International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I12P210

Volume 71 Issue 12, 90-97, December 2023
© 2023 Seventh Sense Research Group®

Original Article

Software Engineering Learning Model Framework using Agile Techniques

Davit Sanpote[1], Worrakit Sanpote[2]

[1,2]School of Information and Communication Technology, University of Phayao, Thailand

[1]Corresponding Author : davit.sa@up.ac.th

Received: 01 August 2023 Revised: 12 October 2023 Accepted: 25 October 2023 Published: 06 December 2023

Abstract *- Agile testing contains significant factors of success in implementing software products with good productivity. This is an important skill for software developers and software testers. Moreover, agile testing also needs to be taught appropriately to students studying in the field area of computers and technology. This research will describe some drawbacks and benefits aspects in a case study from an undergraduate Software Engineer at the University of Phayao (SEUP). At the end of the paper will be a discussion in terms of successful teaching methods of agile testing in testing classes.*

Keywords *- Agile testing process, Agile testing quadrant, Unit Test, Component test, Software engineer curriculum, teaching unit.*

1. Introduction

In software-industrial, including small scale and large scale. Control of the software development process is an important part before the team releases a final product to customers. In the basics of developing the software product, the software requirement is another significant key between the development team in software companies and users. Nowadays, the agile software development process has become a more necessary method. Agile bring the development team, and user can go along together by releasing progress frequently. From the statement mentioned above, software testing is a necessary part of the software development process that can blend in an agile software development process as a parallel called" Agile Testing". This research focused on finding teaching factors in software testing subjects with the software engineering curriculum at the University of Phayao (SEUP) [12**]**, Thailand.

The knowledge gap will be mentioned in a new model of teaching framework, which aligns with the improvement of software testing skills set in the literature review and discussion parts. The software testing has been divided into two models, the traditional (PC1.0**)** model and the redesigned model (PC1.1-1.2**)** based on the teaching framework [**7**], which will be described in the related work section. The software testing subject has broken down assignments into four tasks: test case design, unit test, automated test, and test report. These four tasks are included in the mini-project assignment. In addition, the limitation of time is an evaluation method for students. In the research and methodology section, there are three categories to separate student capability: "on-time complete", "On time not complete", and "not on time. Moreover, the research results are beneficial to improve and update teaching methods in the future.

2. Literature Review

2.1. Model for Agile Teaching and Software Testing Concept

Recently, the agile method is another important basic software development process in various scales of tech companies. At the University level, teaching in the software development process also includes agile concepts. Moreover, in terms of teaching software testing processes in an agile context, this paper needs to consider active teaching. An active teaching method focuses on the learning center [7][8]. The most important activities are discussion, emphasizing, and group work. In SEUP, similar activities for students to play different roles in tech teams, such as product owner can be another tester at the same. Another highlight of the teaching concept is reinforcement in curriculum practice.

A real project from a real stakeholder is a necessary experience for undergraduate students. The real project will provide a powerful requirement to build the right Software Requirement Specs (SRS) that can be used for project closure with a User Acceptance Test (UAT). Another benefit for students to work with a real project is when they face an initial problem, the understanding of software testing principles will be applied throughout the project rapidly. Communication and expression skills are also included in this part [4]. Furthermore, the evaluation result by collecting

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

student feedback is also important. To find out the success and failure factors to improve the teaching plan. The voice of the customer needs to be noted in an evaluation process, such as a short questionnaire [3]. Last, the significant model guidelines to build a course syllabus to achieve the learning objective of Bloom's Taxonomy [1]. The SEUP concentrated on the level of understanding to apply for bachelor's degree achievement.

2.2. Building Syllabus of Software Testing using agile Method Software Testing

Concepts contain a variety of fundamental and professional experience skill sets. In teaching software testing as an appropriate concept for undergraduate students, necessary guidelines and methods to design a course syllabus must be considered for the software engineering curriculum. There are two major methods for designing the SEUP curriculum, which are ACM/IEEE and SWEBOK. The ACM /IEEE guide was used for the basic syllabus in computing in the mid-2014 version.

ACM/IEEE was a collaboration between ACM and IEEE to build a standard of the first computer science 2001,2008 and 2013, accordingly [6].

SWEBOK is a popular guideline method for the software engineering curriculum since SEUP has a multidisciplinary and academic skill set. SWEBOK has evolved to ISO/IEC 19759 standards that describe many specific and important SE curricula [2].

In the next step, after completing a standard teaching plan, a practical task will be created along with a plan. The teaching strategy for each stage needs to locate a result outcome to evaluate a student's learning. There is previous research that produced a generic construction of a teaching unit [8], as shown in Table 1.

The SEUP has modified more details to evaluate teaching factors in Table 2.

Table 1. Standard teaching framework

<table>
<tr><th colspan="2">Teaching Unit</th></tr>
<tr><td colspan="2"><b>Prerequisites</b></td></tr>
<tr><td colspan="2">Previous foundation subject's unit. These units are related to the course syllabus and are provided for students to attend.</td></tr>
<tr><td colspan="2"><b>Guiding Questions</b></td></tr>
<tr><td colspan="2">This is a discussion question to ask students at the beginning of each class.</td></tr>
<tr><td colspan="2"><b>Programmatic Content (PC)</b></td></tr>
<tr><td colspan="2">The top view of competencies plan for a whole teaching period. A learning topic will be added to this session.</td></tr>
<tr><td><b>Expected Result</b></td><td><b>Learning Level</b></td></tr>
<tr><td>The outcome that students will be able to achieve after finishing a unit.</td><td>According to expected results can be described into certain levels, such as cognitive ability and Bloom's Taxonomy.</td></tr>
</table>

Table 2. Software testing teaching framework for software engineering students, University of Phayao, Thailand

<table>
<tr><th colspan="2">Teaching Unit</th></tr>
<tr><td colspan="2"><b>Prerequisites</b></td></tr>
<tr><td colspan="2">Students have basic knowledge of structure programming, functional programming, and OOP programming. Students have a good understanding of the software requirement process. Students get used to the user's behavior related to business growth.</td></tr>
<tr><td colspan="2"><b>Programmatic Content (PC1.1)</b></td></tr>
<tr><td colspan="2">1. Conceptual testing process and agile testing
2. Type of software testing
3. Test scenario
4. White box and black box testing</td></tr>
<tr><td><b>Expected Result</b></td><td><b>Learning Level</b></td></tr>
<tr><td>The students understand the basics of software testing concepts and can apply them to test case scenario design by using white box and Blackbox testing techniques.</td><td>Understand/apply/build/analyze.</td></tr>
<tr><td colspan="2"><b>Programmatic Content (PC1.2)</b></td></tr>
<tr><td colspan="2">1. Unit test
2. Automated test
3. Standard of software testing document and report
4. CI/CD</td></tr>
<tr><td><b>Expected Result</b></td><td><b>Learning Level</b></td></tr>
<tr><td>The student can apply knowledge on a real project and report to users who give a software requirement appropriately.</td><td>Understand/apply/analyze</td></tr>
</table>

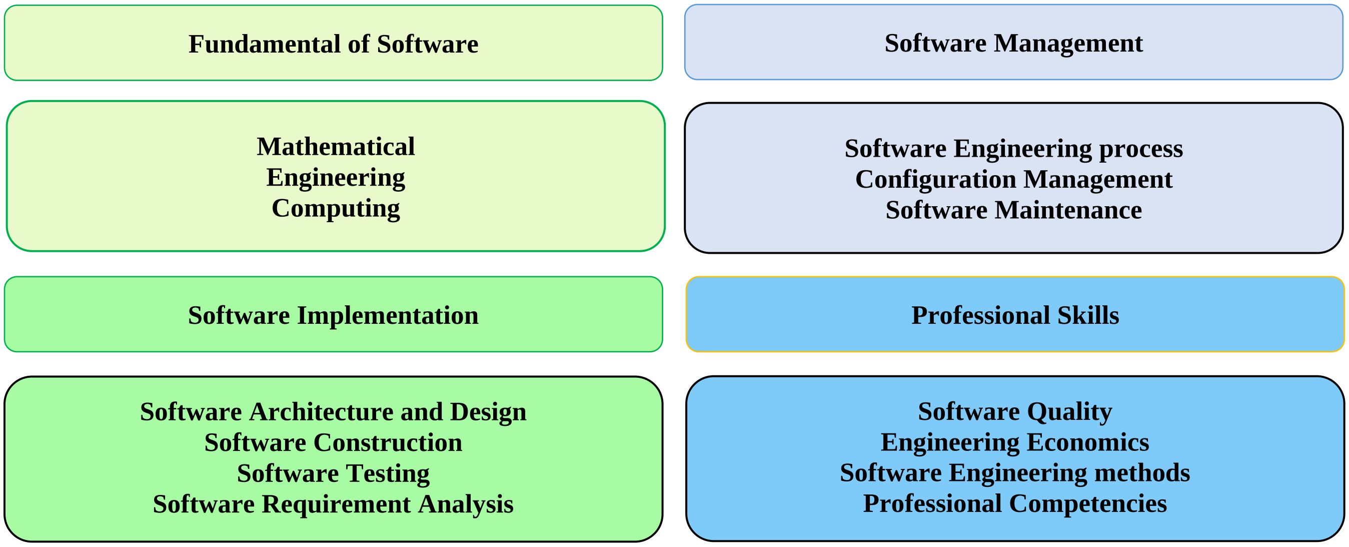

Fig. 1 An overview of four-dimension significant software engineering skills

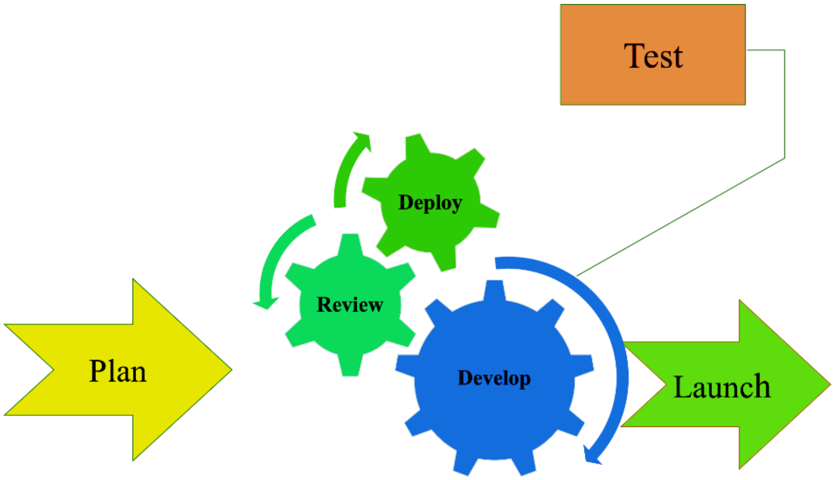

Fig. 2 An overview of the agile testing concept

According to Table 2, the software testing structure is divided into two parts. Part one is in the midterm, which contains basic knowledge after receiving software requirement specs. After that, the student will analyze the requirement into an initial test case version. The first test case is used to implement a basic prototype to recheck once again with the user. Part two contains technical terms using testing automation tools and low-level tests with unit tests. Lastly, students will collect all important results to summarize into a final report. Both parts follow basic concepts of the agile testing process with a real-world project.

2.3. Software Syllabus of Software Testing using Agile Method Software Testing

Software Engineering (SE) curriculum combines a multidisciplinary capability to design course structures for education in university. Software engineering body of knowledge (SWEBOK) represents the necessary skills for software engineering. SWEBOK provided 15 important significant skills, as shown in Figure 1.

Software testing is located in software development, an important part of SE undergraduate students. SE curriculum in the University of Phayao was designed based on a backbone of SWEBOK and ACM Curricula guidelines. In addition, this research objective also focuses on the concept of teaching agile testing that is based on student knowledge improvement [11].

However, some results from previous research mentioned the advantages of knowledge improvement for undergraduate student success in SE major by facing the right software market [2], which is interesting to point to future study.

2.4. Agile Testing in Success Factor

Agile testing is a similar concept of agile development in the software development life cycle (SDLC), which focuses on the delivery quality of software in a short period of time, as shown in Figure 2.

Table 3. Comparison information between traditional and agile teaching frameworks

Traditional Testing	Agile Testing
Team works independently	Recurring thought implementation process
Testers are not included in a part of the requirement analysis	Testers are a significant position for the requirement phase.
Time consumption will be more in the development phase	Time consumption in development is less than in the testing part.

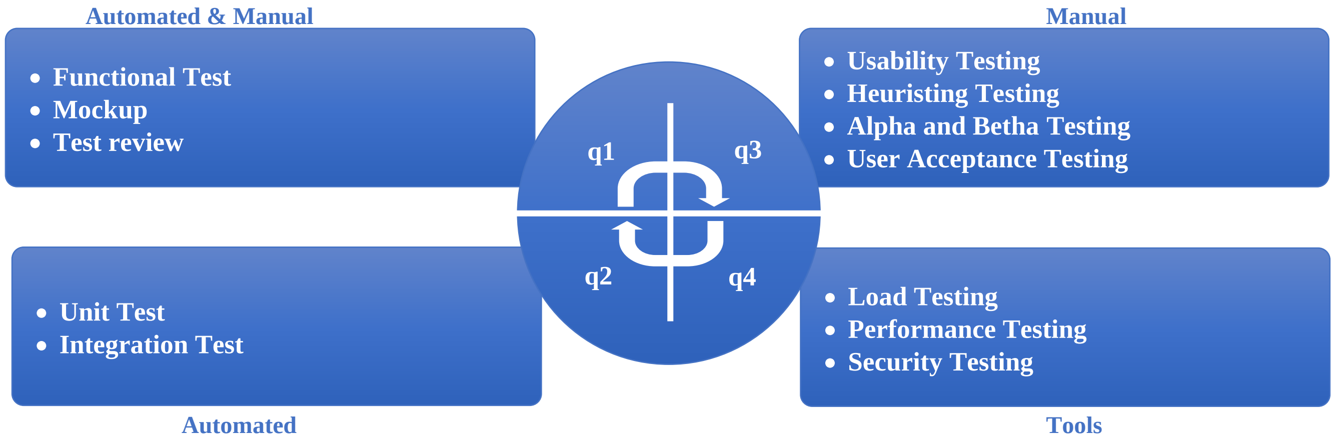

Fig. 3 Agile testing quadrants aspects

Agile testing has been selected to explain to students in testing class. In terms of understanding, the difference between traditional testing and Agile testing [10] is shown in Table 3. According to the comparison table, the main understanding concept is indicated at number one. Testing processes can be combined through the development process and are done side by side.

The SEUP used the concept of Brian Marick, who introduced the four Agile testing quadrants, combining two aspects of testing types as shown in Figure 3.

In Quadrant 1 (Q1) at unit level and technology facing that support developer, Q1 is related to unit and automated tests. The SEUP used various Unit testing tools for testing multi-programming languages such as pytest, jasmine, and Junit. In Quadrant 2 (Q2), at the system level and business-facing that support validates product behavior, the functional test indicated in Q2. Test method in Q2 can be both manual and automated. Next, in Quadrant 3 (Q3), the system and user acceptance related to technology facing. In the last quadrant 4 (Q4), operational acceptance is relevant to technology facing. The perspective in Q4 focuses on the performance test, load, stress, and maintainability testing.

The SEUP in testing class has been used by JMeter to evaluate student load tests. From an overall of the four quadrant aspects, all four quadrants have been mentioned and taught in testing class. However, the majority concept of teaching pointed at Q1, Q2, Q3, and Q4, accordingly.

3. Research and Methodology

This research is pointed at two sample groups between teaching students with traditional software testing models and redesign models. The traditional and redesign model frameworks are shown in Table 4 and Table 5 below. The experimental research result from the traditional model will be used to improve the redesigned model. There are two necessary parts that have been modified: Programmatic content and evaluation.

Table 4. Traditional software testing framework model

Teaching Unit: Traditional Model	
Prerequisites	
Students have basic knowledge of structure programming, functional programming, and OOP programming. Students have a good understanding of the software requirement process. Students get used to the user's behavior related to business growth.	
Programmatic Content (PC1.0)	
1. Conceptual of the testing process 2. White box and black box testing 3. Test case scenario 4. Unit test and automate test concept	
Expected Result	Learning Level
The students understand the basics of the software testing concept, and they can apply it to test case scenario design by using techniques of white box and blackbox testing.	Remembering/understanding

Table 5. Redesign software testing framework model

Teaching Unit: Redesign Model
Prerequisites
Students have basic knowledge of structure programming, functional programming and OOP programming. Students have a good understanding of software requirement processes. Students get used to the user's behavior related to business growth. Understand the majority point of the agile testing method process. Good communication skills.
Programmatic Content (PC1.0)
1. understand the concept of testing process and agile testing 2. understand the test scenario by using the concept of white-box and black-box testing 3. Mini project 1 3.1 Analyze requirements from a project of Phayao probation matching system. 3.2 Apply the concept of black box testing to create test data and test case scenarios. 4. Attend an advanced User interface testing course during a testing class present an initial testing progress to stakeholders

<table>
<tr><th colspan="2">Evaluation</th></tr>
<tr><td colspan="2">1. Midterm exam (25%) 2. Quiz 1(10%) 3. Mini Project part 1 (15%)</td></tr>
<tr><th>Expected Result</th><th>Learning Level</th></tr>
<tr><td>The student understands the basics of the software testing concept and agile testing process. They can apply test case scenario design by using the white-box and black-box testing techniques with a real project that is given in a class. Students are able to report test case design to stakeholders appropriately.</td><td>Understanding/apply/analyze</td></tr>
<tr><th colspan="2">Programmatic Content (PC1.2)</th></tr>
<tr><td colspan="2">1. Understand the concept of using unit tests and automated testing
2. Attend an advanced automated testing course during a testing class
3. Standard of software testing document and report
4.CI/CD Present a final testing progress to stakeholders</td></tr>
<tr><th colspan="2">Evaluation</th></tr>
<tr><td colspan="2">1. Final exam (25%) 2.Quiz 2(10%)
3. Mini project part 2 (15%)</td></tr>
<tr><th>Expected Result</th><th>Learning Level</th></tr>
<tr><td>Students understand the concept of manual and automated testing.</td><td>Understand/apply/analyze</td></tr>
</table>

There are some outstanding differences between traditional and redesign frameworks. Firstly, the redesigned model divided criteria into two parts to evaluate student progress at mid and final. Secondly, an advanced course in a current trend related to course design method has occurred in two parts. Next, the mini project during a class has been added up because the student will play a role in the real-world project with a real stakeholder, as shown in Figure 4, during March 2021.

To conclude, the newer design has been improved from previous students' feedback with a main issue in practical terms. In terms of the evaluation process can be seen in Figure 5.

There are two main aspects to evaluate between traditional and redesigned models with the SEUP students in the 2021 software testing class. The testing class has been divided into two parts: PC1.0, PC1.1, and PC1.2. PC1.0 represents a traditional model without adding agile context. PC1.1 and PC1.2 have been redesigned with an agile testing process.

Both contents contain the same mini-project assignment, which is the key evaluation point. However, one difference between traditional mini projects and redesign mini projects was a software requirement. The traditional model's requirement is based on a basic example of an Enterprise Resource Planning (ERP) system. The redesigned model provided a real software requirement in a Phayao probation matching system case study.

Furthermore, the mini-project of testing will be broken down into four tasks, including test case design, unit test, automated test, and testing report, respectively. These four tasks are related to the agile testing quadrant in Q1, Q2, and Q3. The evaluation score will be counted in the percentage of submissions on time and not on time with the completion of the testing process. The Chi-square equation will summarize the two groups above between PC1.0 in the traditional model and PC1.1 to 1.2 in the redesigned model. This evaluation process's result will support this research's aims to find out significant factors of teaching software testing.

4. Result and Discussion

This section represented the significant success of teaching software testing factors in software testing class with a case study of 14 groups from the SEUP 29 students in two different teaching methods: traditional model without agile testing context at PC 1.0 framework and redesign model based on agile testing concept PC 1.1 to 1.2 framework. Another difference is the real software requirement specs are in the redesigned model. Moreover, there are four testing outcomes for evaluation: test case design, unit test, automated test, and test report. The result will be marked within the limit of the due date and out of time.

However, there can be more factors that will be considered in the sections below, according to a result from the 14 groups (29 SEUP Students) submission assignment in testing class. There are three categories' dimensions to evaluate the quality of submission: "On time complete", "On time not complete" and "Not On Time", as shown in Table 6.

Table 6. The total result in terms of "Ontime complete", "Ontime not complete", and "Not Ontime" in software teaching class, University of Phayao

14 Groups in the SEUP 29 students				
Groups		**Ontime Complete**	**Ontime Not Complete**	**Not Ontime**
Traditional	Test case Design	4	7	3
	Unit Test	3	5	6
	Automated Test	4	6	4
	Test Report	5	4	5
Total		16	22	18
Redesign	Test case Design	12	1	1
	Unit Test	10	3	1
	Automated Test	9	4	1
	Test Report	14	0	0
Total		45	8	3

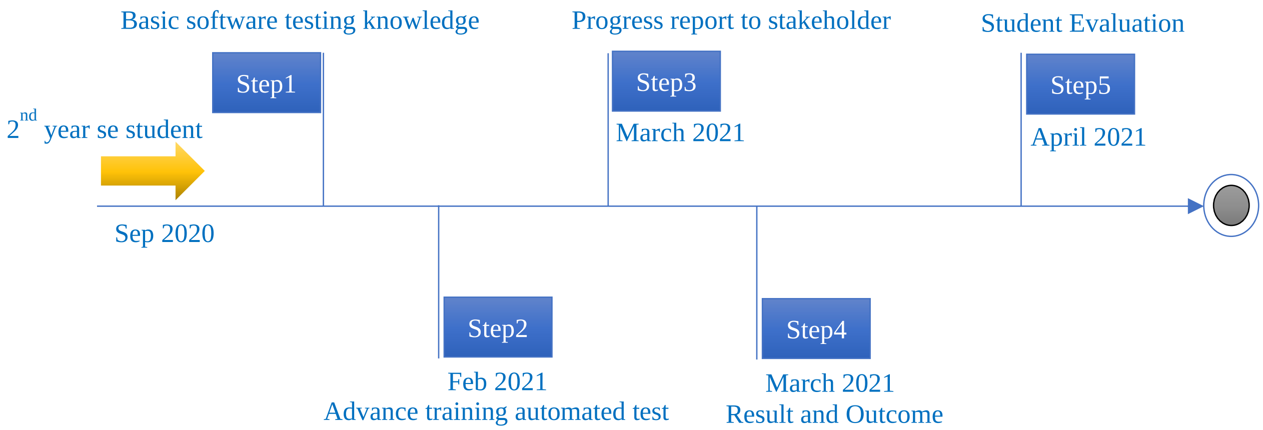

Fig. 4 Software testing teaching road map, University of Phayao, Thailand

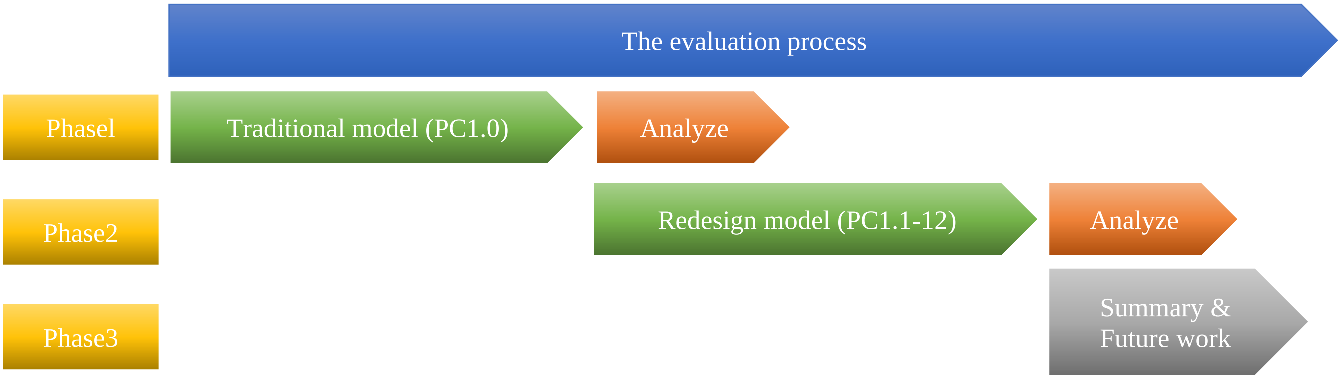

Fig. 5 The evaluation process for the traditional teaching model and redesign model

The assignment tasks have been broken down into four parts. The traditional model with PC1.0 specs showed a significant number of Test reports, which were submitted on time in 5 groups. In the drawback aspect, the minority number submitted not on time is located on Unit test tasks with 6 groups. This explanation can be seen in Figure 6.

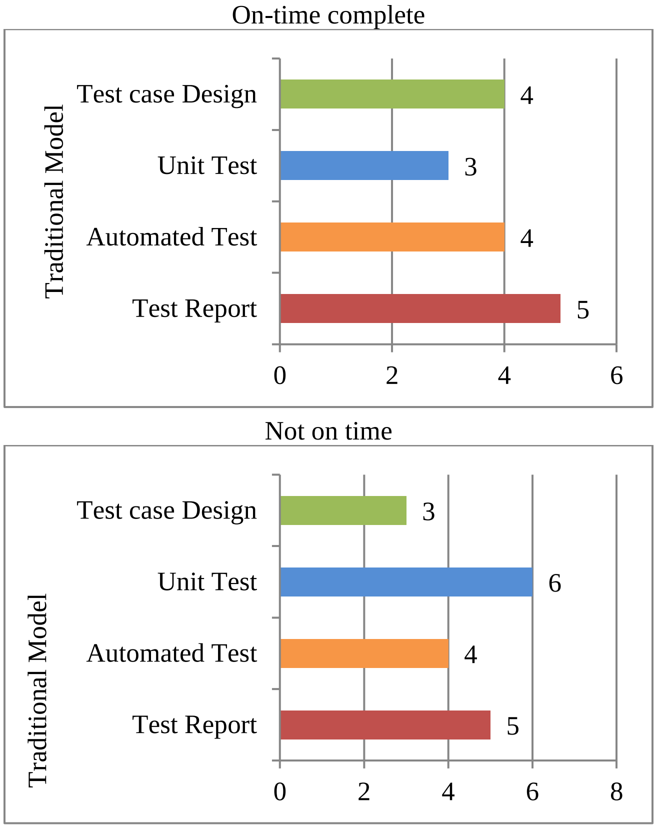

Fig. 6 The chart of Ontime complete and Not Ontime complete in the traditional teaching model

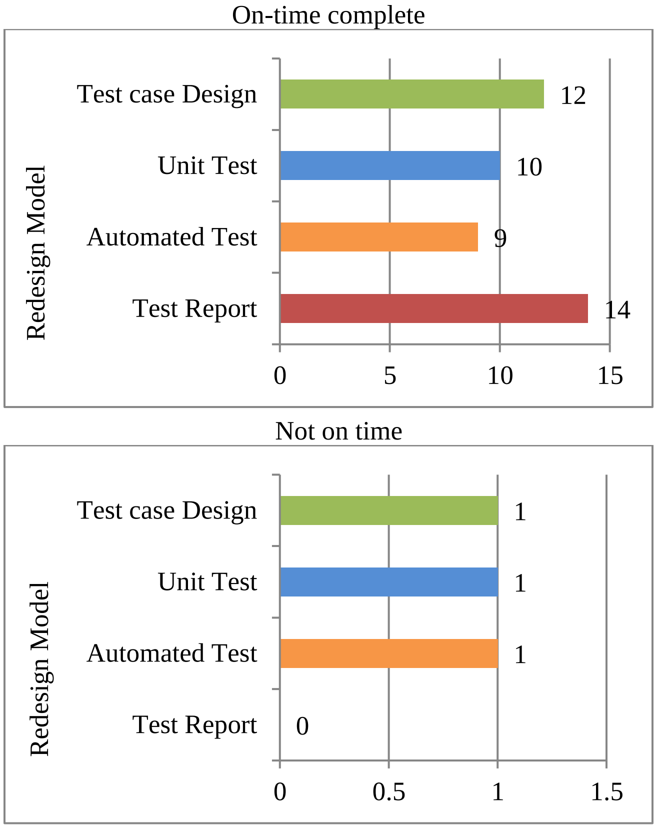

Fig. 7 The chart of Ontime complete and Not Ontime complete in redesigning the teaching model

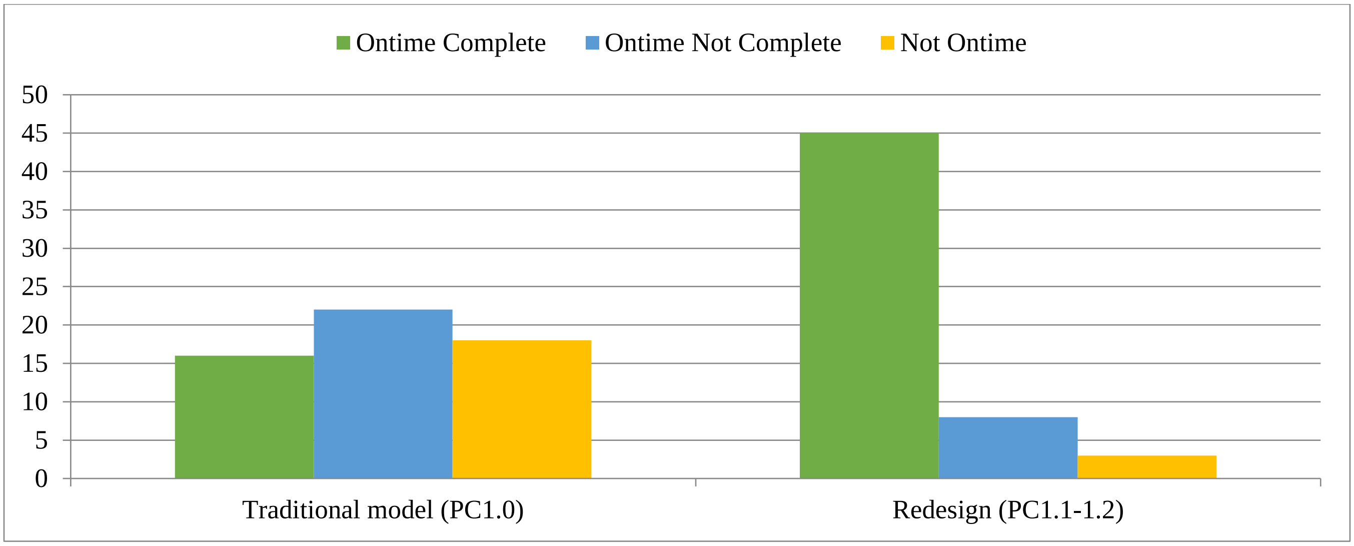

Fig. 8 An overall result between traditional and redesign teaching model

In the redesigned model with PC1.1-1.2 program specs, the highest number of on-time completions showed the same result as PC1.0 specs. From an overall result of the redesigned model, the group number of online completion in four tasks grew significantly. Furthermore, the number of group submissions in the Unit test was greater than the Automated test task in Figure 7. This result can be described along with the agile testing process because the SEUP student will be delivered a task every week. Moreover, in every sprint release, the report from automated tests will be used in every meeting.

From an overall 4 indicators in Figure 8, the redesigned model (PC1.1-1.2) shows a greater number of on-time completions at 8.96% from the traditional model to 25.2 percentages in the redesigned model. In terms of not on time, the number has been reduced from 10.08 percentages (traditional model) to 1.68% (redesign model). Moreover, software testing reports, test designs, and automated tests are improved skill sets accordingly.

5. Conclusion and Practical Implications

From an overall result, the on-time completion in the redesigned model showed a greater number from 8.96 to 25.2 percentages. Since the redesigned model added the new teaching with agile testing context, the amount of not on time has been changed to zero groups (5 to 0).

The unit test is another concern task that contains soft advance skills; the result in the redesigned model was reduced not on time in the traditional model from 6 groups late to 1 group late in the redesigned model. To conclude, improving teaching software testing for SEUP students or other undergraduate students is about a practical skill [9]. Practical skills are supposed to go along with a real requirement problem.

Furthermore, the real software requirement specification from a real user is an initial phase of the agile testing context. Lastly, this research study is another useful resource for future study to determine more factors in teaching abilities when technology and business change.

In addition, the current version of the software testing teaching framework will be improved by collecting positive and negative results from the previous version.

Funding Statement

This research was supported by the Thailand Science and Research and Innovation Fund and the University of Phayao (Grant No. FF65)

Acknowledgement

This research study was evaluated by software engineering students at the School of Information and Communication Technology, University of Phayao. This research was also supported by the Department of Probation office in Phayao, Thailand.

References

[1] Francis A. Adesoji, "Bloom Taxonomy of Educational Objectives and the Modification of Cognitive Levels," *Advances in Social Sciences Research Journal,* vol. 5, no. 5, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Abdulrahman Alarifi et al., "SECDEP: Software Engineering Curricula Development and Evaluation Process using SWEBOK," *Information and Software Technology,* vol. 74, pp. 114-126, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[3] D. Carrington, "Teaching Software Design and Testing," *FIE'98. 28th Annual Frontiers in Education Conference Moving from'Teacher-Centered'to'Learner-Centered'Education, Conference Proceedings,* vol. 2, pp. 547-550, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[4] Honghong Chen, Xu Wang, and Liangguang Pan, "Research on Teaching Methods and Tools of Software Testing," *15th International Conference on Computer Science & Education, IEEE,* pp. 760-763, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] James de Castro Martins et al., "Agile Testing Quadrants on Problem-Based Learning Involving Agile Development, Big Data, and Cloud Computing," *Information Technology-New Generations,* Springer, Cham, pp. 429-441, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] Joel Ayala de la Vega, and Irene Aguilar Juarez, "Comparison of the ACM/IEEE CE2004 and ACM/IEEE CE2016 Curricular Guide / ACM/IEEE CE2004 and ACM/IEEE CE2016 Curricular Guide Comparative," *RECI Ibero-American Magazine of Computer Sciences and Informatics,* vol. 7, no. 13, pp. 19-42, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[7] Isaac Souza Elgrably, and Sandro Ronaldo Bezerra Oliveira, "Construction of a Syllabus Adhering to the Teaching of Software Testing Using Agile Practices," *IEEE Frontiers in Education Conference, IEEE,* pp. 1-9, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Isaac Souza Elgrably, and Sandro Ronaldo Bezerra Oliveira, "Model for Teaching and Training Software Testing in an Agile Context," *IEEE Frontiers in Education Conference, IEEE,* pp. 1-9, 2020. [CrossRef] [Google Scholar] [Publisher link]

[9] Kun Ma et al., "Project-Driven Learning-by-Doing Method for Teaching Software Engineering using Virtualization Technology," *International Journal of Emerging Technologies in Learning,* vol. 9, no. 9, pp. 26-31, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[10] D. Talby et al., "Agile Software Testing in a Large-Scale Project," *IEEE Software,* vol. 23, no. 4, pp. 30-37, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[11] Qing Hong et al., "Occupational Ability Oriented Graduate Education in Software Engineering," *International Journal of Emerging Technologies in Learning,* vol. 10, no. 8, pp. 25-29, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] Juan C. Yelmo, and Juan Fernández-Corugedo, "An Experience of Educational Innovation for the Collaborative Learning in Software Engineering," *International Journal of Emerging Technologies in Learning,* vol. 6, no. 2, pp. 26-32, 2011. [CrossRef] [Google Scholar] [Publisher Link]