An approach to Teaching Software Testing Supported by Two Different Online Content Delivery Methods

Ingrid A. Buckley, PhD[1], and Peter J. Clarke, PhD[2]
[1]Florida Gulf Coast University, USA, ibuckley@fgcu.edu
[2]Florida International University, USA, *clarkep@cis.fiu.edu*

Abstract–*Software testing is an important aspect of software development, which often is left until the end of the software development life cycle. As a result, testing is generally neglected or inadequately performed, resulting in poor quality software products. To ameliorate this, universities are now offering courses in software testing to introduce this important skill to students before they begin their careers in industry. At our university, undergraduate students take software testing in the last semester of their degree program. One of the objectives of this course is to teach students how to perform continuous software testing at each stage and phase of the software life cycle, not just at the end of the development process. This course is delivered using a hybrid approach. We are interested in determining whether presenting this information online in an interactive manner that employs a combination of collaborative learning, gamification, problem-based learning, and social interaction leads to higher levels of knowledge gain in software testing. We conducted a study using (i) an interactive online learning system called SEP-CyLE- Software Engineering and Programming Cyberlearning Environment (Experimental Group) and (ii) a plain static text formatted website with no interactivity (Control Group). We evaluate the data collected and discuss the implications of our findings.*

Keywords- *Software Testing, Cyber learning, software engineering, software development, engineering educational tool*

I. Introduction

Software continues to impact all aspects of our lives, including the way we use our phones, computers, home appliances, medical devices, and cars, just to name a few. Due to the ubiquitous nature of software there is a great demand for skilled software developers. Software testing is a subset of software development and is an essential aspect to ensure that software is built correctly to reduce bugs and vulnerabilities that can threaten the software we rely on. Software vulnerabilities and bugs have caused significant lost [13], [17] and inconvenience when they fail or are exploited by hackers across different domains: health care, financial, government, telecommunications and transportation systems.

In general software testing is an often neglected aspect of software development that is either rushed or scaled down significantly to meet deadlines. This approach is often adopted by students who are already inexperienced in software development. Students in particular generally, test in a way that shows that their programs work, often times, they perform very little testing to find bugs or defects in their programs; and testing is rarely ever an automated, planned or systematic activity. Inadequate testing is a major issue in the software development field and bugs and defects account for huge losses [13] and there is usually a huge financial cost to customers when testing is neglected or mismanaged. In academia it is important to motivate students to take a responsible approach to software development by integrating the teaching of testing with the goal of finding and correcting bugs [2]. The cost of inadequate testing is high since it increases maintenance cost, negatively impacts customer perception of a product and leads to loss in profits.

Due to our increasing reliance on software, there is a need to educate and equip students with effective software testing skills and knowledge. In this paper, we conducted a study to find an effective approach to teach undergraduate students software testing principles. The study was conducted at Florida Gulf Coast University. The testing principles are presented to students supported by two different formats to determine which approach is more effective at relaying these important skills to novice software testers and developers [2].

The goal of this study is to identify how well students assimilate software testing knowledge in a self-paced and feedback driven online learning environment. The learning environment used in the study is *SEP-CyLE* (*Software Engineering and Programming Cyberlearning Environment*) [4] that contains learning content in the form of learning objects and tutorials. SEP-CyLE provides an instructor with the ability to use several embedded learning and engagement strategies (ELESs) to improve student learning. These ELESs are collaborative learning, gamification, problem-based learning, and social interaction.

The outline of the paper is as follows. Section 2 presents a background on software testing and learning content and engagement strategies. Sections 3 briefly introduce the online content delivery systems used in the studies in this paper. Section 4 presents a study including the objective of the study, study setup and an evaluation of the results. Section 5 discusses related work and Section 6 provide some conclusions.

2. Background

In this section we present background on software testing at it relates mainly to the content of the courses mentioned in the studies in Section 4. In addition, the learning content and engagements strategies used in the online delivery systems are also briefly introduced.

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2018.1.1.377
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773

2.1 Software Testing

There are many definitions used in the literature for software testing. The IEEE Standard defines software testing as the *dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior* [1]. Copeland states that in its simplest form testing is the process of comparing "what is" with "what ought to be" [6]. Testing approaches can be grouped into three broad categories, black box testing, white box testing and gray box testing.

In black box testing the module to be tested is treated as a black box with only the inputs and outputs accessible to the tester, thus this approach is based solely on the requirements and specifications of the module. White box testing (or glass box testing) is based on the internal structure of the module, such as the control and data flow paths in the module. Gray box testing is a hybrid of black box and white box testing allowing the tester to peek into the module to see how it is implemented [6], [14].

Testing software is usually done at three levels, these are: unit, subsystem and system. Unit testing focuses on verifying the smallest meaningful module in the software, for object-oriented (OO) programs this is usually the class. Subsystem testing focuses on a group of class that represents a component of a system that provides services to other component, in OO programs the class are group into a package. Before subsystem testing can be performed, individual classes in the in the subsystem must be combined by performing integration testing. System testing is verifying the entire software system against the requirements of the client or end-user [6], [14].

2.2 Learning Content and Engagement Strategies

The learning content in the online delivery systems are in the form of learning objects and tutorials. A learning object (LO) is a module of content that usually requires 2 to 15 minutes for completion, is self-contained, interactive, reusable and can be aggregated [15]. Each LO has four components: *Learning Objective* – describes audience the LO targets, the behavior expected of the learner, condition under which the behavior occurs, and accepted standard of behavior; *Content* – learning material to support the objective; *Practice* – exercises for learners to review the facts and key concepts; *Assessment* - a means to check if the learner has achieved the learning objective. The tutorials are similar to LOs but there are no Practice or Assessment components. The LOs are used to deliver key facts and concepts to the learner, while the tutorials focuses on providing information on how to use various tools associated with the specific discipline, e.g. software unit testing tool – JUnit [10].

The learning and engagement strategies (LESs) used in one of the online delivery system (SEP-CyLE) include *collaborative learning, gamification, problem-based learning,* and *social interaction*. Clarke et al. [5] introduce how these LESs are used in WReSTT-CyLE (Web-Based Repository of Software Testing Tutorials: a Cyberlearning Environment) [3] which is the forerunner to SEP-CyLE. *Collaborative learning* is where two or more people work in groups mutually searching for understanding, solutions, or meanings, or creating a product. *Gamification* uses game design elements and game mechanics to improve user experience and engagement with a system. *Social Interaction* enhances knowledge acquisition using various social activities. *Problem-based learning,* which is new in SEP-CyLE, *it* is a learner-centered instructional method in which students learn through solving ill-structured problems [12].

3. ONLINE LEARNING SYSTEMS

In this section we describe the two online learning systems used in the study presented in this paper. These online learning systems are the Simple WReSTT and SEP-CyLE, we first introduce SEP-CyLE then describe how Simple WReSTT differs from SEP-CyLE.

3.1 Software Engineering and Programming Cyberlearning Environment (SEP-CyLE)

SEP-CyLE is a NSF funded project that uses a combination of learning and engagement strategies (LESs) to get students involved in the learning process. The project aims to provide a cyberlearning environment that facilitates the improvement of students' conceptual understanding and practical skills in software engineering and programming. The main goals of SEP-CyLE are to *create new learning materials* and *develop faculty expertise* to significantly increase the number of undergraduate students that are exposed to testing methodologies and tools in undergraduate courses with a programming component.

Fig. 2 shows the hierarchical structure of the SEP-CyLE functionality. SEP-CyLE contains 5 major components similar to the ones for WReSTT-CyLE as described in Clarke et al. [5]. These components include: (1) Authentication – requires all users to log on to the system; (2) Embedded learning and Engagement Strategies (ELESs) – these strategies include collaborative learning, gamification, problem-based learning and social interaction; (3) Learning Content – contains the digital learning objects (DLOs) and tutorials accessible by the students; (4) Administration – provide access to the administrator to configure SEP-CyLE, e.g., setting up reports and configuring courses; and (5) Course Management – provides instructors with the ability to configure and generate reports related to a course. Note that there are differences between WReSTT-CyLE and SEP-CyLE , that is, the following elements are not in WReSTT-CyLE: problem-based learning component (2.3), chat (2.4.4), and ratings (2.4.5).

The inclusion of DLOs was an important enhancement to the contents of WReSTT-CyLE for software testing education and community. The DLOs were made in response to feedback from users in the academic community.

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773

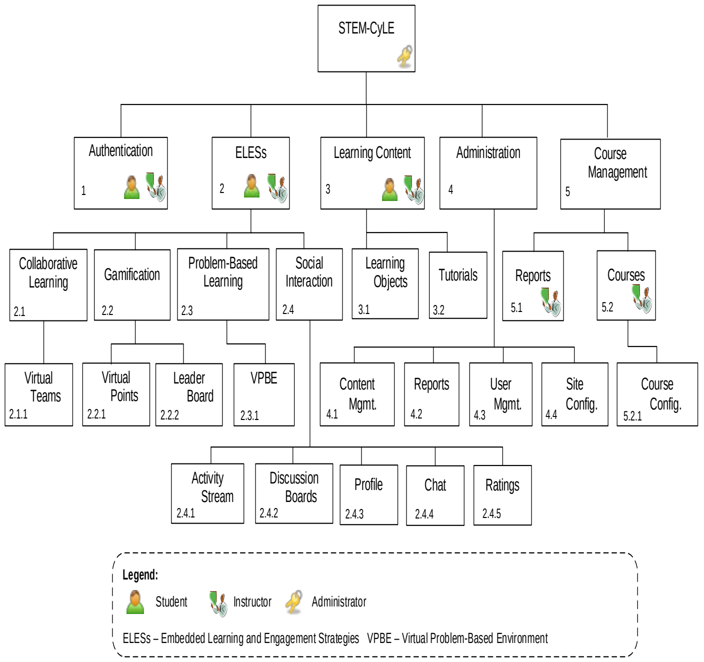
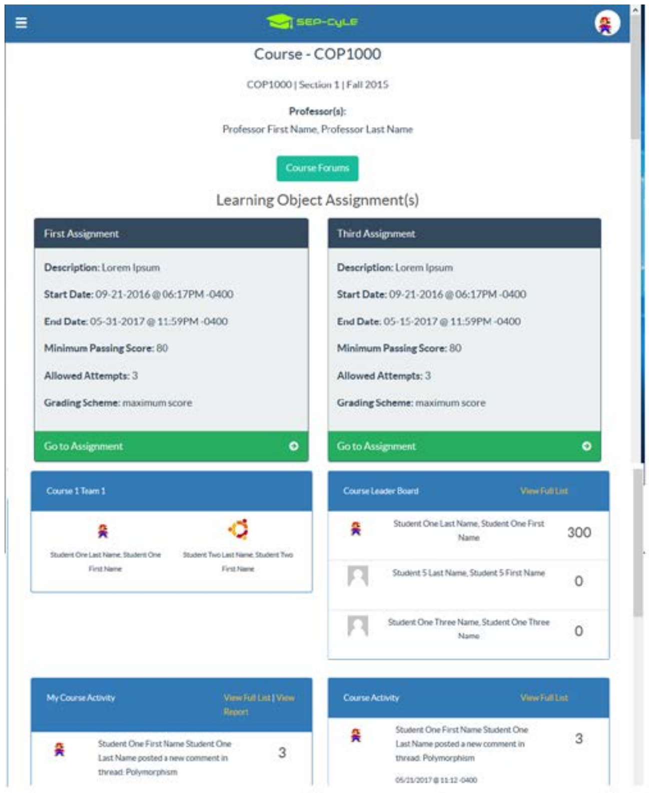


Fig. 1: Block diagram showing the hierarchical structure of the functions of SEP-CyLE.

These enhancements include: (1) presenting the material in the learning objects using varied formats (e.g., video, audio and text); (2) new learning objects on testing techniques for black-box and white-box testing; and (3) new tutorials for testing tools based on cross-platform IDE (e.g., Eclipse and NetBeans).The transition to learning objects from tutorials allows for the sequencing of different levels of content on a specific testing topic and the ability to link objects on testing techniques to tutorials on testing tools. SEP-CyLE improved on WReSTT-CyLE by including a LO creator that allows instructors to create DLOs on various topics and share them with the community.

The ELESs are implemented in SEP-CyLE to provide the following services to the students: Collaborative learning – students are placed in virtual teams and participate in both team and class-wide activities, e.g., completion of DLOs and posting messages to the class forum. Gamification – is centered on the use of virtual points where students obtain virtual points when they complete various tasks, such as completing DLOs and passing the quiz, posting to the forum, and bonus points are awarded for teams that complete activities. Social Interaction – students are provided with features that include student profiles, message forums, and ratings of posts and DLOs, among others. The problem-based learning component is not yet fully implemented in SEP-CyLE and is expected to incorporate the use of an IDE where students can test simple programs.

Fig. 2 shows a student's course page in the demo version of SEP-CyLE for COP 1000. The top of the page shows the specifics of the course, including the course number, and the professor's name. Below the course information is a link to the course forum, followed by links to the assigned DLOs and tutorials (not shown).



Course - COP1000

COP1000 | Section 1 | Fall 2015

Professor(s):
Professor First Name, Professor Last Name

Learning Object Assignment(s)

First Assignment

Description: Lorem Ipsum
Start Date: 09-21-2016 @ 06:17PM -0400
End Date: 05-31-2017 @ 11:59PM -0400
Minimum Passing Score: 80
Allowed Attempts: 3
Grading Scheme: maximum score

Third Assignment

Description: Lorem Ipsum
Start Date: 09-21-2016 @ 06:17PM -0400
End Date: 05-15-2017 @ 11:59PM -0400
Minimum Passing Score: 80
Allowed Attempts: 3
Grading Scheme: maximum score

Course 1 Team 1

Student One Last Name, Student One First Name
Student Two Last Name, Student Two First Name

Course Leader Board

Student One Last Name, Student One First Name	300
Student 5 Last Name, Student 5 First Name	0
Student One Three Name, Student One Three Name	0

My Course Activity

Student One First Name Student One Last Name posted a new comment in thread: Polymorphism 3

Course Activity

Student One First Name Student One Last Name posted a new comment in thread: Polymorphism 3
05/21/2017 @ 11:12 -0400

Fig. 2: Student course page in SEP-CyLE.

The DLO information includes a description of the DLO, start and end dates when the DLO will be available, minimum passing score, number of attempts on the recorded assessment, and grading scheme. Below the DLOs on the left are the members of the team and on the right is the course leader board for the top 3 students in the class showing their virtual points. The final row of the page shows the students' course activities on the left, and the course activities for the all the students in the class on the right.

3.2 Simple Content Delivery System (Simple WReSTT)

A version of WReSTT-CyLE was created to assist with performing studies in the classroom, we refer to it as *Simple WReSTT* (http://simple.wrestt.cis.fiu.edu/). This version of WReSTT is a plain web site which provides access to all the learning content in SEP-CyLE (and WReSTT-CyLE) but does not contain any of the ELESs. This means that instructors cannot use collaborative learning, gamification, or social interaction in their classes through the Simple WReSTT. In addition there is no user authentication required thereby providing students with easy access to the DLOs and tutorials.

The DLOs in Simple WReSTT do not provide the practice or assessment quizzes, the learning content is presented on a single page, and there is no facility for students to rate the tutorial or post comments on the quality of the DLO. All the tutorials in SEP-CyLE are available in Simple

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773

WReSTT but do not allow students to rate the tutorial or post comments on the quality of the tutorial. Finally DLOs and tutorials cannot be assigned with a deadline for submission. It should be noted that students using SEP-CyLE can also access the DLOs and tutorials using the menu located in the upper left of the student's dashboard (3 bars), see Fig. 2.

4. Software Testing Study

In this section we describe the study that was performed to evaluate the impact of using SEP-CyLE in the classroom at Florida Gulf Coast University. The goal of the study is to determine if given two different online learning systems, if there a significant difference between university seniors learning software testing principles using SEP-CyLE and those who do not?

4.1 Study Setup

The study included students across two sections of a software testing course. This course is a hybrid required class that seniors typically complete in the last semester of their degree program. Each section of the course meets with the instructor once per week and students are assigned online work outside of class time. Here the word hybrid, describes a course in which some traditional face-to-face time has been replaced by online learning activities. On the first day of class for each section, students were given an overview of the software testing study and they were invited to volunteer to participate in the study. The required student consent forms were supplied to those who chose to participate. Note that all students are still given access to the software testing material whether they volunteered to have their data used as a part of the study or not.

The study consisted of a control and an experimental group. The students enrolled in the 1st section of the course formed the control group. They were given access to various software testing materials on an open plain text-based website without interactivity (Simple WResTT). While, students in the 2nd section of the course formed the experimental group. They were assigned the same learning material on SEP-CyLE, where they got to experience an interactive learning environment with ELESs. The experimental group had access to quizzes on each assigned software testing learning object (LO) and each of the students in that group was given a unique login to access SEP-CyLE.

Throughout the duration of the course, students in both groups were assigned a series of supplemental LOs on black-box and white-box testing. In this study, students completed a pretest and posttest at the start at the course and again at the end of the course, respectively. The pretest/posttest is comprised of 10 software testing questions which includes questions on various aspects of software testing such as black box and white box testing techniques, bugs, branch, and statement coverage [6], [14].

4.2 Evaluation of the Results

40 students completed the pretest and 36 students completed the post test. Overall, the results in Table 1 illustrates that there was a 50% increase in the mean score of the control group versus 15% for the experimental group. Likewise, there was a 60% increase in the median score of the control group versus 14% for the experimental group. There was a 4.3% decrease in the standard deviation of the scores of the control group versus 17.41% for the experimental group. Both groups seem to perform at the same posttest level, averaging around 80%. However the control group performed significantly lower at the pretest stage, averaging around 51%, compared to 70% for the experimental group.

TABLE 1

	Control Group			**Experimental Group**		
	Pretest	Posttest	% Change	Pretest	Posttest	% Change
Mean:	51	77	50	70	81	15
Median:	50	80	60	70	80	14
Std. Deviation:	22.2	21.24	-4.3	22.5	18.38	-17.41

Comparison performance between control and experimental group

Throughout the duration of the study the students shared the challenges they were experiencing. Students in the experimental group stated that the interactive environment had some glitches that impacted their ability to learn the material. For example when taking quizzes when any of the available answers were selected, it would be marked incorrectly even if it was correct. Some students mentioned that they also were not given the option to submit their practice quizzes; after spending the time to go through each question. Some students expressed that there was missing information on some learning objects. Students in the control group, who utilized the plain website, did not experience any of the problems mentioned above and did not make any complaints about the contents or format on the plain website. Additionally, the plain website did not require username and password to gain access to the software testing learning objects, while login was required to access the learning objects on SEP-CyLE. It may be that these technical difficulties interfered with the enhancements of the interactive website used by the students in the experimental group, thus limiting its use from its fullest potential.

Additional factors may be required to help in determining whether easy access to the software testing material was the reason why students in the control and experimental groups performed equally on the posttest. Looking more closely at student factors, in general, students in the control group completed all assignments given, however some students in the experimental group were more prone to not submit assignments. This likely impacted their mastery of the material. Similarly, both groups completed a

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773

comprehensive software testing final examination and the mean score for the experimental group was 81% versus 74% for the control group. Overall both groups completed quizzes, exams and a group project, and each student in each section received a course grade, the letter grade distribution is given below in Fig. 3 and Fig. 4. The grade distribution shows that overall both groups were on par grade wise.

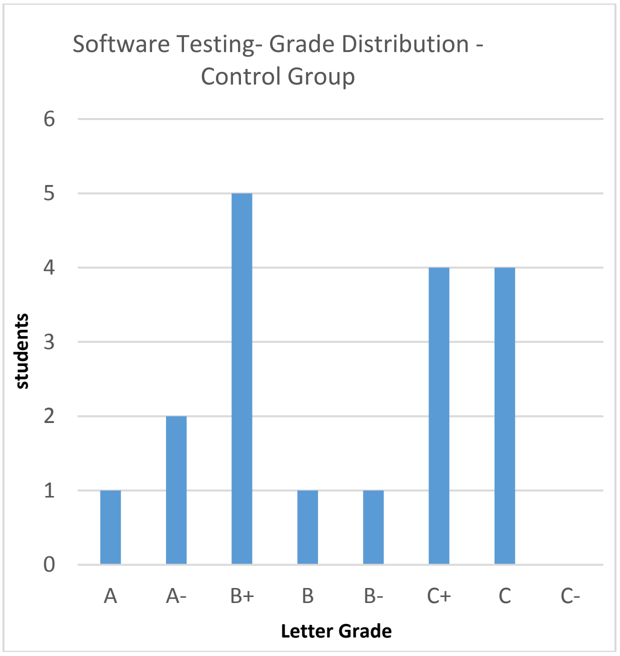


Fig. 3 Control Group- Course Grade

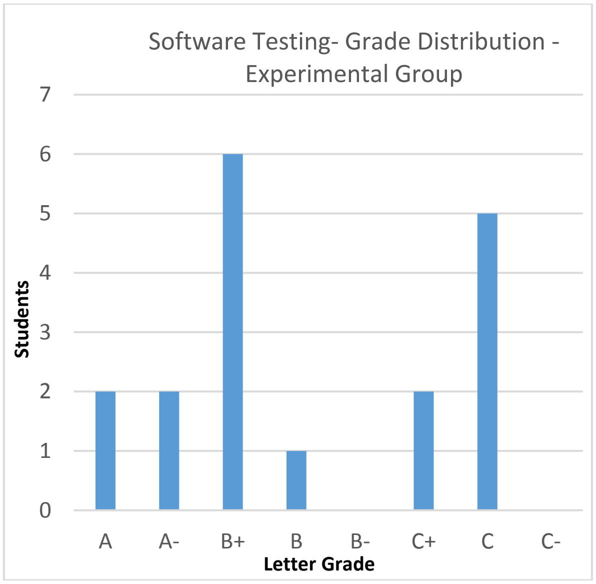


Fig. 4. Experimental Group - Course Grade

Overall, both the experimental and the control groups showed improvement in their software testing knowledge and skills but the control group performed much better on the posttest. However, they did not perform as well on the pretest and as such, their posttest performance showed a greater improvement. Another factor that may have affected the performance was the interactive cyberlearning environment (SEP-CyLE) which was a recently developed and launched. It had some bugs and glitches that may have hampered students as they tried to complete the LOs and quizzes. This experience may have prevented students from fully benefitting from the additional features it had to offer.

The control group did not require a login to access the material on the plain website, and this may have made it easier for them to access and consume the material without experiencing disruptions. In order to better understand what factors contributed to the experimental group's lower performance on the posttest, the study will have to be repeated using additional controls.

5. Related Work

Several studies have been performed that reinforce the fact that many software failures could be prevented by performing basic software testing. Yuan et al. [16] studied 198 randomly sampled real world failures and their findings state that basic testing of error handling code could prevent 58% of catastrophic failures; and 92% of system failures were caused due to incorrect handling of non-fatal errors. Gazzola [11] looked at ways to address faults that cannot be feasibly tested in house by utilizing the field itself as testbed for running test cases to uncover bugs. This approach requires a large diverse testing environment where a large number of test cases can be run tested with the aim of revealing errors in a timely manner, which would not be possible to identify during regular in house testing. This approach presents a faster and effective means of identifying bugs, however a large diverse testing environment is not always feasible or practical.

Other approaches have adopted the use of tools to uncover bugs. Dolby et al. [7] propose the use of relational logic and a SAT checker which checks code against software specifications. In this approach, they encode a program's relational logic using a constraint solver to find specification violations which cause bugs. The SAT checker can check a mixture of structural and numerical properties written in a rich specification language on realistic fragments of programs. Dukes et al. [8] present a case study where five different tools are used for web application security. They employed the use of different tools to identify different types of defects and bugs. This case study exposed students to a variety of software testing tools. However since each tool is independent of each other, all of the bugs cannot be viewed together as a whole in one central integrated development environment (IDE).

Other approaches, utilize different teaching strategies to teaching software testing. Buckley et al. [2] proposed a teaching strategy which leverages the use of basic data structures to teach the fundamentals of software testing principles. In this approach, students must first understand the fundamental properties and constraints of a stack, binary tree or a recursive problem. The idea is to encourage students to fully understand the core properties and constrains of a

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773

system, this is analogous to understanding the requirements of a system; as this aspect is imperative in order to write effective test cases to uncover bugs. In this project, students were given exercises to write test cases that ensure that each data structure's properties and constraints are upheld throughout implementation to uncover hidden bugs. Overall, the students showed an improvement in their ability to write test cases that consider the fundamental principles and constrains surrounding a given problem.

Previous work that is related to the use of WReSTT-CyLE in the classroom include the work by Fu et al. [9] and Clarke et al. [5]. Fu et al. [9] employed the use of gamification in WReSTT-CyLE to teach students software testing at Florida A&M University. WReSTT-CyLE is a cyberlearning environment that provides a collaborative and sustainable platform for learners to continue studying outside of the classroom.

Similarly, gamification is an emerging pedagogical technique that is used to engage students in a non-game context. The results of their study shows that gamification in conjunction with WReSTT-CyLE increased the engagement and motivation of students in learning software testing. Clarke et al. [5] described how WReSTT-CyLE was used to help students and instructors learn various software testing techniques and testing tools. The work reported in the paper was part of a four year project with the objectives of training instructors in using WReSTT-CyLE in the classroom and using WReSTT-CyLE to support pedagogy in the classroom. The results of the project showed a positive impact on the classes taught by those instructors that attended the professional development workshops in the project. The results from a student survey showed that students found the WReSTT-CyLE site user friendly, they viewed the collaborative learning LES as positive, and that the course activity stream social feature encourage team members to complete their tasks.

6. Conclusion

The study presented in this paper illustrates at a high level that the learning objects provided to students during their courses provided significant value to students in learning software testing principles even though the material was presented using two different online learning systems. Both the experimental and the control groups showed improvement in their software testing knowledge and skills at the end of the course. Even though both group performed similarly on the posttest, the experimental group performed better overall on their final examination for the course. Although there are some clues, it is not clear exactly what factors may have contributed to the similar performance on the posttest. The study may have to be repeated with other measurable controls to better understand the cause. SEP-CyLE is a promising tool and environment that can be tailored and adapted to aid instructors in teaching software testing and other STEM based courses. Once the features are fine-tuned, SEP-CyLE can become a good supplement tool, which provides students with feedback and interactivity outside of the class room that will advance their software testing knowledge and skills.

References

[1] P. Bourque and R. Dupuis, "*Guide to the Software Engineering Body of Knowledge 2004 Version"*, IEEE Computer Society, Los Alamitos, California, 2004.

[2] I. A. Buckley and W. S. Buckley, "Teaching software testing using data structures", *International Journal of Advanced Computer Science and Applications (IJACSA)*, 22(4), 2017.

[3] R. Chang-lau and P. J. Clarke, "Web-based repository of software testing tutorials a cyberlearning environment (WReSTT-CyLE)", 2017. [Online]. Available: http://wrestt.cis.fiu.edu/. [Accessed June 2017].

[4] R. Chang-lau and P. J. Clarke, "Software engineering and programming cyberlearning environment (SEP-CyLE)", 2018. [Online]. Available: https://stem-cyle.cis.fiu.edu/instances [Accessed Jan. 31, 2018].

[5] P. J. Clarke, D. L. Davis, R. C. Lau, Y. Fu, J. D. Kiper, and G. S. Walia, "Using WReSTT cyberlearning environment in the classroom", Proceedings of the 124rd Annual ASEE Annual Conference and Exposition, June 2017. Paper ID: 19953.

[6] L. Copeland, *A Practitioner's Guide to Software Test Design*. Artech House, Inc., Norwood, MA, USA, 2003.

[7] J. Dolby, M. Vaziri, and F. Tip, " Finding bugs efficiently with a sat solver", Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ESEC-FSE '07, pages 195-204, New York, NY, USA, 2007. ACM.

[8] L. Dukes, X. Yuan, and F. Akowuah, "A case study on web application security testing with tools and manual testing", In *2013 Proceedings of IEEE Southeastcon*, pages 1-6, April 2013.

[9] Y. Fu and P. J. Clarke, "Gamification-based cyber-enabled learning environment of software testing", In Proceedings of the 123rd Annual ASEE Annual Conference and Exposition, June 2016. Paper ID: 15359.

[10] E. Gamma and K. Beck. JUnit, January 2017. [Online]. Available: http://www.junit.org/ [Accessed Jan. 31, 2018].

[11] L. Gazzola, "Field testing of software applications", In *Proceedings of the 39th International Conference on Software Engineering Companion*, ICSE-C '17, pages 429-432, Piscataway, NJ, USA, 2017. IEEE Press.

[12] C. E. Hmelo-Silver and C. Eberbach, "Learning theories and problem-based learning", In S. Bridges, C. McGrath, and T. L. Whitehill, editors, Problem-Based Learning in Clinical Education: The Next Generation, pages 3,17. Springer Netherlands, Dordrecht, 2012.

[13] N. Harley, "10 of the most costly software errors in history". May 29, 2014. [Online]. Available: https://raygun.com/blog/10-costly-software-errors-history/. [Accessed Feb. 5, 2018]

[14] A. P. Mathur, "Foundations of Software Testing", Addison-Wesley Professional, 2nd edition, 2013.

[15] R. S. Smith, "Guidelines for authors of learning objects", The New Media Consortium, 2004.

[16] D. Yuan, Y. Luo, X. Zhuang, G. R. Rodrigues, X. Zhao, Yongle Zhang, P. U. Jain, and M. Stumm, " Simple testing can prevent most critical failures: An analysis of production failures in distributed data-intensive systems", In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 249-265, Berkeley, CA, USA, 2014. USENIX Association.

[17] Wikipedia, "List of Software Bugs", [Online], Available: https://en.wikipedia.org/wiki/List_of_software_bugs [Accessed Jan. 31, 2018]

This material is based upon work supported by the National Science Foundation under Grant number DUE-1562773