Combining Learning and Engagement Strategies in a Software Testing Learning Environment

PETER J. CLARKE and DEBRA L. DAVIS, Florida International University, USA
INGRID A. BUCKLEY, Florida Gulf Coast University, USA
GEOFF POTVIN and MANDAYAM THIRUNARAYANAN, Florida International University, USA
EDWARD L. JONES, Florida A&M University, USA

There continues to be an increase in enrollments in various computing programs at academic institutions due to many job opportunities available in the information, communication, and technology sectors. This enrollment surge has presented several challenges in many Computer Science (CS), Information Technology (IT), and Software Engineering (SE) programs at universities and colleges. One such challenge is that many instructors in CS/IT/SE programs continue to use learning approaches that are not learner centered and therefore are not adequately preparing students to be proficient in the ever-changing computing industry. To mitigate this challenge, instructors need to use evidence-based pedagogical approaches, e.g., active learning, to improve student learning and engagement in the classroom and equip students with the skills necessary to be lifelong learners.

This article presents an approach that combines learning and engagement strategies (LESs) in learning environments using different teaching modalities to improve student learning and engagement. We describe how LESs are integrated into face-to-face (F2F) and online class activities. The LESs currently used are *collaborative learning*, *gamification*, *problem-based learning*, and *social interaction*. We describe an approach used to quantify each LES used during class activities based on a set of characteristics for LESs and the traditional lecture-style pedagogical approaches. To demonstrate the impact of using LESs in F2F class activities, we report on a study conducted over seven semesters in a software testing class at a large urban minority serving institution. The study uses a posttest-only study design, the scores of two midterm exams, and approximate class times dedicated to each LES and traditional lecture style to quantify their usage in a face-to-face software testing class. The study results showed that increasing the time dedicated to collaborative learning, gamification, and social interaction and decreasing the traditional lecture-style approach resulted in a statistically significant improvement in student learning, as reflected in the exam scores.

CCS Concepts: • **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → *Software testing and debugging*; • **Applied computing** → *Collaborative learning;*

Additional Key Words and Phrases: Active learning, gamification, problem-based learning, social interaction

This work is supported in part by the National Science Foundation's Improving Undergraduate STEM Education (IUSE) program under Award Numbers DUE-1225742 and DUE-1525112 (FIU), DUE-1525555 (FAMU), and DUE-1562773 (FGCU). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Authors' addresses: P. J. Clarke, D. L. Davis, G. Potvin, and M. Thirunarayanan, Florida International University, 11200 SW 8th Street, Miami, FL, 33199, USA; emails: {clarkep, dledavis}@cis.fiu.edu, {gpotvin, thiru}@fiu.edu; I. A. Buckley, Florida Gulf Coast University, 10501 FGCU Blvd. S., Fort Myers, FL, 33965, USA; email: ibuckley@fgcu.edu; E. L. Jones, Florida A&M University, 1601 S. Martin L. King Jr. Blvd., Tallahassee, FL, 32307, USA; email: edward.jones@famu.edu.

1946-6226/2021/11-ART11 $15.00
https://doi.org/10.1145/3469131

ACM Reference format:
Peter J. Clarke, Debra L. Davis, Ingrid A. Buckley, Geoff Potvin, Mandayam Thirunarayanan, and Edward L. Jones. 2021. Combining Learning and Engagement Strategies in a Software Testing Learning Environment. *ACM Trans. Comput. Educ.* 22, 2, Article 11 (November 2021), 25 pages.
https://doi.org/10.1145/3469131

1 INTRODUCTION

The ubiquitous nature of computing in the 21st century has resulted in a great demand for a workforce that is knowledgeable in the areas of **Computer Science (CS)**, **Information technology (IT)**, and **Software Engineering (SE)**. This demand is reflected in the everyday need for many jobs to be filled in the areas of CS/IT/SE. Employment of CS/IT/SE occupations is projected to grow by 12% from 2018 to 2028, much faster than the average for all occupations [71]. As a result of this demand, there has been a surge in enrollments in CS/IT/SE programs nationwide [19, 52, 68, 74]. This increase in enrollment has several challenges, including insufficient classroom space, insufficient faculty/instructors, and the demand for lab space, among others [19]. One challenge that needs further attention is student performance due to the large number of students entering programs who may not be well prepared to undertake CS/IT/SE courses. Although there has been a slight increase in the passing rates in the introductory programming classes since 2007 as stated by Bennedsen and Caspersen [3], more work still needs to be done to improve pedagogy in CS/IT/SE classes [6, 61]

To further address the challenge facing students who may be underprepared as they enter CS/IT/SE programs, it is imperative that CS/IT/SE instructors use evidence-based pedagogical approaches that are known to improve student learning and engagement in the classroom [28]. Several of these approaches have started to filter into CS/IT/SE education, including active learning [9], which includes **problem-based learning (PBL)** [41], **collaborative learning (CL)** [12], **gamification (GA)** [25], and **social interaction (SI)** [35]. Although there has been some use of these approaches in the CS/IT/SE classroom [67], little research has been done on how quantified combinations of these approaches impact student learning and engagement. More specifically, how much does each approach contribute to improving student learning and engagement? Throughout this article, we refer to *collaborative learning*, *gamification*, *problem-based learning*, and *social interaction* as **learning and engagement strategies (LESs)**, the approaches we will focus on.

In this article, we present an approach intended to motivate research on how to improve student learning and engagement by integrating quantified combinations of LESs into learning environments using different teaching modalities. We expect that our approach to integrating LESs into class activities will describe how each LES is quantified and the context in which it is used. During the past five years, the authors of this article have been involved in using LESs in a cyberlearning environment to determine the impact on student learning in CS/IT/SE courses. This cyberlearning environment, initially known as **Web-based Repository of Software Testing Tutorials (WReSTT)** (now **Software Engineering and Programming Cyberlearning Environment (SEP-CyLE)** [11]), has shown that using LESs in a cyberlearning environment can improve student learning in software testing [16], **Computer Science 1 (CS1)**, and CS2 classes [27, 50, 51].

Unlike previous works by the authors, this article focuses mainly on combining LESs in the **face-to-face (F2F)** activities of a software testing class at **Florida International University (FIU)**. A study was conducted to answer the following core research questions. (1) Does increasing the use of LESs in F2F class activities positively impact student learning? (2) Assuming there was a positive impact on student learning, what quantified combination of LESs resulted in the positive impacts on student learning? (3) What was the students' perception of the usefulness of the cyberlearning environment in online class activities while using LESs in the F2F activities?

This article is an extension of the work presented in Reference [15].[1] The extensions include the following: (1) a more detailed explanation of the model to integrate combinations of LESs into F2F and online learning activities; (2) additional details on the software testing course, specifically the midterm examinations; and (3) the study has been expanded to include results from (a) five additional semesters (spring 2017, summer 2017, fall 2017, spring 2019, and summer 2019) of integrating LESs into F2F class activities.

The main contributions of this work are as follows:

- A proposed pedagogical approach that instructors and researchers can use to determine which combinations of LESs best improve student learning and engagement. Instructors can use the description of the LESs to reflect on the approach(es) they are currently using in the classroom and adjust them appropriately to improve student learning. Researchers can build on the proposed pedagogical approach by extending or adapting the approach to improve student learning by using appropriate research designs.
- A description of how LESs may be used in F2F and online CS class activities, specifically software testing class activities.
- A method that may be used to quantify the use of LESs in F2F class activities.
- Results showing the positive impact on student learning when using various combinations of LESs in the F2F activities for a software testing class.

The remainder of the article is organized as follows. Section 2 provides a literature review on LESs and how they are used in software engineering classes. Section 3 presents the pedagogical model that integrates combinations of LESs into class activities. Section 4 describes how LESs may be integrated into a software testing class. Section 5 describes our study that uses LESs in the class activities, and we conclude in Section 6.

2 LITERATURE REVIEW

Many evidence-based pedagogical approaches have been used to improve student learning in engineering and computer science education [67]. This section provides a literature review of a subset of these pedagogical approaches referred to as LESs, and describes how LESs are currently used in CS/IT/SE courses.

2.1 Learning and Engagement Strategies

The LESs we focus on in this article include collaborative learning, gamification, problem-based learning, and social interaction. *Collaborative learning* as defined by Smith and MacGregor [62] is where "students are working in groups of two or more, mutually searching for understanding, solutions, or meanings, or creating a product." Smith et al. [62] state that there is a wide variation in the collaborative learning activities, but they must be centered on the students' exploration or application of the material. Several education goals of collaborative learning, include *involvement* (students participate more in their learning, interact with other students and interact with the teachers), *cooperation and teamwork* (students recognize different views and work together to build consensus to resolve these differences), and *civic responsibility* (encourages students to have a voice in shaping the ideas and values) [62]. The underlying theories for collaborative learning include social constructivism [24, 72] and social learning [1].

Gamification as defined by Deterding et al. [23] is "the use of game design elements and game mechanics to improve user experience and engagement with a system, which may be applied to an educational context." More importantly, gamification is about taking elements commonly

[1]©2019 American Society for Engineering Education, 2019 ASEE Annual Conference & Exposition, Tampa, FL.

implemented in games that humans find fun and enjoyable and applying those to other domains, specifically in a manner that promotes people's motivation to engage in desired behaviors and increase engagement [25, 26, 47]. These game design elements include goals/challenges, rapid feedback, leader boards, a points system, and levels. The underlying theories for gamification include self-determination theory [22], and intrinsic and extrinsic motivation theory [56].

PBL as defined by Schmidt [59] is "an approach to learning and instruction in which students tackle problems in small groups under the supervision of a tutor." PBL is well suited to helping students become active learners, because it situates learning in real-world problems, usually ill structured and open ended, and makes students responsible for their learning [2]. PBL has the potential to significantly increase student performance in the classroom by (a) activating prior knowledge, (b) elaborating on prior knowledge through discussion, (c) restructuring of knowledge, (d) learning in context, and (e) engaging in open-ended discussion thereby increasing the student's curiosity [59]. PBL underlying theories are based on a constructivist approach (cognitive and social) [24, 54, 58, 72] and include self-determination theory [22], student-centered strategies (based on client-centered therapy) [55], and discovery (inquiry) learning [7].

Social Interaction in the context of pedagogy is an approach that enhances knowledge acquisition through social activities, such as students establishing meaningful dialogue within student groups and with teachers [35, 39]. There are several social interaction learning styles that learners use when acquiring knowledge and information, usually involving the teacher and learner in different settings [39]. Okita [53] identified several social interaction learning styles, which include peer learning, reciprocal teaching, learning by teaching, learning by observation, and learning by doing. The underlying theories for social interaction include cognitive constructivism [54], social constructivism [72], situated learning [45], and cognitive apprenticeship [18]. It is worth noting that social interaction is inherently part of collaborative learning and problem-based learning, since the underlying theories for these approaches are built on some form of social theory.

2.2 Related Work on Using LESs in CS/IT/SE Classes

Gehringer [30] identifies how active and collaborative learning strategies have been used in teaching programming. This result was arrived at after the survey of dozens of pedagogical approaches used to teach programming. It is worth highlighting that the work by Gehringer was a step in the direction we are taking, since multiple active and collaborative learning approaches were used to teach programming in some instances. In our work, we go a step further by quantifying the percentages of each LES used during F2F class activities for a software testing class.

Soundarajan et al. [65] state that courses with a major component being the team project, e.g., software engineering, use collaborative learning as the pedagogical approach of choice. Furthermore, using collaborative learning in this way has the potential risk of individual learning getting lost, since the main focus is on the team's success. To mitigate this potential risk, Soundarajan et al. developed a mobile and web technologies tool to help individual students understand the relevant software engineering concepts while working on the team project. The main idea used in the tool is conflict-driven cooperative learning, where each student is effectively forced to consider and evaluate alternatives of the answers to quiz questions before the class meeting using a flipped class approach. The tool allows students to select each quiz question, justify the solution presented, and defend the answer during several rounds of discussion using the tool. This work provides evidence that collaborative learning alone may not be adequate for courses with team projects.

Junhua et al. [38] performed a systematic investigation on combining active learning and design-based learning during software engineering instruction. The investigation compares the performance of a control group using a traditional instruction approach, i.e., lecture-homework-project teaching, and a treatment group, using active learning through peer to peer instruction integrated into design project modules. Two quantitative assessments were used in the study, a self-assessment on the students' learning experience and a pre/posttest designed and constructed from the course inventory. The pre/posttest results were not disclosed to the instructors until after the course was completed. Although both approaches showed improved student performance, the peer-to-peer active learning and design-based learning approach received higher ratings for interest, engagement, and intrinsic motivation. The pre/posttest did not show a statistically significant difference in performance between the groups.

Lauvås et al. [44] performed a systematic literature review on the topics of teaching software testing, and the results point to different trends in teaching software testing. Gamification was one of the approaches to teaching software testing that seem to be gaining momentum. This approach is done by introducing game elements to support (1) practicing testing concepts by playing a game [17, 33] or (2) by disguising software testing techniques as quests that they need to complete and upon completing these quests students are rewarded with achievements, titles, or experience points [60]. Many of the gamified approaches to teaching software testing do have support via various research tools. Dicheva et al. [25] present a survey of published empirical results on the application of gamification in education. The authors further stated that although most of the papers surveyed showed promising results, more research was needed to determine the impact on student learning.

Liccardi et al. [46] investigated the role of social networks in computer science education, focusing on analyzing the role social networks play in students' learning experiences. The authors state that the social dimension of learning has been of great significance to both teachers and students and is critical to the learning process. The underlying theory to support this claim is based on Vygotsky's social development theory that states that social interaction is fundamental to the development of cognition [72]. It was found that social interaction within an online framework can help university students share experiences and collaborate on relevant topics. As such, social networks can act as a pedagogical agent, for example, with problem-based learning. However, it was pointed out that simple face-to-face interactions can become complex when done online due to the limitations of the available technologies to capture nonverbal cues during an interaction.

The work presented in this section shows how different pedagogical approaches may be used in software engineering and software testing classes to improve student learning and engagement. Our work differs from the works presented by proposing a method that can be used when combining LESs in different teaching modalities and be used to evaluate which combination(s) of LESs can be most effective in improving student learning and engagement. In addition, we provide guidelines on how each LES may be quantified when integrating LESs into a learning environment.

3 METHOD TO INTEGRATE LEARNING AND ENGAGEMENT STRATEGIES

This section describes our method for integrating LESs into CS/IT/SE learning environments. Before introducing the method, we present our motivation for this work and categorize each LES and **lecture-style (LS)** approach using characteristics described in the literature.

3.1 Motivation

Integrating LESs into learning environments is motivated based on the work conducted in a project coordinated by FIU and involving seven other academic intuitions. The project's initial focus was to integrate LESs into a cyberlearning environment and determine what impact would occur on student learning at a cross section of institutions. Based on the project's initial results, it was evident that one combination of LESs would not have the same impact on all the student groups using the cyberlearning environment. For example, the LESs that seem to appeal to students in upper division courses did not have the same impact on students in the introductory courses.

The cyberlearning environment previously mentioned is SEP-CyLE. SEP-CyLE provides students and instructors with vetted digital learning content and uses various embedded LESs to support pedagogy. The LESs currently implemented in SEP-CyLE are collaborative learning, gamification, and social interaction. The learning content is provided in the form of digital learning objects [64] on various **Software Engineering and Programming (SEP)** topics and tutorials for several SEP tools. More specifically, there is learning content on CS1, CS2, cybersecurity, software engineering, and software testing. Each digital learning object in SEP-CyLE consists of a unique identifier, a learning objective, a learning activity, a practice assessment, and a recorded assessment.

In most of the studies conducted during the project, the students found SEP-CyLE (previously WReSTT) to be a useful resource for learning about software testing techniques and testing tools [8, 16] and programming concepts in CS1 and CS2 [27, 50, 51]. Two studies reported in the literature are as follows. Clarke et al. [14] describe a study conducted at FIU that focused on the impact using WReSTT has on students' general knowledge of software testing and the use of tools. The experiment was conducted with a control group ($n = 35$) that did not use WReSTT and the treatment group ($n = 37$) using WReSTT. The pre/posttest was developed by the team and field-testing for face and content validity. Knowledge gains were measured using a pre/posttest approach [20]. The treatment group performed better on the posttest $F(1, 70) = 27.82, p < 0.01$. A follow-up analysis: $t(37) = -8.15, p < 0.01$, showed a significant difference between pre/posttest scores in the treatment group but not in the control group.

More recently, Narasareddygari et al. [50] conducted a study involving two universities to determine the impact of using different combinations of LESs in SEP-CyLE on student learning and understanding of programming concepts. The study was conducted in CS1 courses. The LESs used in the study were CL, gamification (G), and SI. The different groups used in the study were as follows: $CL + GA + SI(n = 50), GA + SI(n = 16), CL + SI(n = 19), CL + GA(n = 26), SI(n = 32)$, $GA(n = 29), None(n = 29)$. The pre/posttest instrument was developed from an existing CS1 concept inventory [69]. The CS1 pre/posttest scores show that the $GA + SI$ group outperformed the other groups with an improvement of +5.19 and a $p < 0.001$. It was concluded that gamification and social-interaction LESs are more useful (when used individually or in combination) than collaborative learning in CS1 classes.

Based on the early project results, some investigators started looking into integrating LESs in F2F class activities. An initial method to integrate LESs into F2F and online activities was presented in Reference [15], an updated version of the method is presented in this article. One of the major hurdles on integrating LESs, or other active learning approaches, is that there are no apparent boundaries for these pedagogical approaches [9, 21, 57]. In the next section, we use a set of characteristics to delineate LESs and the traditional lecture-style approach to quantify F2F and online class activities. It is worth pointing out that this article is a starting point on how practitioners and researchers may go about quantifying pedagogical approaches used in various classes. Anecdotal evidence has suggested that practitioners currently use multiple pedagogical approaches in their classes.

3.2 Characteristics of LESs and Traditional Lecture-Style

To quantify the contributions of different pedagogical approaches when teaching a class, it is important to clearly describe each approach's characteristics, showing where they overlap and where they differ. A summary of the characteristics of the pedagogical approaches (LESs and traditional lecture style) used as the basis of our method is presented in Table 1.

Table 1. Identifies the Characteristics Exhibited by the Four LESs and Traditional Approach

No.	Characteristics	LESs				Traditional
		CL	PBL	SI	GA	LS
1	Value on academic learning	Yes	Yes	Yes	Yes	Yes
2	Value on social learning	Yes	Yes	Yes	Maybe	No
3	Learner-centered	Yes	Yes	Yes	Maybe	No
4	Self-assessment: intrinsically motivated learning	Yes	Yes	Maybe	No	No
5	Self-reflection: on learning process and what was learned	Yes	Yes	Yes	No	No
6	A common tasks or learning activity suitable for group work	Yes	Yes	Yes	Maybe	No
7	Required Task that is a real problem	No	Yes	No	No	No
8	Small-group interaction focused on the learning activity	Yes	Yes	Yes	Maybe	No
9	Individual accountability and responsibility	Yes	Yes	Yes	Yes	Yes
10	Interdependence in working together through:					
	a. Goals: both social and academic goals	Yes	Yes	Yes	Yes	Maybe
	b. Tasks: structured learning tasks or assignments	Yes	Yes	Yes	Yes	No
	c. Resources: materials to be shared among group members	Maybe	Yes	Maybe	No	No
	d. Roles: group members assigned roles	No	Maybe	Maybe	Maybe	No
	e. Extrinsic Rewards: points being awarded for the completion of tasks	No	No	Maybe	Yes	Yes
11	Teaching and processing social skills	No	Maybe	Yes	No	No
12	Reflection on group process	Maybe	Maybe	Yes	Maybe	No
13	Community-building activities	No	No	Yes	No	No
14	Team-building activities	No	Maybe	Yes	No	No
15	Equal participation	No	Maybe	Maybe	No	No
16	Simultaneous interaction	No	Maybe	Yes	No	No
17	Use of structures: communication patterns in group e.g., round robin	Maybe	Maybe	Yes	Yes	No
18	Instructor as activist	Maybe	Maybe	Yes	Maybe	Yes
19	Status treatments: acknowledges good work of a student publicly	No	No	Yes	Yes	Maybe
20	Perspective-taking activities—help students understand perspective of other students	No	No	Yes	No	No
21	Freedom to fail: no penalty for failure	Maybe	Maybe	Maybe	Yes	No
22	Freedom of choice—student decides on which activity to perform	Maybe	Maybe	Maybe	Yes	No
23	Passive form of learning	No	No	No	No	Yes
24	Declining learner's attention	No	No	No	No	Yes

Note. "Yes" means present in given approach. "Maybe" could be present in the approach. "No" means not typically present in the approach.

CL - Collaborative Learning, GA - Gamification, PBL - Problem-Based Learning, SI - Social Interaction, LS - Lecture Style.

The contents of the table were created based on the work presented by the following authors. Davison and Major [21] compared cooperative learning, collaborative learning, and problem-based learning. Cattaneo [9] classified five active learning pedagogies using constructivist elements through theoretical and practical lenses. Kreijns et al. [42] reviewed the research literature to identify pitfalls for social interaction in computer-supported collaborative learning environments. Spears [66] investigated students' perception of social presence, social interaction, satisfaction, and collaborative learning of online courses compared to F2F courses. Dicheva et al. [25] performed a systematic mapping study of gamification in education, and Hamari et al. [32] a literature review of empirical studies in gamification. Jusoff and Samah [39] reviewed social learning styles, and Okita [53] a description of social interactions and learning. Bligh [5] described many of the characteristics of the traditional lecture-style approach to teaching (teacher as the Formal Authority [31]).

The characteristics in Column 2 of Table 1 are compiled from Davison and Major [21], Cattaneo [9], Dicheva et al. [25], and Bligh [5]. The entries in the rows of the table are one of three values: *Yes* (generally the pedagogical approach exhibits the characteristic), *No* (the approach does not generally exhibit the characteristic), *Maybe* (the approach may exhibit the characteristic). As an example, in Row 7 of Table 1, the only approach that requires using a real problem is problem-based learning. As pointed out in the literature, the problem is usually interdisciplinary, authentic, ill structured, and open ended [57].

Based on the underlying theories associated with the LESs defined in Section 2.1, collaborative learning and problem-based learning require some form of social interaction. To this end, there is some overlap of social interaction with collaborative learning and problem-based learning. As a result, social interaction will only be quantified separately when the characteristics for the other LESs are *No*. As an example, see Row 13 of Table 1, where social interaction is the only LES that exhibits community-building activities, i.e., the entire class is involved in a common class activity to get everyone acquainted with each other in a positive way. We will use a similar approach when quantifying all the pedagogical approaches.

3.3 Method to Integrating LESs

Conceptually, the method used to integrate LESs into learning environments is based on the diagram shown in Figure 1. The top of the figure shows the pedagogical approaches (LESs and traditional LS approach) that are integrated and applied to the class activities and learning content. Instructors can decide to use whichever approach(es) best suite their learning environment, and they can use the contents of Table 1 to help identify which approach(es) they are using. By combining the LESs and the lecture style approach to learning in different ways, we expect that instructors will maximize student learning and engagement over time and possibly tailor different combinations to different types of learners. Note there is a loop in the figure between combining pedagogical approaches and increased student learning and engagement, which signifies that the process is not linear but involves trying different combinations of approaches and achieving the best result for a set of learners. In Section 4 we describe how we quantify the pedagogical approaches used in a Software Testing class.

The main objective of the proposed method is through research find the values of p_i, in the following expression, that most improve student learning and engagement,

$$\mathrm{TM}-\mathrm{C}(p_1\mathrm{CL} + p_2\mathrm{GA} + p_3\mathrm{PBL} + p_4\mathrm{SI} + p_5\mathrm{LS}). \tag{1}$$

The term TM-C in Expression (1) represents the context in which the LESs and traditional lecture style are used for a given teaching modality. The teaching modality may be F2F, online, hybrid (blended), or synchronous remote [49, 70]. Currently, the p_i variables in Expression 1 represent the percentages of the time dedicated to each pedagogical approach. However, we expect that in the future, the p_i variables will represent the broader concept of teaching effort dedicated to using each pedagogical approach (LESs or LS).

Basing our method in the context of cognitive load theory [10, 13], our objective is to improve a student's knowledge on a topic. To this end, using an LES in the wrong context can do more harm than good. Cognitive load theory is usually described in terms of three types of cognitive load, intrinsic, extraneous, and germane. Kalyuga [40] describes the different types of cognitive load as follows. *Intrinsic load* is related to the learning task and the level of the learner experience, and it is independent of the instructional design methods used with the task. *Extraneous load* is related to

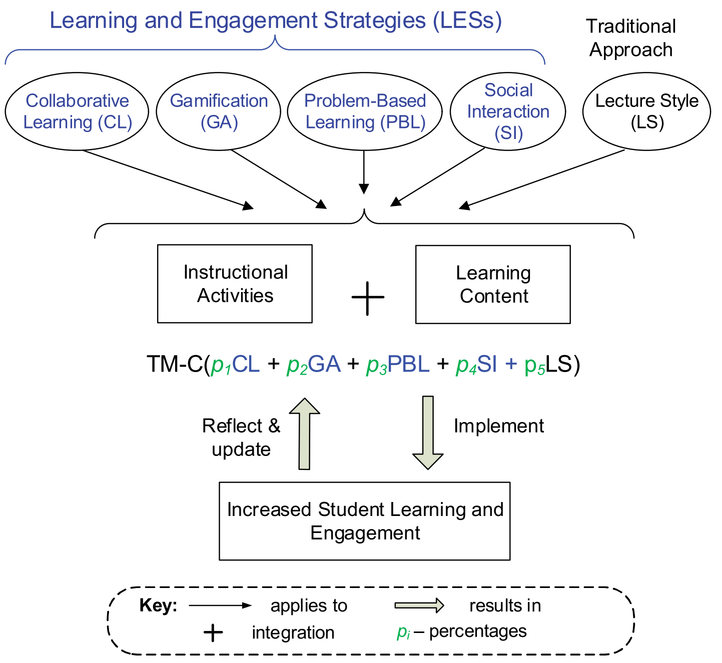


Fig. 1. LES integration model.

LES Integration Method]Model showing how LESs and the lecture-style approach are integrated into class activities and learning content to increase student learning and engagement.

the cognitive processes not required for learning and is activated by an instructional design method that is not optimal. *Germane load* may be thought of as the working memory resources allocated to dealing with the intrinsic load that results in learning. To this end, good instructional design methods should reduce the extraneous load and effectively manage the intrinsic (and germane) load to optimize learning.

Another aspect of learning that should be considered when deciding on integrating LESs and a lecture-style approach is the role that unconscious processing plays in the cognitive process. Kuldas et al. [43] state that aligning instructional design with learner expertise can improve learning by optimizing cognitive load. Another dimension that may impact learning is how to motivate learners to better use their available cognitive capacity. The key to addressing this problem may be the unconscious learning process. Kuldas et al. [43] further state that empirical evidence shows that knowledge construction mostly takes place in unconscious perceptual and emotional functions. Therefore when integrating LESs and lecture-style, one should consider instructional design methods that decrease extraneous load, effectively managing intrinsic load, and motivate learning using unconscious perceptual and emotional functions.

An example that nicely captures the guideline provided above is using gamification in class activities. Using a gamification activity that entails a complex game with many players and levels (extraneous load), students may gain little or no knowledge after the game is over (poor management of intrinsic load). However, to improve student learning, some LESs may need to be used in concert to enhance student learning. As an example, using additional social interaction activities with collaborative learning and problem-based learning can improve learning by providing a stronger motivation to learn. It should be pointed out that the method used to integrate LESs shown in Figure 1 is *extensible* and *adaptable*, that is, over time we expect the number of LESs to change based on the targeted learners.

4 PEDAGOGICAL APPROACH USING LESs IN A SOFTWARE TESTING CLASS

In this section, a brief overview of software testing is provided, the structure of the software testing course is given, and a description of how LESs are integrated into the F2F and online class activities is provided. The software testing course previously referenced is *CEN4072 Fundamentals of Software Testing*, an FIU undergraduate course. The interested reader can find additional details on software testing and the software testing course in Clarke et al. [16].

4.1 Software Testing

The main objective of software testing is to determine if the behavior of an implementation (program) is what is expected based on the specification (requirements). This objective is achieved by selecting inputs from the usually infinite domain and comparing the actual output from the implementation to the expected output from the specification [48]. The most common software testing approaches are black-box and white-box testing [48]. Black-box testing, sometimes referred to as *specification-based testing*, generates test cases based on the specification of the component without considering its implementation. White-box testing, sometimes referred to as *implementation-based testing*, guides the generation of test cases based on properties of the implementation [48]. White-box testing techniques are usually associated with test adequacy criteria, which are generally associated with coverage metrics, such as the control-flow coverage and data-flow coverage. The black-box testing techniques [48] taught in the CEN4072 class include the following:

- *equivalence partitioning:* partitions the input domain into sub-domains that tend to observe the properties of equivalence classes and inputs are selected from each sub-domain;
- *boundary value analysis:* selects test input at the boundaries of equivalence classes; and
- *state-based:* **finite state machines (FSM)** are created to represent the behavior of a system during its design and can later be used to generate test inputs based on various coverage criteria of the FSM.

The white-box testing techniques [48] taught in the CEN4072 class include the following:

- *control-flow* (statement coverage, branch coverage and multiple condition coverage); and *data-flow* (all-definitions coverage and all-uses coverage).

Some of the tools used in the CEN4072 class to support testing include *JUnit*, a free unit testing framework for the Java programming language that supports black-box testing of object-oriented classes [29]; *IBM's Rational Functional Tester*, a commercial automated functional and regression testing tool that supports black-box testing of a system [36]; and *EclEmma*, a free Java code coverage tool for web and desktop applications that supports white-box testing [34].

4.2 Software Testing Course

CEN4072 Fundamentals of Software Testing is the only undergraduate software testing course at FIU, and it is offered during the fall, spring, and summer semesters. The course catalog description is as follows: test plan creation, test case generation, program inspections, black-box testing, white-box testing, GUI testing, and the use of testing tools. The prerequisite for CEN4072 is the Data Structures course. The grade for the course is based on two midterms exams (25% each), a group project (25%), attendance and class participation (5%), and a final exam (20%). The course is structured as a lecture section with no lab component, and students are expected to work on the project outside of class on their own time, with some in-class question sessions. The textbook currently used in the course is "Foundations of Software Testing" by Aditya P. Mathur [48]. Other reading material includes class notes and tutorials on testing tools.

The course topics are grouped into three parts: before the first midterm exam (Exam 1) and first project deliverable; after Exam 1 and before the second midterm exam (Exam 2); and after Exam 2, and before the final exam and second project deliverable. Topics covered in each of the three sections are as follows: before Exam 1 (introduction to the software process model, basic testing definitions and concepts, review of object-oriented characteristics, design by contract and defensive design, documenting the testing process, and black-box testing techniques (equivalence partitioning, boundary value analysis, state-based)); after Exam 1 and before Exam 2 (flow control graphs, control-flow coverage (statement, branch, and multiple condition), identification of definition-use pairs using a flow control graph, and data-flow coverage (all definitions and all uses)); after Exam 2 and before the final exam (integration testing, strategies to minimize stubs (mocks), systems testing, and regression testing).

The objective of the course project is to provide students with the experience of analyzing and validating software, writing test cases, creating test documents, and working in teams. The software to be tested during the project is a software application written by a previous software engineering class (undergraduate or graduate). For additional details on the software testing course, including details of the class project, the interested reader can peruse Clarke et al. [16].

4.3 Integrating LESs into Class Activities

To this point in the article, we have described LESs, the characteristics of LESs and the lecture-style approach, and the structure of the software testing course (CEN4072). We now describe how LESs are integrated into the CEN4072 course for the F2F and the online class activities at FIU. Note that the online activities are done using SEP-CyLE. However, these activities can be done using any learning management system, e.g., Canvas [37] or Blackboard [4], with some additional work. The CEN4072 is taught in an active learning classroom with five whiteboards and a wireless connection to the instructor's projector.

F2F Activities: The LESs used in the F2F activities are integrated using the following approach:

- *Collaborative Learning:* occurs both at the class level and at the team level. The characteristics used to distinguished collaborative learning from the other approaches are shown in Rows 4, 5, 6, 8, 9, 10(a), 10(b) of Table 1. At the class level, students present solutions and answer questions during a class session that may benefit the entire class, e.g., writing test cases for a given problem specification. Collaborative learning at the team level involves team members solving problems in class using the whiteboards. The students initially solve problems on the whiteboards in their assigned project teams but are then rotated to other teams to interact with other students in the class.
- *Gamification:* during the F2F class activities, students are awarded exam points both individually and at the team level if they correctly answer questions posed by the instructor. The characteristics used to distinguish gamification are shown in Rows 10(e), 17, 19, 21, and 22 of Table 1. For example, a student may be rewarded with one exam point if they can correctly identify a black-box testing technique. Each team member may be awarded two exam points if the team can collectively identify the inputs to a method that generates 100% statement coverage using a control flow graph. The solution to such a problem would be written on the whiteboards in the classroom. If a student finds an error with the solution presented by another team, then that student gets an exam point, and the team presenting the solution gets no points. Note there is a maximum of 10 exam points a student can use towards each exam, and each student in the class is provided an opportunity to answer a question for points.

- PBL: involves students working in their teams to solve problems. As stated in Section 2.1, these problems are usually real-world problems, usually ill-structured and open-ended, and make students responsible for their learning. The characteristics that distinguish PBL from the other LESs are Rows 4, 5, 6, 8, 9, 10(a), 10(b), 10(c) and *most importantly* Row 7. The PBL approach is usually encountered when working on the class project, which is testing an application created by students in another class. The students in their teams experience many of the testing techniques and tools used to test the software before they are discussed in the class.
- *Social Interaction:* similarly to collaborative learning, social interaction occurs both at the class level and at the team level. The distinguishing characteristics for social interaction are shown in Rows 11, 12, 13, 14, 15, 16, 17, and 18. At the class level, students interact socially in F2F class activities by working on the problems provided by the instructor, and conducting project team meetings in and out of class. During the in-class activities, all students are required to assess the correctness of the solutions placed on the whiteboards by a team (see GA above). Although assessing the correctness of solutions on the whiteboards does involve collaborative learning, it tends to be more of a social interaction activity, since there is more interaction between the other students in the class and the instructor. Social interaction also occurs during the project presentations, since a student in each team must ask the team presenting a question. The student asking the question is required to state their name, team number, and role in the team. In addition, the students in a project team are required to rotate the managerial roles (team leader, minute taker, and timekeeper) of the project between themselves for the two phases of the project.

Since there are three exams in the course, students are given the opportunity to earn exam points before each of the exams. If a student earns more than 10 exam points, then the extra points are *not* carried forward to the next exam.

Online Activities: The online activities are centered on using the features in SEP-CyLE and involve some combination of the LESs: collaborative learning, gamification, and social interaction. Currently, we have not yet integrated the environment required for problem-based learning into SEP-CyLE. It is worth stating that the points awarded in SEP-CyLE are virtual and are different from the exam points for F2F in-class activities. The LESs integrated into the online class activities use the following approach:

- *Collaborative Learning:* Students are placed in virtual teams (usually the same as the F2F project teams) and are required to complete digital learning objects or tutorials individually, but are rewarded as a team (see GA below). Students may also post learning content in SEP-CyLE, or another learning management system, that benefits the entire class. Learning content is considered beneficial to the class if students post messages on the forum stating the content was helpful or positively rated.
- *Gamification:* Students are rewarded with virtual points based on the completion of various learning activities (digital learning objects or tutorials) both individually and as a team (assuming collaborative learning is enabled). Below are examples of how points may be awarded in SEP-CyLE:
  - —Each student receives 10 points if she/he completes a digital learning object and scores 70% or above (or some other specified value) on the learning object graded assessment quiz; two points for writing on the discussion board, and one point for uploading a picture.
  - —Each member of a team receives five points if all members of the team score 70% or higher on a learning object assessment quiz; 10 bonus points for the first team to complete a

Table 2. Scenarios Showing How the Class Time was Computed (in Minutes) for Each Pedagogical Approach using the Lesson Plan, Class Activities, and the Number of Slides for a Given F2F Class

Scenario	Lesson Plan	# Slides	Time (mins)				
			CL	GA	PBL	SI	LS
1	Class 1: New topics presented: course syllabus, project overview, overview of the software process, and terminology; formation of teams;	16	0	0	0	20	55
2	Class 5: Review of class 4; new topics presented—test adequacy criteria and test documentation	28	5	5	5	5	55
3	Class 11: Exam 1 review session	0	45	15	5	10	0
4	Class 16: Part 1 of project presentations for Deliverable 1—include slide presentations and testing demonstrations by each student team	0	0	0	60	15	0

CL - Collaborative Learning, GA - Gamification, PBL - Problem-Based Learning, SI - Social Interaction, LS - Lecture Style.

learning object assessment quiz with all team members scoring 70% or higher; five bonus points for the second team; and three bonus points for the third team.
—There is a leader board showing the top 5 students in the class (which may or may not be anonymized). Each student can see their progress on each assigned digital learning object and tutorial. We recommend using the anonymized option in SEP-CyLE to mask the student names if the virtual points count towards the course grade.

- *Social Interaction:* Students post learning content that help other students in the class (see collaborative learning), rate and comment on the contents of learning objects and tutorials, and upload a picture. We are still working on a rating system to automatically determine how valuable a post may be to the other students in the class, other than using student ratings.

Students in the CEN4072 class are usually assigned 10 learning objects and four tool tutorials, and some combinations of the three LESs currently available in SEP-CyLE are enabled. The virtual points students earned during the semester are translated into a small number of course percentage points (e.g., 2 or 3 percentage points) and added to the course grade after all the other points for the course (exams, project deliverables, and class participation) are computed. Some instructors in the project use the learning objects in SEP-CyLE as weekly assignments and explicitly include the virtual points as part of the course grade.

4.4 Quantifying the Class Time for Pedagogical Approaches

This section describes how the class time for each of the LESs and the lecture-style approach was quantified using a set of scenarios from actual classes. Although there may be more fine-grained methods to determine the class time for each approach used in class, e.g., the use of a COPUS-like instrument [63], our method seemed appropriate for the study presented in the next section. The method used to quantify the class time for each LES and lecture style's was computed in a post-facto manner utilizing the lesson plan, class activities, and the number of slides presented during class. In the subsequent text, we describe scenarios that clarify our approach.

Table 2 shows scenarios for four CEN 4072 classes (1, 5, 11, and 16) from fall 2018 with the class time (in minutes) for each pedagogical approach. Note each class has a duration of 75 minutes. As an example, Class 1, shown in Row 1 of the table, represents the first class of the semester where the course syllabus is presented, an overview of the class project is presented, and an overview of the software process is presented. During this class, time is spent creating teams where students exchange contact information and set up their social media accounts. Most of this class is dedicated to a presentation by the instructor (55 mins) using lecture-style, and the remaining time is spent creating teams, assigning team roles, and students getting to know each other.

Unlike Class 1, Class 16 (Scenario 4) is dedicated to the first part of the project presentations for Deliverable 1. Since the class project has the characteristics of a problem typically used in PBL, most of the time in the class is dedicated to the PBL approach (60 mins), and the remaining time is dedicated to social interaction. The social interaction time is required, since each student team is required to ask a question after each presentation, and the student asking the question must introduce themselves to the class by providing their name, team number, and role in the team.

5 EXPERIMENTAL STUDY

This section describes the study that was performed to determine the impact of integrating LESs into F2F class activities. The study's description includes the research questions, methods (sample, data collection, and design of the study), results and analysis, and a discussion of the findings in the context of the research questions.

5.1 Research Questions

Identifying how best to integrate LESs into F2F and online class activities to improve student learning and engagement is expected to be a long term research endeavor. This integration process involves selecting the LESs to be used and quantifying how much of each LES is used in the class activities. In this study, we initiate the process of finding how best to integrate LESs into F2F class activities, starting with the software testing class (CEN4072) at FIU. As previously mentioned, we posit there is no single approach to integrating LESs in all CS/IT/SE classes. Learning occurs differently in classes based on the attributes of a given class, such as the year the course is being taught, the content being taught, the size of the class, and aptitude of students, among others. The research questions we investigate in this study are as follows:

RQ1: Does increasing LESs in the F2F class activities of a software testing class improve student learning?
RQ2: Assuming there is an increase in student learning in the software testing class, what combination of LESs resulted in the positive impacts on student learning?
RQ3: Does the extra credit (Exam points) awarded to students in the software testing class reflect the increased use of LESs in F2F class activities?

5.2 Methods

Sample: Students from seven (7) *CEN4072 Fundamentals of Software Testing* classes at FIU participated in the study, see Table 3. The control group included 60 students from fall 2018, spring 2019, and summer 2019 CEN4072 classes who were minimally exposed to LESs between Exam 1 and Exam 2 in F2F class activities, see Row 2 in Table 3 with entry *Control (MinLESs)* in Column 1. The treatment group included 62 students from the spring 2017, summer 2017, fall 2017, and spring 2018 classes who were fully exposed to LESs between Exam 1 and Exam 2, see Row 4 in Table 3 with entry *Treatment (LESs)* in Column 1.

Data Collection: The data collected during the study include the following: the scores on the midterm exams (Exams) and the extra points awarded before Exam 1 and after Exam 1 and before Exam 2. The extra points earned by the students in the classes were recorded manually by the instructor and added to the exam scores after the exams were graded.

Table 3. Samples Used in the Control and Treatment Groups, Showing the Semesters and Class Enrollments for the Software Testing Classes

Group	Semester CEN4072 Class	Enrollment
Control (MinLESs)	Fall 2018	20
	Spring 2019	27
	Summer 2019	13
	Total	60
Treatment (LESs)	Spring 2017	17
	Summer 2017	15
	Fall 2017	11
	Spring 2018	19
	Total	62

Over the semesters listed in Table 3, Exam 1 and Exam 2 assess the students' knowledge on different topics in the CEN4072 course. However, across semesters Exam 1 assessed similar software testing topics, similarly to Exam 2. For example, the first question on Exam 1 is always to define various software testing terms, e.g., the definition of *software testing*. The structure of Exam 2 is as follows:

Q1: defining several software testing terms.
Q2: drawing a diagram to show the testing process for the class project or explaining additional terms.
Q3: drawing a flow graph for a method and identifying test inputs for the following code coverage criteria: statement coverage, branch coverage and multiple condition coverage.
Q4: identifying definition-use pairs for variables in the method in Q3, and identifying test inputs for all definitions coverage or all uses coverage criteria; and
Q5: performing strong and weak mutation testing using the method in Q3.

For this study, Exam 2 in fall 2018, summer 2019, and spring 2018 were identical. Similarly, Exam 2 for spring 2019 and fall 2017 were identical.
Design: We use a quasi-experimental posttest-only design [20] to determine the impact of using MinLESs (control) versus full use of LESs (treatment) in the software testing class. The posttest measure used in the study is the grades of Exam 2 for both control and treatment groups. Although there is no pretest used in this type of design, we include Exam 1 to gauge the students' abilities in the two software testing class groups at the start of the study. During the first class of each semester in which the study was conducted, students were informed of the study and were allowed to opt-out of the study without penalty. The study presented in this article is covered under IRB Exemption number IRB-15-0220 approved by the Office of Research and Integrity at FIU.

Table 3 shows that the control group was set up after the treatment group, which is not typical for most educational studies. The authors decided to take this approach, since the LES integration method shown in Figure 1 has been implemented over several semesters, as reflected in the loop shown at the bottom of Figure 1. After the use of LESs in the classroom for several semesters became relatively constant, we decided to start the study with the treatment group. After the use of LESs stabilized, we could have conducted the study beginning with the control group followed by the treatment group, but this would not have been a true reflection of the work performed on the project.

Table 4. Approximate Class Meeting Time (in Minutes) and Percentages Dedicated to Each LES and Lecture Style for the Class Interval between Exam 1 and Exam 2

Pedagogical Approach	Control (MinLESs)		Treatment (LESs)	
	Time (mins.)	Percentage	Time (mins.)	Percentage
Lecture Style (LS)	380	51%	165	22%
Collaborative Learning (CL)	65	9%	240	32%
Gamification (GA)	60	7%	90	12%
Problem-based Learning (PBL)	170	23%	170	23%
Social Interaction (SI)	75	10%	85	11%

The total class meeting time for this interval was 750 minutes.

The intervention in the study is the exposure of students in the treatment group to higher percentages of LESs and a lower percentage of the traditional lecture-style approach when teaching a class. Table 4 shows the approximate amount of class meeting time and the percentage of the total amount of meeting time dedicated to each LES and the traditional lecture-style approach used during the study. The study was conducted during the semester after Exam 1 and before Exam 2, consisting of 750 minutes total meeting time (5 weeks of the semester). These class meeting times in the table were computed as described in Section 4.4. Examples of the class meeting times (in minutes) for Week 7 (first week of the study) for the control and treatment groups are as follows: *Control*: LS, 100 (67%); CL, 15 (10%); Gamification (GA) - 15 (10%); PBL, 10 (7%); and SI, 10 (7%); and *Treatment*: LS, 60 (40%), CL, 40 (26%); GA, 30 (20%); PBL, 10 (7%); and SI, 10 (7%). While the control group remained relatively constant with respect to class times for LESs and lecture-style during the study, in Week 11 (last week of the study), the times for the treatment group changed considerably see Table 2, Scenario 3.

The LESs are defined in Section 2.1 and a description of how they are integrated into the class activities is in Section 4.3. The description of the integration of LESs into class activities involves some overlap; however, we minimize this overlap by adhering to the characteristics presented in Table 1. The main differences between the control group and the treatment group are as follows.

(1) The number of problems students worked on as part of the in-class activities. These problems include mainly problems that focus on the basic concepts of software testing and exam review problems. In the treatment group, students collaboratively worked on these problems thereby increasing the class time for collaborative learning and social interaction. While in the control group, the instructor worked on the problems using the traditional lecture-style approach, thereby maintaining a high percentage of class time for lecture style.
The same amount of time was spent on problem-solving in both groups. However, the strategies were different, as previously described. In addition, both groups were given access to past exam papers to practice their problem-solving skills. Note there was little change in the time spent on problem-based learning due to the nature of the class project.
(2) The interval between Exam 1 and Exam 2 includes the project's first presentation, which consisted mainly of problem-based learning and social interaction, see Table 2, Scenario 4.

Based on the results of preliminary studies conducted in prior semesters, students in the control group (MinLESs) were given an extra credit question providing them with an opportunity to gain additional exam points. These additional points were used to make up for any points the students may have lost due to the pedagogical approaches used (MinLESs). Note that these additional points *are not* included in the Exam 2 scores for the control group reported in Section 5.3. As previously stated, both the control and treatment groups were exposed to SEP-CyLE, which impacted the

Table 5. Student Data Including Mean Scores and Standard Deviation (M (SD)) on Exams 1 and 2 and Extra Points Earned toward Each Exam

Group	Class	N	Student Grades (%) Exam 1* M (SD)	Student Grades (%) Exam 2* M (SD)	Extra Points Exam 1$^+$ M (SD)	Extra Points Exam 2$^+$ M (SD)
Control (MinLESs)	FA18	20	80.5 (7.7)	*73.0 (10.6)*	2.2 (2.2)	1.2 (1.7)
	SP19	27	82.0 (8.2)	69.6 (10.4)	2.9 (2.2)	2.1 (2.9)
	SU19	13	78.3 (7.3)	*77.0 (11.4)*	3.8 (3.6)	2.4 (2.1)
		60	**80.7 (7.8)**	**72.4 (10.9)**	**2.9 (2.6)**	**1.9 (2.4)**
Treatment (LESs)	SP17	17	71.8 (11.2)	73.6 (12.8)	1.5 (1.9)	4.9 (3.8)
	SU17	15	79.7 (10.9)	82.1 (13.5)	3.2 (2.7)	7.9 (1.9)
	FA17	11	80.0 (6.3)	81.0 (12.3)	4.3 (3.9)	5.5 (3.5)
	SP18	19	78.8 (11.0)	*76.1 (13.0)*	3.4 (3.7)	5.2 (3.6)
		62	**77.3 (10.3)**	**77.7 (13.1)**	**3.0 (3.2)**	**5.8 (3.4)**

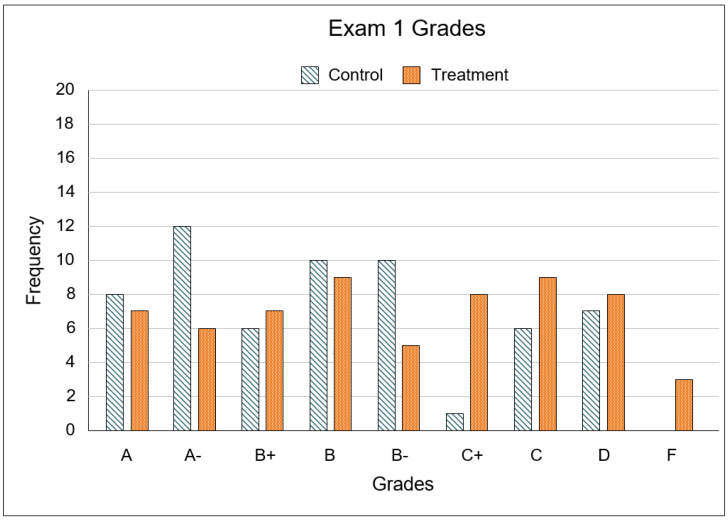
Exam 1* and Exam 2* do not include the extra points earned in class.
Exam 1$^+$ extra points earned before Exam 1, and
Exam 2$^+$ extra points earned between Exam 1 and Exam 2.
Green and italicized show the identical Exams.
Purple and underlined show the identical Exams.

classes' online activities. These online activities were *the same* and required the completion of 10 digital learning objects and four tutorials during the semester. Since the conditions when using SEP-CyLE were the same in both groups, we did not expect that using LESs in online class activities would impact the study results.

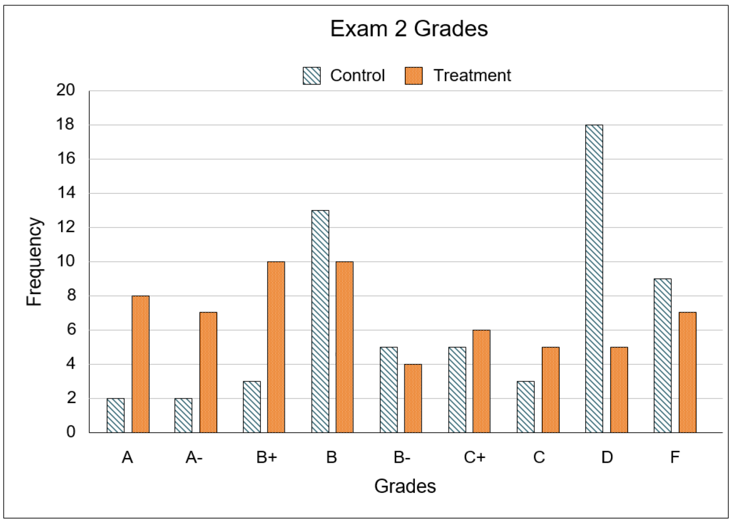
5.3 Results and Analysis

A summary of the exam scores and the extra points obtained by students in the control and treatment groups are shown in Table 5. The table consists of seven columns, which are from left to right: group identifier, classes participating in the study, number of students (N), the mean and standard deviation (M(SD)) of the student scores for the two exams (Exam 1* and Exam 2*), and the mean and standard deviation for extra points awarded in class (Exam 1$^+$ and Exam 2$^+$). The "*" indicates the exams do not include the points awarded in class, and the "+" the period when the extra points were awarded in the class. It is worth stressing that the exams were two *different exams* that are testing two completely different sets of topics. Therefore, it is impossible to use the scores in Exam 1 and Exam 2 to measure knowledge gain, hence the posttest-only design.

The initial reading of the results in Table 5 indicates that the overall scores on Exam 2 were lower than on Exam 1, and that the treatment group did better on Exam 2 (mean = 77.7%, standard deviation = 13.1) than the control group (mean = 72.4%, standard deviation = 10.9). The table also shows that the control group did better on Exam 1 (mean = 80.7%, standard deviation = 7.8) than the treatment group (mean = 77.3%, standard deviation = 10.3). The in-class points earned by the students in both groups before Exam 1 are similar as reflected in their mean values, control group 2.9 (standard deviation = 2.6) and treatment group 3.0 (standard deviation = 3.2), respectively. However, there is a more significant difference in the points earned between Exam 1 and Exam 2. The control group having a mean of 1.9 (standard deviation = 2.4), and the treatment group a mean of 5.8 (standard deviation = 3.4).



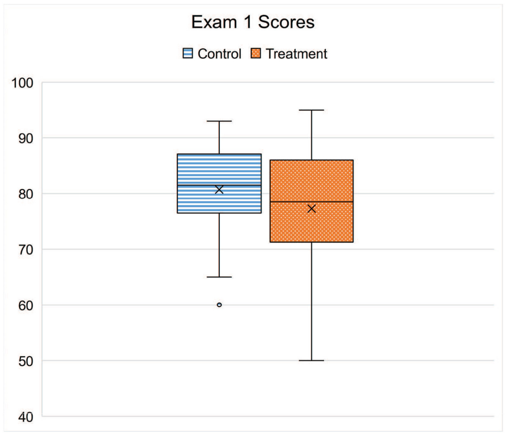
(a)



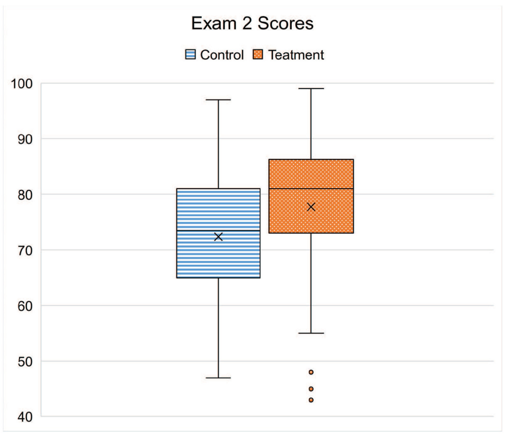
(b)

Fig. 2. Bar charts showing the grade distribution for the control and treatment groups for (a) Exam 1 and (b) Exam 2.

Figure 2 shows the grade distributions for the control and treatment groups for (a) Exam 1 and (b) Exam 2. The range of grades for the exam are A, A-, B+, B, B-, C+, C, D, F. As reflected in the figure, students in the control group did better on Exam 1 than those in the treatment group. However, the reverse is true for the Exam 2 grades. Figure 2(b) shows an outlier with the number of students obtaining a grade of D in the control group in Exam 2 (18 students). Figure 3 shows the overall variability of the data for the Exam 1 and Exam 2 scores using boxplots. The plots also show the median, inter-quartile range, and outliers for the control and treatment groups for both exams. Figure 3(a) shows that the control group has a larger median than the treatment group, and the



(a)



(b)

Fig. 3. Boxplots showing the median, inter-quartile range and outliers for the control and treatment groups for (a) Exam 1 and (b) Exam 2.

variability for the treatment group's scores is larger than those of the control group. There is one outlier for the control group, but it does not affect the overall variability compared to the treatment group. In Figure 3(b), the median value is higher for the treatment group, and the variability is similar for both groups, with the treatment group having four outliers (only three are visible).

A more in-depth statistical analysis was conducted using the data collected during the study, and the results are as follows. To answer Research Question 1 (*RQ1*) we performed a statistical analysis on the exam scores for both groups. The Exam 2 scores are not normally distributed as specified by the Shapiro–Wilk test for normality ($p < 0.05$, skew = −0.518, kurtosis = −0.250). To determine if the Exam 2 scores between the control and treatment groups were statistically significant, we conducted a Mann–Whitney U test. The Exam 2 scores for the control (MinLESs) group (*Mdn = 72.4*) *does* differ significantly from the treatment (LESs) group (*Mdn = 77.7*), where U = 2421, z = 2.875, p = 0.004, r = 0.26. Note that the mean rank for the treatment group is 70.55 and for the control group is 52.15, which supports the fact that the treatment group performed better than the control group on Exam 2. Therefore we can conclude that the treatment group performed significantly better than the control group with a small to medium effect size (r = 0.26).

A statistical analysis of the Exam 1 scores prior to administering the treatment was performed to see if there was a statistically significant difference between the two groups. Similarly to the Exam 2 scores, the Exam 1 scores are not normally distributed using the Shapiro–Wilk test ($p < 0.05$, skew = −0.707, kurtosis = 0.197). The Mann–Whitney U test for the Exam 1 scores of the control (MinLESs) group (*Mdn = 80.7*) *does not* differ significantly from the treatment (LESs) group (*Mdn = 78.5*) , where U = 1530, z = −1.691, p = 0.091. The mean rank for Exam 1 shows that the control group is 67.0 while the treatment group is 56.2. Therefore, we can conclude that there is *no* statistically significant difference between the Exam1 scores for the control and treatment groups.

To answer *RQ3* we performed a similar analysis for the extra points awarded to students in the control and treatment groups. Using the Shapiro–Wilk test for normality showed that neither the points awarded before Exam 1, or between Exam 1 and Exam 2 are normally distributed. The Mann–Whitney U test for the extra points awarded before Exam 1 for the control (MinLESs) group (*Mdn = 2*) *does not* differ significantly from the treatment (LESs) group (*Mdn = 2*), where U = 1797, z = −0.324, p = 0.746. However, the extra points between Exam 1 and Exam 2 for the control group (*Mdn = 1*) *does* differ significantly from the treatment (LESs) group (*Mdn = 6*) , where U = 3077,

$z = -6.291$, $p = 0.000$. This result is reflected in the mean ranks of the control and treatment groups, 41.22 and 81.13, respectively. It can be concluded that due to the increased use of gamification students were able to obtain more extra points towards Exam 2 in the treatment group than the control group.

5.4 Discussion

This section discusses the results presented in the previous section and draws conclusions related to the research questions stated in Section 5.1.

RQ1: Does increasing LESs in F2F class activities improve student learning? Based on the results shown in Table 5 and the statistical analysis of the results there is a statistically significant difference between the Exam 2 scores for the control and the treatment groups. The Exam 2 scores on average for the classes in the control group was 72.4 and for the treatment group was 77.7, implying that the students in the treatment group performed better on Exam 2 as compared to the control group. To determine if increasing the use of LESs in F2F class activities impacted student learning we use the Expression 1 and substitute p_i with the values in Table 4 representing the percentage of time allocated to each LES and LS (traditional lecture style). The resulting expressions for integrating LESs into F2F class activities between weeks 7 and 11, during the intervention, are as follows: control - `F2F-MinLESs(0.09CL + 0.07GA + 0.23PBL + 0.10SI + 0.51LS)` and treatment - `F2F2F-LESs(0.32CL + 0.12GA + 0.23PBL + 0.11SI + 0.22LS)`. Although several studies reported in the literature show reducing the level of lecture-style pedagogy increases student learning, few, if any, have quantified the pedagogical approaches used as shown above.

RQ2: What combination of LESs resulted in the positive impacts on student learning? Since there was an increase in student learning as reflected in the Exam 2 scores, we can answer *RQ2* using the specific class time dedicated to each LES in our study. Note that we only report one specific set of percentages for LESs for the software testing classes at FIU, given the approach described in Section 4.3. Using the expressions presented when discussing *RQ1*, the treatment group increased the time dedicated to LESs during the study: CL by 270% (65 mins to 240 mins); GA by 50% (60 mins to 90 mins); and SI by 13% (75 mins to 85 mins). Problem-based learning was unchanged between the control and treatment groups due to the nature of the group project. There was a decrease in time dedicated to LS by 56% (380 mins to 165 mins). We stress that these results should be interpreted in the context of the F2F class activities in the software testing classes at FIU that participated in the study and using the approach described in Section 4.3.

RQ3: Does the extra credit (Exam points) awarded in class reflect the increased use of LESs in F2F class activities? Similar to the answer for *RQ1*, the results shown in Table 5 and the statistical analysis performed on these results show there is a statistically significant difference between the Exam 2 points awarded to students in the control and the treatment groups. This difference is reflected in the p value for the Mann–Whitney U test being less than 0.01 and the mean ranks being 41.22 and 81.13 for the control and treatment groups, respectively. The points awarded before Exam 1 do not show a statistically significant difference between the control and treatment groups, with the p value being 0.746. This result is obtained since the gamification LES was used for the entire semester. The increase in extra credit points between Exam 1 and Exam 2 supports the increased use of LESs in the treatment group. More specifically, the increases in collaborative learning (270%) and gamification (50%), and to a lesser extent, gamification (13%). The increases in collaborative learning, gamification, and social interaction were mainly due to the increased problem-solving activities related to software testing concepts and applying software techniques to toy problems.

Threats to Validity. One of the main internal threats to the study was the method used to determine the class time spent on different pedagogical approaches during the studies. As previously mentioned, the approach used to determine the time spent on each LES was based on a review of the lesson plan, class activities, and the number of slides presented during class. A more accurate approach would be to have a software application that can be used during the class and activated by the instructor (or research assistant) to record the time taken when different pedagogical approaches were used. Other fine-grained methods, like the use of an instrument like COPUS [63], could also provide additional details related to the class activities' times and their nature. Another factor that compounds this threat is that the boundaries between some active learning approaches have become blurred over time as practitioners move away from the initial guidance on how to use these approaches [73]. To ameliorate this factor, we performed a detailed literature review that resulted in Table 1, showing the nuances between the various pedagogical approaches used in the study.

Using the exams to measure student learning may also pose a threat to internal validity. We used seven different classes in our study, which made it impractical to give all the classes the same exams. However, we attempted to mitigate this threat by keeping the concepts related to each question in the exams the same. Some of the exams were the same as described in the following text. For Exam 1: fall 2018 (control) was the same as spring 2017 (treatment); spring 2019 (control) and summer 2019 (control) were the same as spring 2018 (treatment). For Exam 2: fall 2018 (control) and summer 2019 (control) were the same as spring 2018 (treatment); spring 2019 (control) was the same as fall 2017 (treatment).

The external threats to validity include the following: generalizing the results of the study for other software testing classes; the threats that are associated with quasi-experimental studies, i.e., not selecting participants for the study using a random approach; and the aptitude of the students in classes participating in the study. Generalizing the effect of integrating LESs in other software testing classes may be possible for classes with similar characteristics to the FIU classes. These characteristics include class size, availability of active learning classrooms, and the instructor's experience using LESs in the classroom. On average, the study's class size was equal to or less than 30 students, which allowed for posing questions to individual students for about three to five minutes in each class. Each team contained about four to five students for collaborative learning activities and required five whiteboards to solve problems. The external threats related to the instructor's experience for our study were mitigated as follows. All the classes in our study were taught by the same instructor who has been teaching the software testing class for over ten years and is very familiar with implementing LESs in F2F class activities, as described in Section 4.3.

6 CONCLUSIONS AND FUTURE WORK

This article presents a method that can be used to integrate LESs into F2F and online class activities. These LESs include *collaborative learning*, *gamification*, *problem-based learning* and *social interaction*. The method used to integrate LESs into class activities was developed based on a project involving LESs in a cyberlearning environment (SEP-CyLE).[2] SEP-CyLE provides vetted learning content in the form of digital learning objects and tutorials in the context of a selected combination of LESs. Although previous studies in the literature have focused on integrating LESs for online class activities using SEP-CyLE, this article is the first major work to report on a comprehensive study that focuses mainly on integrating LESs into F2F class activities. In addition, the article describes an approach to integrating LESs into both F2F and online class activities.

A study is presented that reports on integrating LESs into F2F class activities that spanned seven semesters and was conducted at a large urban university. Three research questions are identified, and a quasi-experimental posttest-only design was used to determine if using LESs in F2F class

[2]https://stem-cyle.cis.fiu.edu/.

activities had any effect on student learning. The results showed that students exposed to the increased use of LESs in F2F class activities performed significantly better on the second exam than those who did not. To determine if there were any threats from student aptitude between the different groups, we performed a similar statistical analysis on the first exam, and there was no statistically significant difference. To reflect the increased use of LESs, we stated the estimated class times dedicated to each LES and the traditional lecture-style approach.

We plan to repeat the study described in this article using different teaching modalities and a more accurate approach to determining the class times used for each LES. Using the results from these studies, we will compare them to determine if using the LESs as prescribed in this article improves student learning. We are currently conducting studies for synchronous remote classes due to changes implemented as a result of the COVID-19 pandemic. We expect that calculating the class times will be more accurate, since all the lectures were recorded and have been archived. We also plan to conduct qualitative studies to determine if the students are aware of the changes in the pedagogical approaches used in the classroom. To date, we have completed several focus groups with classes using the synchronous remote modality and will analyze the data shortly.

The method used in the study to integrate LESs into class activities is adaptable and extensible. We expect that the use of various LESs will be different based on several factors including, course content, student year (freshmen through senior year), class size, and the context in which the LESs are used. We expect that these factors will be investigated in future studies at a cross-section of intuitions in courses offered in different years of CS/IT/SE programs.

REFERENCES

[1] Albert Bandura. 1977. *Social Learning Theory* (2nd ed.). Prentice-Hall, Englewood Cliffs. NJ.

[2] Howard S. Barrows and Robyn M. Tamblyn. 1980. *Problem-based Learning: An Approach to Medical Education.* Vol. 1. Springer, New York.

[3] Jens Bennedsen and Michael E. Caspersen. 2019. Failure rates in introductory programming: 12 years later. *ACM Inroads* 10, 2 (Apr. 2019), 30–36. https://doi.org/10.1145/3324888

[4] Blackboard Inc. 2019. Blackboard. Retrieved September 2019 from http://www.blackboard.com/.

[5] Donald A. Bligh. 1998. *What's the Use of Lectures?* (5th ed.). Intellect, Exeter, England.

[6] Kim B. Bruce. 2018. Five big open questions in computing education. *ACM Inroads* 9, 4 (Nov. 2018), 77–80. https://doi.org/10.1145/3230697

[7] Jerome S Bruner. 1961. The act of discovery. *Harv. Educ. Rev.* 32 (1961), 21–32.

[8] Ingrid A. Buckley and Peter J. Clarke. 2018. An approach to teaching software testing supported by two different online content delivery methods. In *Proceedings of the 16th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Innovation in Education and Inclusion.* LACCEI, Lima, Peru.

[9] Kelsey Hood Cattaneo. 2017. Telling active learning pedagogies apart: From theory to practice. *J. New Approach. Educ. Res.* 6, 2 (2017), 144–152. https://doi.org/10.7821/naer.2017.7.237

[10] Paul Chandler and John Sweller. 1991. Cognitive load theory and the format of instruction. *Cogn. Instruct.* 8, 4 (1991), 293–332. https://doi.org/10.1207/s1532690xci0804_2

[11] Raymond Chang-lau and Peter J. Clarke. 2018. Software Engineering and Programming Cyberlearning Environment (SEP-CyLE). Retrieved from https://stem-cyle.cis.fiu.edu/instances.

[12] Juanjuan Chen, Minhong Wang, Paul A. Kirschner, and Chin-Chung Tsai. 2018. The role of collaboration, computer use, learning environments, and supporting strategies in CSCL: A meta-analysis. *Rev. Educ. Res.* 88, 6 (2018), 799–843. https://doi.org/10.3102/0034654318791584 arXiv:https://doi.org/10.3102/0034654318791584

[13] Ruth Colvin Clark, Frank Nguyen, John Sweller, and Melissa Baddeley. 2006. Efficiency in learning: Evidence-based guidelines to manage cognitive load. *Perf. Improve.* 45, 9 (2006), 46–47. https://doi.org/10.1002/pfi.4930450920 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/pfi.4930450920.

[14] Peter J. Clarke, Debra Davis, Tariq M. King, Jairo Pava, and Edward L. Jones. 2014. Integrating testing into software engineering courses supported by a collaborative learning environment. *Trans. Comput. Educ.* 14, 3, Article 18 (Oct. 2014), 33 pages. https://doi.org/10.1145/2648787

[15] Peter J. Clarke, Debra L. Davis, Ingrid A. Buckley, Geoffrey Potvin, Mandayam Thirunarayanan, and Edward L. Jones. 2019. An approach to integrating learning and engagement strategies (LESs) into CS class activities. In *Proceedings of the 126th American Society for Engineering Education (ASEE'19).* ASEE, Washington DC.

[16] Peter J. Clarke, Debra L. Davis, Raymond Chang-Lau, and Tariq M. King. 2017. Impact of using tools in an undergraduate software testing course supported by WReSTT. *ACM Trans. Comput. Educ.* 17, 4, Article 18 (Aug. 2017), 28 pages. https://doi.org/10.1145/3068324

[17] Benjamin S. Clegg, Jose Miguel Rojas, and Gordon Fraser. 2017. Teaching software testing concepts using a mutation testing game. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET'17).* IEEE, Los Alamitos, CA, 33–36. https://doi.org/10.1109/ICSE-SEET.2017.1

[18] Allan Collins, John Seely Brown, and Susan E Newman. 1988. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *J. Philos. Childr.* 8, 1 (1988), 2–10.

[19] Computing Research Association. 2017. Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006. Retrieved August 2020 from https://cra.org/data/Generation-CS/.

[20] John W. Creswell. 2014. *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research* (4th ed.). Prentice Hall, Upper Saddle River, NJ.

[21] Neil Davidson and Claire Howell Major. 2014. Boundary crossings: Cooperative learning, collaborative learning, and problem-based learning. *J. Excell. Coll. Teach.* 25, 3&4 (2014), 7–55.

[22] Edward L. Deci and Richard M. Ryan. 1985. *Intrinsic Motivation and Self-determination in Human Behavior.* Plenum, New York.

[23] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From game design elements to gamefulness: Defining "gamification." In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek'11).* Association for Computing Machinery, New York, NY, 9–15. https://doi.org/10.1145/2181037.2181040

[24] John Dewey. 1963. *Experience and Education.* Macmillan, New York.

[25] Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. 2015. Gamification in education: A systematic mapping study. *Educ. Technol. Soc.* 18, 3 (2015), 75–88.

[26] Adrián Domínguez, Joseba Saenz de Navarrete, Luis de Marcos, Luis Fernández-Sanz, Carmen Pagés, and José-Javier Martínez-Herráiz. 2013. Gamifying learning experiences: Practical implications and outcomes. *Comput. Educ.* 63 (2013), 380–392. https://doi.org/10.1016/j.compedu.2012.12.020

[27] Debra M. Duke, Mandayam Thirunarayanan, Abigail Byram, and Peter J. Clarke. 2019. Students' perceptions of the implementation of a cyberlearning tool. In *Proceedings of the 126th ASEE Annual Conference & Exposition—Computing and Information Technology Division.* ASEE, Washington, D.C.

[28] Scott Freeman, Sarah L. Eddy, Miles McDonough, Michelle K. Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proc. Natl. Acad. Sci. U.S.A.* 111, 23 (2014), 8410–8415.

[29] E. Gamma and K. Beck. 2018. JUnit. Retrieved August 2018 from http://www.junit.org/.

[30] Edward F. Gehringer. 2007. Active and collaborative learning strategies for teaching computing. In *Proceedings of the ASEE Annual Conference & Exposition.* ASEE, Washington DC, 13.

[31] Anthony F. Grasha. 1994. A matter of style: The teacher as expert, formal authority, personal model, facilitator, and delegator. *Coll. Teach.* 42, 4 (1994), 142–149. https://doi.org/10.1080/87567555.1994.9926845

[32] Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does gamification work?–A literature review of empirical studies on gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences.* IEEE Computer Society, Los Alamitos, CA, 3025–3034.

[33] Pedro Henrique Dias Valle, Armando Maciel Toda, Ellen Francine Barbosa, and José Carlos Maldonado. 2017. Educational games: A contribution to software testing education. In *Proceedings of the IEEE Frontiers in Education Conference (FIE'17).* IEEE, Los Alamitos, CA, 1–8. https://doi.org/10.1109/FIE.2017.8190470

[34] Marc R. Hoffmann, Brock Janiczak, and Evgeny Mandrikov. 2018. EclEmma. Retrieved August 2018 from http://www.eclemma.org/.

[35] Beth Hurst, Randall Wallace, and Sarah B. Nixon. 2013. The impact of social interaction on student learning. *Read. Horiz.* 52, 4 (2013), 375–398.

[36] IBM. 2018. Rational Functional Tester. Retrieved August 2018 from https://www.ibm.com/us-en/marketplace/rational-functional-tester.

[37] Instructure. 2019. Canvas. Retrieved September 2019 from https://www.instructure.com/canvas.

[38] Liu Junhua, Yue Zhang, Justin Ruths, Diana Moreno, Daniel D. Jensen, and Kristin L.Wood. 2013. Innovations in software engineering education: An experimental study of integrating active learning and design-based learning. In *Proceedings of the 120th ASEE Annual Conference & Exposition.* ASEE, Washington, DC, 45.

[39] Kamaruzaman Jusoff and Siti Akmar Abu Samah. 2012. *Social Interaction Learning Styles.* Springer US, Boston, MA, 3101–3104. https://doi.org/10.1007/978-1-4419-1428-6_1785

[40] Slava Kalyuga. 2011. Cognitive load theory: How many types of load does it really need? *Educ. Psychol. Rev.* 23, 1 (2011), 1–19.
[41] Judy Kay, Michael Barg, Alan Fekete, Tony Greening, Owen Hollands, Jeffrey H. Kingston, and Kate Crawford. 2000. Problem-based learning for foundation computer science courses. *Comput. Sci. Educ.* 10, 2 (2000), 109–128. https://doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109
[42] Karel Kreijns, Paul A. Kirschner, and Wim Jochems. 2003. Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: A review of the research. *Comp. Hum. Behav.* 19, 3 (2003), 335–353.
[43] Seffetullah Kuldas, Shahabuddin Hashim, Hairul Ismail, and Zainudin Bakar. 2015. Reviewing the role of cognitive load, expertise level, motivation, and unconscious processing in working memory performance. *Int. J. Educ. Psychol.* 4, 06 (2015), 142–169. https://doi.org/10.17583/ijep.2015.832
[44] Per Lauvås and Andrea Arcuri. 2018. Recent trends in software testing education: A systematic literature review. In *UDIT (The Norwegian Conference on Didactics in IT Education).* Bibsys Open Journal Systems, Norway, 1–11.
[45] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation.* Cambridge University Press, Cambridge, UK.
[46] Ilaria Liccardi, Asma Ounnas, Reena Pau, Elizabeth Massey, Päivi Kinnunen, Sarah Lewthwaite, Marie-Anne Midy, and Chandan Sarkar. 2007. The role of social networks in students' learning experiences. *SIGCSE Bull.* 39, 4 (Dec. 2007), 224–237. http://doi.acm.org/10.1145/1345375.1345442
[47] Thomas W. Malone. 1980. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems (SIGSMALL'80).* ACM, New York, NY, 162–169. https://doi.org/10.1145/800088.802839
[48] Aditya P. Mathur. 2014. *Foundations of Software Testing* (2nd ed.). Pearson Education, India.
[49] Barbara Means, Yuki Toyama, Robert Murphy, Marianne Bakia, and Karla Jones. 2009. *Evaluation of Evidence-based Practices in Online Learning: A Meta-analysis and Review of Online Learning Studies.* Technical Report. U.S. Department of Education.
[50] Mourya Reddy Narasareddygari, Gursimran S. Walia, Debra M. Duke, Vijayalakshmi Ramasamy, James Kiper, Debra Lee Davis, Andrew A. Allen, and Hakam W. Alomari. 2019. Evaluating the impact of combination of engagement strategies in SEP-CyLE on improve student learning of programming concepts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19).* ACM, New York, NY, 1130–1135. https://doi.org/10.1145/3287324.3287413
[51] Mourya Reddy Narasareddygari, Gursimran S. Walia, and Alex Radermacher. 2018. Using gamification and cyber learning environment to improve students' learning in an introductory computer programming course: An empirical case study. In *Proceedings of the 120th ASEE Annual Conference & Exposition.* ASEE, Washington, DC, 13.
[52] National Science Board. 2018. Science and Engineering Indicators 2018. Retrieved August 2020 from https://nsf.gov/statistics/2018/nsb20181/.
[53] Sandra Y. Okita. 2012. Social interactions and learning. In *Encyclopedia of the Sciences of Learning*. Springer US, Boston, MA, 3104–3107. https://doi.org/10.1007/978-1-4419-1428-6_1770
[54] Jean Piaget. 1952. *The Origins of Intelligence in Children.* International Universities Press, New York.
[55] Carl R. Rogers and H. Jerome Freiberg. 1994. *Freedom to Learn* (3rd ed.). Merrill, New York.
[56] Richard M. Ryan and Edward L. Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemp. Educ. Psychol.* 25, 1 (2000), 54–67. https://doi.org/10.1006/ceps.1999.1020
[57] John R. Savery. 2006. Overview of problem-based learning: Definitions and distinctions. *Interdisc. J. Probl.-Bas. Learn.* 1, 1 (2006), 9–20.
[58] John R Savery and Thomas M Duffy. 1995. Problem based learning: An instructional model and its constructivist framework. *Educ. Technol.* 35, 5 (1995), 31–38.
[59] H. G. Schmidt. 1993. Foundations of problem-based learning: Some explanatory notes. *Med. Educ.* 27, 5 (1993), 422–432. https://doi.org/10.1111/j.1365-2923.1993.tb00296.x
[60] Swapneel Sheth, Jonathan Bell, and Gail Kaiser. 2015. A gameful approach to teaching software design and software testing. *Comput. Games Softw. Eng.* 9 (2015), 91.
[61] Simon, Andrew Luxton-Reilly, Vangel V. Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. 2019. Pass rates in introductory programming and in other STEM disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR'19).* Association for Computing Machinery, New York, NY, 53–71. https://doi.org/10.1145/3344429.3372502
[62] Barbara Leigh Smith and Jean T. MacGregor. 1992. What is collaborative learning? In *Collaborative Learning: A Sourcebook for Higher Education.*, A. S. Goodsell, M. R. Maher, and V. Tinto (Eds.). National Center on Postsecondary Teaching, Learning, and Assessment, University Park, PA.

[63] Michelle K. Smith, Francis H. M. Jones, Sarah L. Gilbert, and Carl E. Wieman. 2013. The classroom observation protocol for undergraduate STEM (COPUS): A new instrument to characterize university STEM classroom practices. *CBE—Life Sci. Educ.* 12, 4 (2013), 618–627. https://doi.org/10.1187/cbe.13-08-0154

[64] Rachel S. Smith. 2004. Guidelines for authors of learning objects. The New Media Consortium. Retrieved from http://archive2.nmc.org/guidelines/NMCLOGuidelines.pdf.

[65] Neelam Soundarajan, Swaroop Joshi, and Rajiv Ramnath. 2015. Collaborative and cooperative-learning in software engineering courses. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2 (ICSE'15).* IEEE Press, Los Alamitos, CA, 319–322.

[66] LaJoy Renee Spears. 2012. *Social Presence, Social Interaction, Collaborative Learning, and Satisfaction in Online and Face-to-face Courses.* Ph.D. Dissertation. Iowa State University.

[67] M. Stains, J. Harshman, M. K. Barker, S. V. Chasteen, R. Cole, S. E. DeChenne-Peters, M. K. Eagan, J. M. Esson, J. K. Knight, F. A. Laski, M. Levis-Fitzgerald, C. J. Lee, S. M. Lo, L. M. McDonnell, T. A. McKay, N. Michelotti, A. Musgrove, M. S. Palmer, K. M. Plank, T. M. Rodela, E. R. Sanders, N. G. Schimpf, P. M. Schulte, M. K. Smith, M. Stetzer, B. Van Valkenburgh, E. Vinson, L. K. Weir, P. J. Wendel, L. B. Wheeler, and A. M. Young. 2018. Anatomy of STEM teaching in North American universities. *Science* 359, 6383 (2018), 1468–1470. https://doi.org/10.1126/science.aap8892

[68] C. Stephenson, A. Derbenwick Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. Cannon Mindell, L. Limbird, D. Richardson, M. Sahami, E. Villa, H. Walker, and S. Zweben. 2018. *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions.* ACM, New York, NY.

[69] Allison Elliott Tew and Mark Guzdial. 2010. Developing a validated assessment of fundamental CS1 concepts. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10).* Association for Computing Machinery, New York, NY, 97–101. https://doi.org/10.1145/1734263.1734297

[70] Ngoc Thuy Thi Thai, Bram De Wever, and Martin Valcke. 2020. Face-to-face, blended, flipped, or online learning environment? Impact on learning performance and student cognitions. *J. Comput. Assist. Learn.* 36, 3 (2020), 397–411. https://doi.org/10.1111/jcal.12423

[71] U.S. Bureau of Labor Statistics. 2020. Occupational Outlook Handbook (OOH). Retrieved August 2020 from https://www.bls.gov/ooh/computer-and-information-technology/home.htm.

[72] Lev S. Vygotsky. 2012. *Thought and Language.* MIT Press, Cambridge, MA.

[73] Andrew Walker, Heather Leary, Cindy Hmelo-Silver, and Peggy A. Ertmer (Eds.). 2015. *Essential Readings in Problem-based Learning: Exploring and Extending the Legacy of Howard S. Barrows.* Purdue University Press, West Lafayette, IN.

[74] Stuart Zweben and Betsy Bizot. 2020. 2019 Taulbee Survey. Computing Research Association. Retrieved from https://cra.org/wp-content/uploads/2020/05/2019-Taulbee-Survey.pdf.

Received February 2021; revised July 2021; accepted August 2021