also, overcome the potential problems and highlight any raised side effects. In addition to improve the students' technical and social skills, and in spite of other related works, it investigates many factors affected or have been affected by Agile and hold many significant comparisons. The results show the high satisfaction of the students through the experiment, also show a sufficient evidence to conclude that there is a significant difference in the means of improvements between the experimental and control groups in understanding software engineering and Agile methodology in specific.

Keywords: *Agile Software Development, Scrum, Software Engineering.*

Introduction:

The term Agile Software Development (ASD) has evolved opposite of the plan-centric development, its agility come from being designed to accelerate the software delivery, and to be responsive and to the rapidly changing requirements and integrate them to the product, and increase the productivity as well as ensure the software high quality and minimal development overhead [1].

1.1 Introduction and Motivation:

Agile becomes main stream and it is the public approach in software development nowadays [6]. It was evolved and applied by industry [5].

It becomes critical to teach the process of development in universities, which is a risk-free academic environment allow a fully experiencing of Agile, in order to improve the students' technical and social skills effectively, and build an Agile mindset. Also, helps to investigate whatever factors we intend to study in this convenient educational environment [2].

1.2 Research Objectives and Problem Statement:

- This research will discuss whether we can significantly improve the quality of teaching output by applying the proposed pedagogical model?
- How can course materials be designed to help students stay organized and succeeded? And how to manage such class effectively?
- Also hold a comparison between Waterfall Model and Agile Model against specific criteria.
- Are students satisfied of learning the SWE course with a focus on agile and its tools?

1.3 Overview of this paper:

The paper is organized after the introduction section as follows:
Section 2 gives a background about ASD and Scrum. Besides the pertinent literature.
Section 3 outlines the research methodology to collect and analyze the data.
Section 4 shows results and discussion, followed by a conclusion, recommendation and threats to validity.

Background and Literature Review:

1.4 Definition and Background:

Agile software development (ASD). It has been formally introduced by a group of software practitioners and consultants in 2001 in the "agile manifesto", which establishes four fundamental: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [1]. In addition to twelve principles, Agile people follow. Agile classified as a lightweight methodology in where it is incremental, iterative, cooperative, straightforward and adaptive [12]. Also minimizes the risk by centering on short iterations [4].
The different ASD Methods are: XP (Beck 1999b), scrum (Schwaber 1995; Schwaber and Beedle 2002), crystal methodologies (Cockburn 2002a), feature driven development (Palmer and Felsing 2002), the rational unified process (Kruchten 1999), dynamic systems development (Stapleton 1997), adaptive software development (Highsmith 2000), open source development (O'Reilly 1999), Agile modeling (Ambler 2002a) and pragmatic programming (Hunt 2000).

Scrum Methodology. The most important approaches are: Scrum, extreme, adaptive and dynamic. Of these, the most used is Scrum. Scrum is simply an agile, lightweight process for controlling development in rapidly changing environment, as customers change their mind about the product and the development challenges are unpredictable by their nature. So Scrum focuses on maximizing the ability of the team to quickly deliver in response to emerging requirements [8].
This model is built on three major components [4] [12]:
Scrum Roles. Three not boundaries-crossed roles: Scrum Master, Scrum team and product owner.
Scrum Artifacts. .Product backlog, sprint backlog and burndown chart.
Scrum Process activities. Kick-off, sprint planning meeting, sprint, daily Scrum and sprint review meeting (see fig. 1).

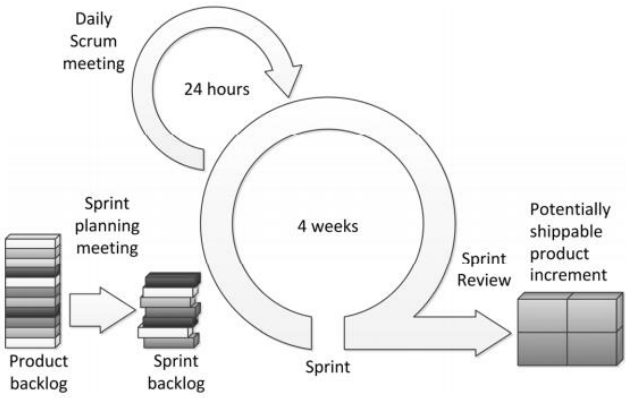


Fig. 1. Scrum Process Loop (Andreas Schroeder 2012).

Why Scrum. There is a revolutionary transmit from prescriptive approaches to empirical processes, which maximizes the results, managers' works with team, creativity and collaboration are the stamp of this method [22,23]. It overcomes any unexpected changes [12]. However, there is no method is a "silver bullet", savvy is where to use each.

Scrum and XP are mostly applicable in studies, having common structures, roles, and values. But fine differences, for example XP pays more attention on engineering practices [5], which is not our interest here. Whereas Scrum focuses on management practices, in addition to its spread in industry, it is easy to comprehend, accessible and effective to introduce agility to undergraduates [2]. And it worth highlighting some orientations weaving that two or more ASD approaches together were acceptable and productive [6,24].

1.5 Literature Review:

Literature shows no related studies prior to 2001, which were published in Extreme Programming and Agile Processes [14,15]. In few universities, ASD laboratories and centers appeared. Most authors from North America and Europe. However, in Arab World there was single Jordanian research [9]. An early literature recommends two points, all subsequent papers confirm: practices should be one assignment consists of all SWE aspects and creating heterogeneous groups of student capabilities and interests, which leads to homogeneity among groups [14].

Agile was evolved and utilized widespread in industry, so its teaching in the academia is just a natural response. It deals with human aspects according to agile manifesto. It is naturally taught in a teamwork-oriented environment. It promotes diversity. Also supports learning processes. Furthermore, it develops mind habits. Agile emphasizes

management skills. It adheres ethical norms as well. Lastly, Agile provides a single all-inclusive teaching framework for SWE [7].

In some lab experiment, authors set two issues preventive to success: Firstly, instructors introduce advanced research topics. Secondly, too much time consumed on the functionality of the built software over approach [2].

Teachers should choose the tools to be correctly function and easy to use, to intensify students' efforts on Agile practices [17,18]. A rising tendency was the simulation games -like Lego4Scrum and card games, to facilitate the transition from the theory to practice [1], and PlanningPoker for stories-points estimations [2,20].

A systematic literature review, written based on experience and empirical studies identified 17 practices: Customer involvement, User stories, Prototyping, Testing and others. And authors lastly point a need for extra attention and empirical results in teaching Agile [10].

Research Methodology:

This experiment focus on undergraduate-level teaching, held over the second semester of year 2018/2019 at Palestine Technical University.

1.6 Goal Definition:

Analyze the final product including the bug rate and the percentage of implemented features, for the purpose of evaluating the impact of adapting the Agile Software Development Methodology in teaching, with respect to their effectiveness, from the point of view of the researcher, in the context of 3rd year SWE course students formed in teams of 4-5 members working on complicated problems. The study is conducted as a blocked subject-object study, since it involves many subjects and more than one requirements document.

1.7 Planning:

Context Selection. The context is a SWE course, with 3rd-year students, who passed the prerequisite courses, also their age is a positive factor [1]. The experiment is run off-line. It considered as general research case that aims to compare two methodologies in general, and from a research perspective. Where the subjects in both tracks have no prior experiences.

Hypotheses Formulation. We would like to compare effectiveness by means of both quality and completeness of the product when using two methods, Agile and Waterfall. Where Completeness will indicates the time needed, and Rank indicates the level of students in understanding and practicing Agile.

a- So the first Factor is Software Development Methodology:

Null hypothesis, H0: Agile needs the same time-to-market that waterfall needs.
H0: µC Waterfall = µC Agile

Alternative hypothesis, H1: Agile optimizes the time-to-market comparing to waterfall.
H1: µC Waterfall < µC Agile

Null hypothesis, H0: Agile methodology produces the same number of bugs as waterfall methodology.
H0: µF Waterfall = µF Agile

Alternative hypothesis, H1: Agile methodology produces less number of bugs than waterfall methodology.
H1: µF Waterfall > µF Agile

b- And the second Factor is Pedagogical model in teaching

Null hypothesis, H0: Students who applied the agile process have the same understanding of it as the students who applied the waterfall process.
H0 : µR Old = µR Agile

Alternative hypothesis, H1: Students who applied the agile process have better understanding of it than the students who applied the waterfall process.
H1 : µR Old < µR Agile

Measures needed. Faults/KLOC, the number of faults divided by the number of lines of code. Completeness, the number of features completed divided by the number of features required. Students Understanding: a pre and post quiz analyzed by T-Test or Mann-Whitney.

Experiment design. With a view to generalize the results, subjects will be chosen using simple random sampling as a probability sampling technique, where they are selected randomly and optionally from the SWE course students at the university. The size of the sample is: 8 groups with a total of 38 student. In addition to randomization, if any group out of the 3rd year engaged the experiment we do blocking on prior experience by a pre-test, and on differences between problems they worked on, in terms of complexity and context. Moreover the experiment uses a balanced design, which means that we have the same number of subjects per treatment in the design. The instruments for performing and monitoring the experiment are: objects represented by real products and its requirements, guidelines and principles for the development methodologies. And to guarantee rightful comparison, we concerned comparable support of available resources and training for the two methods for the teams. Lastly, measurements instrumentation conducted via data collection in manual and online forms and interviews and some reads from some tools used.

1.8 Operation:

We select and inform participants about the experiment nature, and prepare material i.e. forms and tools, then ensure the infrastructure needed.

The course is designed to fulfil the Pyramid of Agile Competences described in [6]. Each group have name and logo to pace up work by adding a kind of competition. The project is implemented during seven 2-weeks sprints. Lectures differ between discussion lectures, gaming, and technical workshops.

Several *tools* have been integrated, mainly which are easy, less-cost, compatible and interrelated tools to use from the same screen. We use “Trello” for Project Management, it is a platform supports the transparency idea in Scrum. Through it teams organize the backlog, and tasks of the current sprint and show their progress and communicate. It is compatible and powered up by other essential applications. We also use “Bitbucket” for source code management, “Catme” for team work assessment, “Webinar” for online meetings with lecturer, and “Moodle” and online forms for quizzes and questionnaires.

Roles assigned with their self-explanatory names [1]. Scrum Master initially assigned to the teacher, then a student chosen according the sprint details [5], or voted [6]. Scrum Coach was external senior developer. Product owner assigned to students. Others became developers and testers. Finally teacher’s role was expressing the “guide on the side, not a sage on the stage” principle [14].

A formative and summative evaluation assess students understanding, skills, projects and satisfaction. Also monitoring by observations and evaluations, written down into dedicated tables.

Results and Discussion:

Projects were of the same context and level of complexity, but different products and details. Below in Table 1 is a summary of teams and projects.

Table 1. Summary of teams and projects.

type\method	Agile	Waterfall	Project Context
Desktop	First team "Hash the Dash"	Fifth Team	Dental Clinic Systems
	Second team “Alpha Team”	Sixth Team	Medical Laboratory Systems
	Third Team “Rainbow Team”	Seventh Team	Cars Insurance Systems
Web	Fourth Team "*CCCare Team*"	Eighth Team	Child Care Center Systems

1.9 Evaluation of Learning Outcomes:

For the Pedagogical model factor, the difference between the pre and post quizzes results were taken to find the improvements, which were about 44% for the Agile students against roughly 28% for the traditional on SPSS software.

Table 2. Shapiro–Wilk test of Normality.

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
improvement	.151	38	.028	.957	38	.152

By Shapiro–Wilk test showed in Table 2, we find p-value=0.152 which is greater than α=0.05, then the null hypothesis that the data came from a normally distributed population can't be rejected, and the data is asymptotically normal. So we were able to use the T-Test which is a parametric test Table 3 shows.

Table 3. Group Statistics, T-Test.

	Method	N	Mean	Std. Deviation	Std.Error Mean
improvement	Agile	18	8.94	2.461	.580
	Waterfall	20	5.50	3.502	.783

According to Levene's Test for Equality of Variances showed in Table 4, we find a homogeneity of variance between the two samples since sig. = 0.130. Then we read our values from the first row. Since p-value = 0.001 which is less than α=0.05, we reject the related null hypothesis, which means at level of significant α=0.05 the data give us a sufficient evidence to conclude that there is a significant difference between the means of the experimental and control groups, which are respectively 8.94 and 5.50.

Table 4. Independent Samples Test, T-Test.

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Improvement	Equal variances assumed	2.404	.130	3.471	36	.001	3.444	.992	1.432	5.457
	Equal variances not assumed			3.535	34.097	.001	3.444	.974	1.464	5.425

At projects discussion, Agile groups showed a better understanding of the materials. In spite of control groups who have cut corners to finish their products with obvious negligence of methodology principles. That is strongly support our results in this most important part of the research.

1.10 Time to Market:

For the methodology factor, completeness indicated the time needed, in term of features completed, divided by features required. The results are stated in Table 5.

Table 5. Projects Completeness.

Project Context	Team	No. feat/ req. feat	No. feat/ req. feat	Team
Dental Clinic Systems	A1	26/28	24/28	W1
Medical Laboratory Systems	A2	36/37	30/34	W2
Cars Insurance Systems	A3	31/36	27/36	W3
Child Care Center Systems	A4	29/29	26/29	W4

The feature is formed in user stories and tasks. It considered completed, if its main user stories are done (coded, reviewed and integrated). The data in Table 5, shows the results for both methods, and it let us reject the null hypothesis.

We easily find the time rhythm in sprints design, which helps Agile teams to be more organized. Also their definition of done is clear through their design of Trello lists, and backlogs content, and tasks cards. Where waterfall teams took a long time in requirement elicitation and SRS preparation, then they had dived in coding, and put off testing, where their colleagues in Scrum teams building their low and high fidelity prototypes, involving their customer, prioritizing their requirements and refine the next sprint planning.

The agile teams were almost self-organized [3], but roles not repeatedly rotated which dissipate the students' efforts and times. Indeed we take into consideration the individuals' abilities and tendencies, and let all members be conscious of all practices and activities even by observation. Another reason is the fact the developer in market does not do a role each morning.

1.11 Product Quality:

For the second set of hypotheses of the methodology factor, the product quality is indicated by bugs divided by lines of code. The bugs included the interface and usability problems. Table 6 shows that rates in general is lower for agile teams, so we can reject the null hypothesis.

Table 6. Products Quality.

Team	Bugs/ KLOC	Bugs/KLOC		Bugs/ KLOC	Team
A1	95/4.2	22.6	24.6	105/4.3	W1
A2	120/5.7	21	18.5	92/5	W2
A3	84/7.7	11	38	243/6.4	W3
A4	70/4.7	14.9	27.3	120/4.4	W4

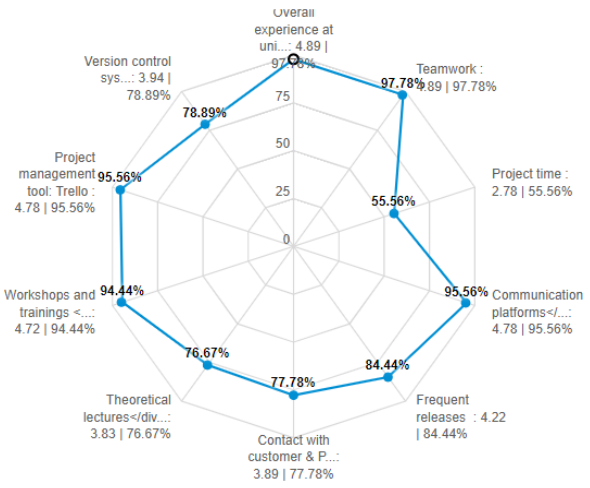
We find less bugs per KLOC at the left part of the table, except pair 2, perhaps of team's less code, and features, and sometimes "W2"team had customer meetings as needed.

Agile team of third pair have distinct result, despite the biggest code and features of an accounting system; that's of frequent releases and testing are very helpful in quality improvement.

1.12 Students Satisfaction:

A questionnaire of fifteen statements on 5-Point Likert Scale. Overall feedback was positive (see Fig. 2). The second part asks if they recommend this course to their colleagues. All answers were "Yes". The results in details includes more graphs and tables [27].

Fig. 2. Top/Bottom five results of the fifteen statements, with its percentage of satisfaction.



With overall satisfaction of 87.6%, results points a sense that the design and content of the course are mature enough. However some negative feedback received on time offered compared to work required.

We satisfied of the experiment significant results, the good influence on the students, and the high quality products delivered. It is also worth mentioning that two groups work on their graduation project using agile methodology.

Conclusion and Future Work:

1.13 Conclusion:

Agile is the most used methodology in software companies, so it is clear that Agile more important than just defined with some of its characteristics in the SWE course in universities, where it is more effective and less risky to get knowledge and behaviors in academic environment, paying attention to the teaching Agile itself should be Agile teaching. The teacher may act negatively if not skilled enough, and external coach helps monitoring the process, that gives more reliability and credibility to results.

Evaluations of learning outcomes and students' satisfaction demonstrate that the course concepts well received while having fun. An important contributions of this paper to light a methodology for teaching the SWE as an agile project based course, and to present different techniques to enhance this methodology.

1.14 Future Work and Recommendation:

Future Work. To accommodate more students, all groups might be given same project, to easily manage and follow them up. Or having a large project distributed between groups, which is closer to reality i.e. outsourcing.

Next times, we will focus more on games and non-functional requirements. And may develop an introduction course to Agile, dedicated to 2nd-year students, so in 3rd-year helps to exploit the time in a better manner.

Recommendations. Other courses like: OOP and Data Structure may adopt agile project, instead of isolating SWE teaching in separate course. Also we suggest to engage master students –who are usually professionals in IT sector- to the experiment through the Construction Course, or elective course.

1.15 Threats to Validity:

For external validity, we are not able to include many universities to have a random sample, so it might not produce generalizable results, due to under-representation of sample subgroups. Another limitation in conclusion validity, we were limited to a certain subjects. We had two choices, to repeat the experiment on other years, which duplicated time needed, but we had done a trial before, it was somehow useful guiding us for a better execution. Or to include the CS students but we didn't, to guarantee the homogeneity. And it still possible later.

Also, it is a human-oriented experiment, as regards internal validity, this implies limitation to the study control, since students have distinct capabilities, skills and interests, which act as independent variable. So we try to establish homogeneous teams, even they were internally heterogeneous which reduces limitation effect.

Regarding the construct validity, we should track students later; since it needs much effort to report they learned such valuable thing, without seeing them at work. Furthermore, it is better to reject or accept null hypothesis of time-to-market, by finding the velocity of each team, velocity is the result of the division of Agile story points delivered by the number of sprints. But it has a poor fit for waterfall strategy of development.

ACKNOWLEDGMENTS:

This research was financially supported by Palestine Technical University - Kadoorie. We thank our colleagues from the Scientific Research Deanship. We thank the CSE department for assistance in conducting such experiment. We would also like to show our gratitude to the external coach from the private sector; Dimensions, he greatly improved the follow-up of projects development process.

References:

1. Devedžić, V., & Milenković, S. R. (2011). Teaching Agile Software Development: A Case Study. IEEE Transactions on Education, 54(2), 273–278.
2. Schroeder, A., Klarl, A., Mayer, P., & Kroiss, C. (2012). Teaching agile software development through lab courses. IEEE Global Engineering Education Conference, EDUCON, 1–10.
3. Rodriguez, G., Soria, Á., & Campo, M. (2015). Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. Computer Applications in Engineering Education, 23(1), 147–156.
4. Cervone, H. F. (2011). Understanding agile project management methods using Scrum. OCLC Systems & Services: International Digital Library Perspectives, 27(1), 18–22.
5. Johnson, A. M. A. (2017). Teaching Agile Methods to Software Engineering Professionals: 10~Years, 1000 Release Plans.

6. Kropp, M., & Meier, A. (2013). Teaching agile software development at university level: Values, management, and craftsmanship. Software Engineering Education Conference, Proceedings, (November 2014), 179–188.
7. Hazzan, O., & Dubinsky, Y. (2007). Why Software Engineering Programs Should Teach Agile Software Development. Software Engineering Notes, 32(2), 1–3.
8. Campbell, J., Kurkovsky, S., Liew, C. W., & Tafliovich, A. (2016). Scrum and Agile Methods in Software Engineering Courses. Proceedings of the 47th ACM Technical Symposium on Computing Science Education – SIGCSE '16, 319–320.
9. El-Khalili, N. H. (2013). Teaching Agile Software Engineering Using Problem-Based Learning. International Journal of Information and Communication Technology Education, 9(3), 1–12.
10. Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior, 51, 915–929.
11. Beck K. (2001). Manifesto for agile software development. [Online]. Available: http://agilemanifesto.org/
12. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. Espoo, Finland: Technical Research Centre of Finland, VTT Publications, 112.
13. Scharff C. (2011). Guiding global software development projects using scrum and agile with quality assurance. Software Engineering Education and Training Conference, IEEE.
14. Del Bianco V. & Sassaroli G. (2003). Agile Teaching of an Agile Software Process. Extreme Programming and Agile Processes in Software Engineering 4th International Conference, XP. Genova, Italy, 417-420.
15. Reichlmayr, T. (2003). The agile approach in an undergraduate software engineering course project. Proceedings - Frontiers in Education Conference, FIE,3, S2C13-S2C18.
16. IEEE. (2009). New Standards Committee (NesCom) recommendations, IEEE-SA Standards Board. [Online]. Available: http://standards.ieee.org/
17. Sato D. T., Corbucci H., & Bravo M. V. (2008). Coding dojo: An environment for learning and sharing agile practices, IEEE. 459–464.
18. CodinDojo. [Online]. Available: http://www.codingdojo.com/
19. Lego4Scrum. [Online]. Available: https://www.lego4scrum.com/
20. Tastycupakes. [Online]. Available: tastycupcakes.org
21. Catme. [Online]. Available: www.catme.org
22. Fowler, M. (ThoughtWorks). (2009). Flaccid Scrum, 2. [Online]. Available: https://martinfowler.com/bliki/FlaccidScrum.html
23. Schwaber, K. (2010). Waterfall, Lean/Kanban, and Scrum, 5. [Online]. available:https://kenschwaber.wordpress.com/2010/06/10/waterfall-leankanbanand-scrum-2/
24. Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel Hybrid Model: Integrating Scrum and XP. International Journal of Information Technology and Computer Science, 4(6),39–44.
25. Trello. [Online]. Available: https://blog.trello.com/
26. Castronova, E. (2005). Synthetic worlds: The business and culture of online games. University of Chicago Press, United States.
27. Satisfaction Questionnaire – questions, results and analysis. Available: https://www.questionpro.com/t/ZRisXHZG2zpYD