DEVELOPING AND VALIDATING ACTIVE LEARNING ENGAGEMENT STRATEGIES TO IMPROVE STUDENTS' UNDERSTANDING OF PROGRAMMING AND SOFTWARE ENGINEERING CONCEPTS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Tamaike Mariane Brown

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Computer Science

May 2020

Fargo, North Dakota

North Dakota State University
Graduate School

Title

DEVELOPING AND VALIDATING ACTIVE LEARNING ENGAGEMENT STRATEGIES TO IMPROVE STUDENTS' UNDERSTANDING OF PROGRAMMING AND SOFTWARE ENGINEERING CONCEPTS

By

Tamaike Mariane Brown

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia
Chair

Dr. Oksana Myronovych

Dr. Jeremy Straub

Dr. Jacob Glower

Approved:

May 29, 2020
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

Introductory computer programming course is one of the fundamental courses in computer science. Students enrolled in computer science courses at the college or university have been reported to lack motivation, and engagement when learning introductory programming (CS1). Traditional classrooms with lecture-based delivery of content do not meet the needs of the students that are being exposed to programming courses for the first time. Students enrolled in first year programming courses are better served with a platform that can provide them with a self-paced learning environment, quicker feedback, easier access to information and different level of learning content/assessment that can keep them motivated and engaged. Introductory programming courses (hereafter referred to as CS1 and CS2 courses) also include students from non-STEM majors who struggle at learning basic programming concepts. Studies report that CS1 courses nationally have high dropout rates, ranging from anywhere between 30-40% on an average. Some of the reasons cited by researchers for high dropout rate are lack of resource support, motivation, lack of engagement, lack of motivation, lack of practice and feedback, and confidence. Although the interest to address these issues in computing is expanding, the dropout rate for CS1/CS2 courses remains high. The software engineering industry often believes that the academic community is missing the mark in the education of computer science students. Employers recognize that students entering the workforce directly from university training often do not have the complete set of software development skills that they will need to be productive, especially in large software development companies.

ACKNOWLEDGEMENTS

This document is based on research conducted at North Dakota State University between 2018 – to present. I am grateful for a number of assistance received to start this, continue with it and finally to publish it.

At North Dakota State University, I thank Dr. Gursimran Walia for academic support.

I am grateful to my supervisory committee members, especially Dr. Oksana Myronovych and Dr. Jeremy Straub from North Dakota State University Computer Science Department and Dr. Jacob Glower from Electrical and Computer Engineering Department for providing their feedback.

This work is supported in part by the National Science Foundation under grants DUE-1225742 and DUE-1525112. 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.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIX FIGURES

1. INTRODUCTION

Introductory computer programming course is one of the fundamental courses in computer science education. Students enrolled in computer science courses at the college or university have been reported to lack motivation, and engagement when learning introductory programming (CS1). Students and instructors have reported that transition from natural language to machine language understanding can be challenging. Additionally, traditional classrooms with lecture-based delivery of content do not meet the needs of the students that are being exposed to programming courses for the first time [41, 49, 43, and 22]. Students enrolled in the first year programming courses are better served with a platform that can provide them with a self-paced learning environment, quicker feedback (either automated or from instructors), easier access to information (information on the go) and different level of learning content/assessment that can keep them motivated and engaged.

Additionally, introductory programming courses (hereafter referred to as CS1 and CS2 courses) also include students from non-STEM majors who struggle at learning basic programming concepts. Studies report that CS1 courses nationally have high dropout rates, ranging from anywhere between 30-40% on an average [55]. Some of the reasons cited by researchers for high dropout rate are lack of resource support, motivation, lack of engagement, lack of motivation, lack of practice and feedback, and confidence [56]. Although the interest to address these issues in computing is expanding, the dropout rate for CS1/CS2 courses remains high [57]. The software engineering industry often believes that the academic community is missing the mark in the education of computer science students [58, 59]. Employers recognize that students entering the workforce directly from university training often do not have the

complete set of software development skills that they will need to be productive, especially in large software development companies [58].

One of the suggested solutions to mitigate this problem in computer science and software engineering is to provide students with an engaging and motivating learning environment using game mechanics in a cyber-learning environment. Researchers have begun using game mechanics to improve students' engagement and learning in higher education [60]. Early finding suggests that the use of gamification elements (GEs) such as competition and rewards to engage students in learning has potential to be a very successful [61].

Gamification is defined as the application of game design and elements in educational settings in order to positively influence students' motivation and engagement [62, 63, and 64]. Fitz-Walter et al reported that applications that use gamification elements have a positive impact on students because it motivates students to learn and explore [65]. Connolly et al carried out a review that aimed to identify research evidence about positive impacts of games. The review confirmed that playing games is linked to a range of perceptual, cognitive, behavioral affective and motivational impacts and outcomes. The most frequently occurring outcomes and impacts were knowledge acquisition/content understanding and affective and motivational outcomes [66]. Barata et al did a study on improving participation and learning with gamification, the results show significant improvements in terms of attention to reference materials and online participation of students [67]. Due to earlier findings about the impact of gamification in education, the terminology rapidly became the next big thing in the education realm and gained widespread attention in 2010 as a learning tool [68, 69].

To this end, as part of this dissertation, a group of CS educators (NDSU and 7 other collaborating US institutions) has led the development of a cyber-learning environment

(supported by NSF awards) to help improve students' fundamental understanding of software engineering and introductory computer programming concepts. The cyber-learning environment presented in this dissertation is hereafter called SEP-CyLE (Software Engineering and Programming – A Cyber learning environment).

SEP-CyLE includes embedded learning and engagement strategies (e.g., gamification and collaborative learning) and contains a repository of vetted learning objects (LOs) and tutorials. Researchers and Educators can use a combination of different learning engagement strategies (LESs) to motivate students to be more involved in learning Software Engineering and programming concepts [9]. Current LESs embedded in SEP-CyLE include collaborative learning, gamification, and social interaction. The learning content in SEP-CyLE is presented in the form of digital learning objects (LOs), and video tutorials. LOs are a small amount of multimedia learning content (may be in the form of text or videos) on a specific topic and are designed to be completed within fifteen minutes. Each LO contains content, a practice assessment and a quiz submission. The online environment (SEP-CyLE) allows students to upload images (avatar), gain virtual points after completing an LO, post comments on discussion boards and monitor activities of other students (leaderboard). Figure 1 shows a block diagram of the major features of SEP-CyLE. The features of SEP-CyLE include authentication, LESs (collaborative learning, gamification, and social interaction), learning content, administration and course management.

Mourya et al. showed that two main drawbacks of SEP-CyLE is that LESs such as collaborative learning, the way it is currently implemented in SEP-CyLE, requires improvement and more gamification elements need to be incorporated in the cyberlearning platform that span across STEM disciplines [83]. Collaborative learning which is targeted around virtual groups taking part in varied SEP-CyLE activities needs to be improved in such a way that it allows

students to work in teams to complete a single task seamlessly. In this dissertation, one way we are exploring how to improve collaborative learning is by incorporating peer code review (PCR) in CS1/CS2 courses. Although there is no single accepted definition of PCR, a common theme includes "a process to check programming tasks performed by others to identify mistakes (that could be seeded) [86]. This dissertation will focus on evaluating if PCR is a good educational tool (both individually and collaboratively) and the impact frequent availability of LOs has on students learning of introductory programming concepts.

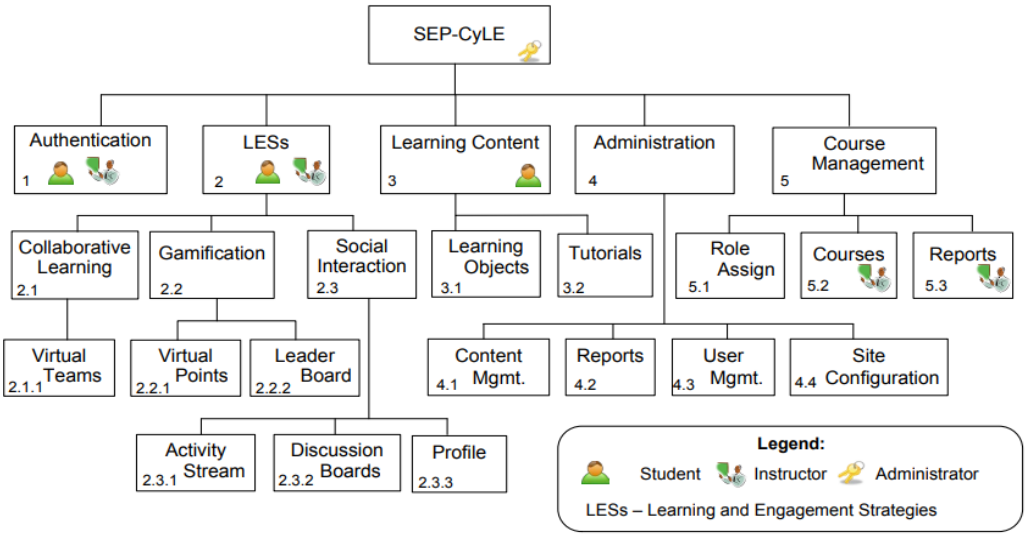


Figure 1: Block Diagram of SEP-CyLE

1.1. Research Goals

Cyber-learning can have a major impact on pedagogy through frequent resource support and learning via collaboration [25, 26]. A meta-analysis of research data reported on online learning conducted by the US Dept. of Ed. revealed that students are more engaged on cyber-learning platforms and tend to perform better than their counterparts in traditional classrooms [27]. Khan Academy is a noteworthy cyber-learning environment that has found widespread success for

a variety of courses backed by studies that show that cyber-learners are more engaged, motivated, and willing to attempt more difficult tasks [28].

Since the introduction of SEP-CyLE, there have been multiple studies [9, 82, 83, 84] reported SEP-CyLE having a positive impact on student knowledge, engagement, and pedagogy support in CS and SE courses, particularly at North Dakota State University. These studies can be accessed at https://stem-cyle.cis.fiu.edu/publications. In this dissertation study, the researcher will build up on work done by Mourya et al., and leverage SEP-CyLE with a focus on improving LESs and measuring their impact on student learning.

To further analyze how SEP-CyLE can support CS/SE pedagogy, the primary goal of this dissertation is to gain insights into different ELES of SEP-CyLE and its impact on students' learning. To explore ways in which SEP-CyLE design can be improved, this dissertation will identify additional gamification elements (GEs) based on prior evidence of its effectiveness in STEM (Science, Technology, Engineering and Mathematics) education. Additionally, the current state of SEP-CyLE is limited in the nature of collaboration. My dissertation will explore the use of peer code review (PCR) as an effective collaborative technique that can be incorporated in SEP-CyLE.

Specifically, my dissertation goals are: 1) To provide a comprehensive empirical analysis of Gamification elements that can aid student learning and implement additional gamification elements in SEP-CyLE; 2) To investigate if peer code review (PCR) is an effective educational collaborative tool that can used in SEP-CyLE to improve fundamental learning of programming concepts and improves programming ability of CS students;

2. LITERATURE REVIEW

This chapter provides a background of pedagogical approaches at teaching CS/SE concepts in a cyber-learning environment and face-to-face computer science classroom. This chapter also motivates the usage of learning engagement strategies (LESs) utilized as a part of SEP-CyLE. Specifically, this section is focused on related work on Gamification elements and Peer Code Review (individual and peer based) as potential learning and engagement strategies that can be used within SEP-CyLE. Additionally, learning objects (LOs) developed as part of SEP-CyLE development is also discussed.

A Learning Object (LO) is an assortment of content, practice questions, and assessment based on a single learning objective to support learning. LOs are available in various structures including content, audio, and video and are created to serve as easily digested chunks of self-contained and re-usable knowledge that could be finished in the time of fifteen minutes or less [82, 83, 84]. Past SEP-CyLE studies in CS1 classrooms have shown that using LOs in SEP-CyLE improves students' learning when compared against traditional classroom instruction. The studies have also found that LOs in SEP-CyLE when enabled with diverse Learning Engagement strategies (Gamification (G), Social Interaction (SI), and Collaborative Learning (CL)) had a big impact on-student learning of programming and software testing concepts [82, 83]. Specifically, results from prior studies showed that the combination of Gamification and social interaction (G+SI) had the largest impact on student learning. Most interestingly, students were more actively engaged in their own learning [84]. While LOs have found to be useful, the impact of the frequency of their assignment and availability to the students can provide additional insights into structuring of these LOs in the context of other learning content that is covered in CS1/CS2 courses.

This dissertation attempts to provide those insights through empirical validation of LO assignment in CS1 courses at NDSU.

Peer Code Review is defined as a process to check programming tasks performed by others to identify mistakes [86]. Peer review is widely adopted and studies in many science disciplines as an effective quality assurance activity. Studies have shown that peer code review helps students in programming concepts knowledge acquisition and transfer of knowledge when completing a programming task. In an empirical study conducted by Hundhausen et al. [87], researchers describe a peer code review team activity with three-course assignments that varied in length and complexity. Students reviewed their members' code against an established checklist of coding best practices, which contained a list of requirements for the specific programming solution being reviewed. The team carefully classified and documented each issue that was raised and discussed with the team. The authors concluded that code reviews improved the quality of students' code, stimulated increasingly sophisticated discussions of programming issues and practices, and promoted a sense of community [87].

Wang et al. [88, 89] describes a slightly different PCR approach that consists of six processes: 1-write, 2-submit, 3-review, 4-feedback, 5-revise and 6-quality assurance. Using this approach, authors conducted an experiment in two academic years of a second year CS course. In this experiment, the PCR approach was performed through email, with only a teaching assistant and an instructor performing phase 3, 4 and 6. The results showed that students with strong dedication to learning programming wrote their programs carefully and positively. These students also had shorter feedback comments as the complexity of the program increases [89].

In another peer code review experiment technique developed by Trytten, the instructor selected one student's assignment for other peers to review. Students review the code by using a

15 true or false questionnaire focus on evaluating students' understanding of the code. At the end, students shared their responses via flashcards and discussed the reason for their selection. The authors observed that this technique was able to improve lab attendance [90].

2.1. Gamification

Gamification is defined as the application of game design and elements in educational settings in order to positively influence students' motivation and engagement [62, 63, and 64]. Gamification elements and their usage can be used to evaluate the user experience. Some of the most common gamification elements used in F2F or online classes include rewards/points, levels and achievements, and leaderboards [64, 73]. Deterding et al., noted that points, levels, achievements and other gamification elements could raise the level of user pleasure due to an increase in experience competence [73]. Since students typically report a scarcity of engagement when learning introductory computer science courses, gamification has the potential to advance student motivation and extend engagement while providing feedback on the student's level of competence of the learned material [74].

In educational settings, learning activities have traditionally been gamified with two primary purposes:

(1) To encourage desired learning behaviors, for example, following software engineering best practices, fostering the participation of students into learning communities, or advancing active participation in peer assessment [5, 9, 10, 11, 14, 17, 18, 19];

(2) To improve engagement of students in learning, for example by the utilization of learning materials like tutorials or digital tools [2, 4, 7, 12, 19, 29]. The effects on gamification in education shows that it is a promising pedagogical technique.

> Majority studies show that gamification has contributed to an improvement in students' understanding and engagement in the classroom. Particularly leaderboards were the most motivating GEs, followed by badges and points. Studies conjointly found a major increase within the performance of the students once utilizing these gamification elements (GEs) [9, 14, 18].

Incorporating gamification in introductory CS courses and SE courses can assist students in learning introductory CS / SE concepts and subsequently increase students' retention rate in the field. This dissertation is focused at conducting a comprehensive analysis of most common gamification elements that can be used to support CS1/CS2 pedagogy.

2.2. Collaborative Learning

Collaboration is recommended for improvement of students' motivation and attitudes towards online learning. It addresses different problems such as assisting students in the development of skills expected by industry such as critical thinking, communication skills, teamwork and engaging students in learning while having fun [80]. Collaborative learning can contribute to students' sense of belonging in their discipline (and reduces their risk of leaving college or choosing another discipline [41, 81]).

While collaborative learning has shown to improve students' attendance and engagement, Mourya et al., showed that collaborative learning (the way it is implemented in SEP-CyLE) did not have a major impact on student learning and engagement when compared to Gamification or Social interaction [83]. One of the major goals of this research is to explore means of improving collaboration when learning programming in CS1/CS2 courses. On that end, Code Reviews, at individual and while working with peers, is being explored as a potential collaboration and

engagement technique that can be integrated in SEP-CyLE. The motivation and background on Code Reviews is explained below.

At North Dakota State University (NDSU), Pair Programming has shown to be an effective tool at fostering communication among programmers, improving student engagement and quality of student code [91]. Pair programming includes two programmers that work collaboratively at one computer, where one code and the other review and they take turns. However, pair programming is limited to two people, lacks individual assessment and requires a lot of planning at identifying working pairs [92]. Our past Pair Programming exercises had provided evidence that when students are reviewing the same problems, they tend to be more engaged and tend to gain knowledge through discussion with their peers. This dissertation is taking crucial aspects of Pair Programming (peer code review) but expanding it to the class level where all students are reviewing the same set of code that are seeded with realistic errors, see Figure 2. The results of the studies show that incorporating PCR in CS1/CS2 face-to-face classroom improvement in students' understanding of basic programming concepts and eventually improved their programming abilities. Additionally, the studies revealed that when students participate in a collaborative active learning environment they learn from each other through social constructs [93].

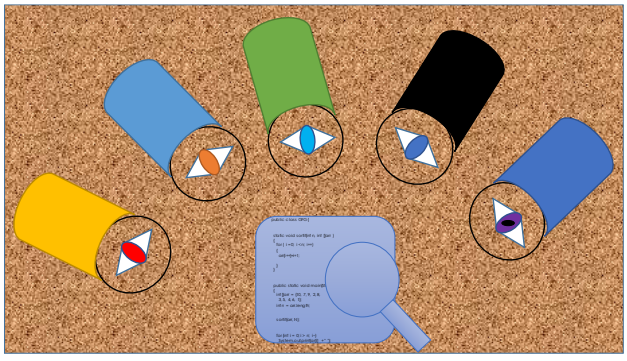


Figure 2: Diagram of Peer Code Review

2.3. SEP-CyLE – Software Engineering and Programming Cyber Learning Environment

While Chapter 1 provides some details on the SEP-CyLE, this section provides a summary of the prior work that has been done at developing and evaluating SEP-CyLE. Since the introduction of SEP-CyLE, there have been some studies that have reported the positive impact the SEP-CyLE had on student knowledge, engagement, and pedagogical support in CS and SE courses. One study result have indicated that the use of SEP-CyLE with gamification elements can positively impact students' performance and engagement in an introductory programming course [83]. In another study, results presented demonstrated that there is a meaningful relationship between the gamification and social interaction learning strategies used in SEP-CyLE. When the virtual points were analyzed, it was found that the students were more productive at completing LO's and those students who earned more virtual points received better grades [82]. However, published research on SEP-CyLE lacks: (1) additional gamification elements that could be of further use in SEP-CyLE (the published study only included two GEs, points and leaderboard. These studies can be accessed at https://stem-cyle.cis.fiu.edu/publications), (2) advanced level programming courses and evaluate how these different learning engagement strategies improve advanced understanding of software programming, and (3) effective implementation of collaborative learning strategy [82, 83].

In this dissertation study, studies will build up on work done by Mourya et al., and leverage SEP-CyLE with a focus on improving LESs, with a special focus on enhancing LOs, providing a deeper understanding of the gamification elements to support CS pedagogy, and improving collaborative learning through Peer Code Review. The researcher believe that the peer code review

(PCR) can be effectively used in SEP-CyLE to improve fundamental learning of programming concepts and improve programming ability of CS students.

3. GAMIFICATION IN CS/SE EDUCATION: RESULTS FROM THE SYSTEMATIC LITERATURE REVIEW

This Section details the systematic literature review conducted to identify and analyze the gamification elements (GEs) that are used in STEM wide discipline and can be used to improve SEP-CyLE impact on students' knowledge gains of CS1/CS2 programming concepts.

3.1. Systematic Literature Review (SLR)

The SLR was conducted to identify major Gamification Elements and their impact on student learning in STEM. This systematic review is based on guidelines established by Kitchenham and Charters in Guidelines for Performing Systematic Literature Reviews in Software Engineering [85]. A systematic literature review is a means of assessing and interpreting all the accessible research applied to a specific research question, area of interest. The following steps were implemented in accordance with the guidelines for a systematic literature review established by Kitchenham and Charters [85].

3.2. SLR: Study Procedure

- Step 1: Definition of research questions. The researcher have defined four research questions based on the study goal, to establish the desired scope of the SLR (systematic literature review) (section 3.2)
- Step 2: Perform search - based on the research questions the researcher defined search string for retrieving papers in selected scientific databases (section 3.3)
- Step 3: Study selection – the researcher applied inclusion and exclusion criteria for selecting only the relevant papers for the study (section 3.4)
- Step 4: Study execution - This outlines the process that has been applied to include the primary studies (section 3.5)

- Step 5: Data extraction – the researcher extracted the data from the papers finalized from step 4 (section 3.6)

3.3. SLR: The Goal and Research questions

The main objective of this study is to analyze literature for the purpose of understanding and evaluating gamification elements and their impacts on students' motivation and engagement in STEM courses. In order to answer the main goal the researcher formulated the following research questions:

3.4. Research Questions

- ***RQ1:*** *What are most commonly reported gamification elements in STEM education?*
- ***RQ2:*** *What disciplines in STEM use gamification?*
- ***RQ3:*** *What is the evidence of impact of game elements on student learning and student engagement?*
- ***RQ4:*** *How can answers to RQ1 and RQ2 be incorporated into the design of cyber learning environments in CS/SE education?*

The First question explores the use of GEs in STEM education. The second question is to identify the STEM disciplines that use gamification. The third research question is constructed to separate the GEs that have been empirically validated from those which are based on anecdotal evidence. The fourth question aims towards improving the design of SEP-CyLE.

3.5. SLR: Search Strategy

To identify possible primary studies for data extraction, the search was based on (i) trial searches using combinations of keywords derived from the study goal for the definition of valid search strings. Initially, the researcher selected relevant keywords related to the following

domains: (a) education; (b) software engineering; (c) computer science; (d) STEM; (e) Science; (f) Technology; (g) Engineering; (h) Mathematics and (i) gamification. In order to identify these relevant keywords, the researcher considered search strings used in related systematic studies, bodies of knowledge, and experts (software engineering educators). Search strings were defined by grouping keywords in the same domain with the logic operator "OR" and grouping the domains with the logic operator "AND". This resulted in the following search string.

(Game or gamification) OR (elements or element) AND (effect or impact or result) AND (education or learning or engagement or course or lecture) AND (STEM or Science or Math or Technology or Engineering or computer science or software engineering)

This study reviews papers published in scientific journals and proceedings of international conferences. Electronic databases identified as relevant to education were searched in this review, namely: ACM (Association for Computing Machinery) Digital Library, IEEE (Institute of Electrical and Electronics Engineers) Xplore, ProQuest, and Web of Science. In addition, the following conference and journal proceedings were also reviewed to ensure that all the relevant results were included: SIGCSE (Special Interest Group on Computer Science Education), CSEE&T (Conference on Software Engineering Education and Training), ICER (International Computing Education Research), ITICSE (Innovation and Technology in Computer Science Education), TOCE (Transaction on Computing Education), Computers and Education, STEM (Interdisciplinary STEM Teaching & Learning Conference, ASEE (American Society for Engineering Education), JTE (Journal of Teacher Education), JITE (Journal of Information Technology Education), IEJME (International Electronic Journal of Mathematics Education), and JSTOR (Journal for research in mathematics)

If the database was not able to deal with the entirety of the search string, the string was separated in order to fit and return an appropriate number of results. No more than 500 results were considered after examination revealed that those results after the first 500 were related. The study also restricted included papers to those from 2010 or later since the term gamification gained worldwide attention from 2010 [44, 20, and 51].

3.6. SLR: Inclusion and Exclusion Criteria

The search resulted in a large number of papers. The researcher filtered the publications retrieved from the search to exclude papers not aligned with the aim of the study. Inclusion and exclusion criteria are typically used in literature reviews in order to identify appropriate papers that could be included in the literature review. Table 1 contains the following inclusion and exclusion criteria used as a part of this review.

Table 1: Inclusion and Exclusion Criteria

Inclusion	Exclusion
GEs in STEM higher education learning	Education of early childhood, k-12 or education level that cannot be classified as higher education
Papers that report empirical data	Non-STEM related courses, for example, political science & government, accounting
Papers that talk about gamification	Papers that outline or framework or methodology not supported by empirical evidence
Papers that are in English	Papers that talk about designing games
	Papers not focused on student learning.

3.7. SLR: Study Execution

Executing the search strings on the source list resulted in 7,812 articles. First, the study filtered results based on titles and reduced the count to 562. Next, each of these 562 articles were reduced to 206 based on abstracts. Next, each of 206 articles were read in entirety and decision to include or exclude was made using the detailed criteria listed in Table 2 contains a list of the papers included in this literature review.

Figure 3: Study Execution Process

Table 2: List of Selected Publications

Id	Title	Publication Venue	Ref
P1	Do Points, Level and Leaderboard harm Intrinsic Motivations? And Empirical Analysis of Common Gamification Elements	First International Conference on Gameful Design	[36]
P2	Yooubi: One software for ubiquitous learning	Computers in Human Behavior	[37]
P3	Achieving Flow through Gamification: A Study on Re-designing Research Methods Courses	European Conference on Games Based Learning	[48]
P4	Improving Student Attitudes Towards the Capstone Laboratory Course Using Gamification	American Society for Engineering Education (ASEE)	[6]

Table 2: List of Selected Publications (continued)

Id	Title	Publication Venue	Ref
P5	Team Organization Method Using Salary Auction Game for sustainable Motivation	Sustainability Journal	[28]
P6	Strengthening an Educational Innovation Strategy: Processes to improve Gamification in Calculus course through performance assessment and Meta-evaluation	International Electronic Journal of Mathematics (IEJME)	[44]
P7	The Effect of Virtual Achievements on Student Engagement	Special Interest Group on Computer-Human Interaction (SIGCHI)	[10]
P8	Gamifying learning experiences: Practical implications and outcomes	Computer and Education	[15]
P9	High Score! – Motivation Strategies for User Participation in Virtual Human Development	International conference on Intelligent Virtual Agents	[19]
P10	How gamification Applies for Educational purpose specifically with College Algebra	Annual International Conference on Biologically Inspired Cognitive Architectures	[16]
P11	Engaging Engineering students with Gamification	International Conference on Games and Virtual Worlds for serious Applications	[3]
P12	The effects of Gamification on Engineering Lab Activities	Frontiers in Education Conference (FIE)	[27]
P13	Teaching case: Applying Gamification techniques and virtual reality for learning building engineering 3D arts	International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM)	[50]
P14	Tech - Gamification in University Engineering Education	International Conference on Computer Science & Education (ICCSE)	[31]
P15	Detecting and Clustering Students by their Gamification Behavior with badged: A Case study in Engineering Education	International Journal of Engineering Education (IJEE)	[46]

Table 2: List of Selected Publications (continued)

Id	Title	Publication Venue	Ref
P16	Challenge-based gamification and its impact in Teaching Mathematical Modeling	International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM)	[45]
P17	The Study of Gamification Application Architecture for Programming Language Course	International Conference on Ubiquitous Information Management and Communication (IMCOM)	[26]
P18	Applying gamification in the context of knowledge management	International Conference on Knowledge Technologies and Data-driven Business (i-KNOW)	[24]
P19	Game2Learn: Improving the motivation of CS1 students	Innovation and technology in computer science education (ITiSCE)	[4]
P20	Gamification in Educational Software Development	Computer Science Education Research Conference (CSERC)	[5]
P21	Improving Participation and Learning with Gamification	Gamification, First International Conference on Gameful Design	[2]
P22	A Playful Game Changer: Fostering Student Retention in Online Education with Social Gamification	Conference on Learning @ Scale	[30]
P23	On the Role of Gamification and Localization in an Open Online Learning Environment: Javala Experiences	Koli Calling Conference on Computing Education Research	[33]
P24	Does Gamification Work? — A Literature Review of Empirical Studies on Gamification	Hawaii International Conference on System Sciences (HICSS)	[20]
P25	TrAcademic: Experiences With Gamified Practical Sessions for a CS1 Course	Western Canadian conference on Computing Education (WCCCE)	[21]
P26	How (not) to Introduce Badges to Online Exercises	Technical symposium on Computer science education	[18]

Table 2: List of Selected Publications (continued)

Id	Title	Publication Venue	Ref
P27	Motivating Skill-Based Promotion with Badges	Special Interest Group on University & College Computing Services (SIGUCCS)	[52]
P28	Increasing Students' Awareness of Their Behavior in Online Learning Environments with Visualizations and Achievement Badges	Transactions on Learning Technologies (TLT)	[1]
P29	Gamification for Engaging Computer Science Students in Learning Activities: A Case Study	Transactions on Learning Technologies (TLT)	[23]
P30	A Gamified Mobile Application for Engaging New Students at University Orientation	Australian Computer-Human Interaction Conference (OZCHI)	[17]
P31	Teaching Software Engineering Through Game Design	Innovation and technology in computer science education (ITiSCE)	[7]

3.8. SLR: Data Extraction Form

After the final set of studies to be included were finalized, the researcher extracted the following data (Table 3) to avoid research bias by ensuring that the study was capturing the same data from all 31 papers.

Table 3: Data Extraction Form

Study ID	Unique identifier for the paper (same as the reference number)
Bibliographic data	Author, year, title, source
Study Type	Journal/conference
Aim of Study	The aims or goals of the primary study
Method	The type of research performed (e.g. case study, controlled experiment)
GEs	The gamification element(s) identified by the study
Focus area	Science or, Technology or Engineering or Mathematics courses
Results	Evidence regarding the usefulness of gamification elements for student learning
Concepts	The key concepts or major ideas in the primary studies
Higher-order interpretations	Limitations, guidelines or any additional information
Participants	Number of study participants

Gamification is defined as a technique that reuses game elements in a non-game context, which helps promote better user experience by improving students' motivation and engagement. This SLR focuses on the list of GEs (gamification elements) that have been used in different disciplines of STEM education. Based on our literature review, the researcher identified twenty-two commonly used GEs in the educational contexts. This dissertation have provided brief descriptions of each GE below (Figure 4). The researcher counted the number of times (frequency) each of the above GEs appeared in the primary studies. Figure 5 also shows the number of times each of the different GEs was identified in the primary study. Based on the results, leaderboard followed closely by badges and points, have been the most widely recognized (and reported) game

element in STEM education. Level was reported as the fourth commonly used GE. The remaining 18 GEs had much fewer papers reporting the evidence on their impact of student learning.

GE	Definition
1. Level (L)	A value that increases as students earn more experience points
2. Leaderboard (LB)	A board that displays score and the name of competing participants.
3. Badge (B)	Rewards allocated for completing a task
4. Points (P) /score/scoreboards	Points allocated for completing a task
5. Avatar (A)	Creating the character of a participant
6. Feedback (F)	Comments given to student participants from teachers.
7. Challenge (CLL)	Creation of Multiple-choice shared by users of the system.
8. Competition (CP)	Groups contesting against each other
9. Collaboration (CO)	Participants working in teams to complete an exercise.
10. Achievements (AC)	Given to students for completing a task that fulfil clearly stated requirements.
11. Narratives (NA) /storyline	The recounting of a story or account of events or experience
12. Deadline (D)	A define timeframe in which participant is expected to complete a task
13. Gaming Environment (G)	An imaginary place created by participants
14. Progress Bar (PB)	This shows the students about their progress in reaching a goal
15. Reputation Points (RP)	Allocated points that did not have a direct impact on student grades.
16. Quest (Q)	Tasks that are created to teach students how to complete a task effectively
17. Hints (H)	Using messages to reduce learning curve as well as to motivate participants
18. Knowledge map (K)	A guide that shows the progression of the class content
19. Virtual money (VM)	The allocation of coins to an individual in a team for completing a task
20. Problem-solving (PS)	Allocated task with a deadline
21. Visualization (V)	Students positions are represented in the form of dot and this dot gives them a predicts the end results based on current pace at which they progress
22. Punishment (PUN)	Awarded to students if they commit a mistake

Figure 4: Definition of GE Elements in the Primary Studies

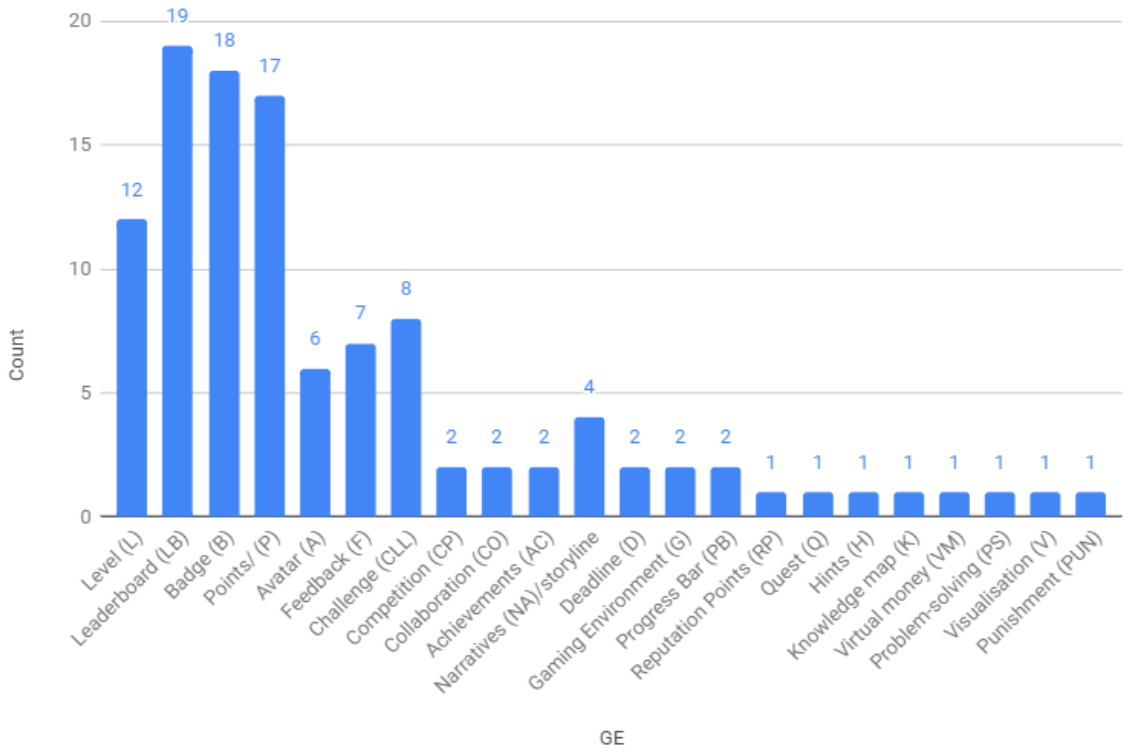


Figure 5: Occurrence of GE Elements in the Primary Studies

Whereas the empirical evidence for the benefit of using GEs which answers the third research question is given below: More details of the SLR results can be found in the Appendix.

Table 4: Empirical Evidence of Usefulness of GEs in STEM Education

Paper ID	Study Result	*Course / Subject*
P1	The results showed that implementation of points, leaderboard and levels game elements significantly increased performance, but did not affect perceived autonomy, competence or intrinsic motivation	Science & Technology- image annotation
P2	The analysis of questionnaires and reports from student participants showed that students were accepting of the u-learning environment.	Science & Technology, Biotechnology
P3	The study suggests that it is possible to increase students' involvement with the help of gamification. Although the preponderance of the students were happy with the format of the course some of them did not enjoy the game-like aspects at all	Research methods to Information and Communication technology (ICT) students
P4	Students became more interested and engaged in the course	Chemical Engineering Laboratory
P5	The result of the paper demonstrates that a gamified team organization method in engineering class could be used as an effective tool to enhance motivation and to improve learning outcomes of engineering students	Engineering Course
P6	The study shows that while it is true that gamification is a strategy that introduces a high level of innovation and brings the type of motivation and emotion that encourages learning its educational intent can be further strengthened by including performance assessment and meta-evaluation process to better understand its function and make adjustment to its design on a timely manner.	Mathematics
P7	The researchers discovered a highly significant positive effect on the quality of students' contribution, without a corresponding reduction in their quality, as well as on the amount of time over which students engaged with the tool. They also reveal that students enjoyed being able to earn badges and indicated strong preferences for having them available in the user interface.	Biology - Population Health

Table 4: Empirical Evidence of Usefulness of GEs in STEM Education (continued)

Paper ID	Study Result	*Course / Subject*
P8	The findings of the research suggest that some common beliefs about the benefits obtained when using games in education can be challenged. Students who completed the gamified experience received better scores in practical assignments and in overall score, but the researchers finding also suggest that these students performed poorly on written assignments and participated less on class activities, although their motivation was higher	Qualification for users of ICT
P9	Leaderboards, narratives and deadlines were observed to be effective in improving user participation. However, the users seem to approach the externally motivated human VH interactions in a less realistic way.	Medical Students
P10	The number of students who used the designed system (MathDungeon) and scored above the median score on the test of math performance was greater than the number of students who used most popular math tutoring programs used in US colleges (ALEKS – for higher education)) and scored higher than the median score.	College Algebra
P11	Results show significant increase ranging from lecture attendance to online participation, proactive behaviors and reading the course reference materials. Moreover, students considered gamified instances to be more motivating, interesting and easier to learn as compared to other courses.	Multimedia Content Production – MSc course in Information systems and computer Engineering
P12	The results suggest that gamification of engineering lab activities had a positive effect in terms of motivation, engagement and performance. This is indicated by a higher number of students who join the gamification (GM) website, the number of answers submitted by the GM group, the number of distinct days of participation and the score of exams for the GM group.	Engineering course – human factor
P13	The use of gamification in a classroom is expected to increase student engagement and motivation.	Computer Tools II course- Building Engineer and Architecture Degree

Table 4: Empirical Evidence of Usefulness of GEs in STEM Education (continued)

Paper ID	Study Result	*Course / Subject*
P14	Assistance of students increased on average from 60% to 86%. Punctuality of students increased in average from 10% to 79%. Participation of students increased on average from 15% to 47%	Process Management Challenge - Engineering Course
P15	There were some students who were motivated by gamification features while others were not. Students also have different badge indicators.	Physics, mathematics and chemistry course
P16	Based on the quantitative results it is concluded that there is no significant difference between partial and final test in the mathematical modeling routes phases.	Mathematics – Engineering and Integral Calculus course
P17	There was a positive impact on learning effectiveness. When the gamification elements were omitted from the application, there was decline in student achievements, decrease in user participation and user engagement.	Programming in Java
P18	The results evidenced an improvement in three areas: participation, collaboration and contribution). It was observed that each team member took an interest in each of the activities. The results showed 100% achievement in participation.	Software Engineering - Knowledge Management Process
P19	The student interviews and observations provided strong evidence that Game2Learn could be successful at enhancing student engagement and motivation. The results were positive only when students understood the game design concepts. Conversely, the performance was poor when students attempted tasks without reading the instructions.	Computer Science 1
P20	The results indicate an increase in the points earned by the group where the students can compare themselves with others.	Project Management
P21	There was an increase in the lecture downloads from 1.5 to 3 times. Compared to non-gamified years, the number of posts per student grew significantly 4 to 6 times in the first gamified year and 6 to 10 times in the second gamified year. They also observed higher minimum grades during gamified years.	Multimedia Content Production – MSc course in Information systems and computer Engineering

Table 4: Empirical Evidence of Usefulness of GEs in STEM Education (continued)

Paper ID	Study Result	*Course / Subject*
P22	Using game and social conditions resulted in higher average retention periods. Students in the game and social conditions group had higher test scores than students in the control group.	Computer Science - python
P23	When the gamification is used, users spend more time and complete more exercises. The total time that the student spent was significantly smaller when gamification was turned off.	Computer Science Programming - Java
P24	The major finding of the paper is that most of the gamification elements acted as motivational affordances, but reward points, leaderboards and badges were the most influential game design elements.	Multiple discipline
P25	There was a 500% increase in the attendance. TA's strongly agreed that the practical sessions were helpful to students.	Introductory Computer Science Course
P26	The results showed that one third of the students agreed that badges were motivating, while another third indicated they were demotivating, and another third said they had no effect.	Data Structures and Algorithms
P27	Authors saw an improvement in student's performance in learning technologies as badges acted as their intrinsic motivator.	HTML
P28	The badges helped students improve their course performance due to an urge to grab more badges.	Data Structures and Algorithms
P29	There is a statistically significant difference in mastering topics before and after the gamification was introduced. The results also showed students worked beyond the requirement to master the unexplored topics of the course.	C-Programming Language
P30	The survey results suggested that game elements were positive a addition to the application and the students reported that they motivated them to learn about the campus. Authors found that leaderboard was a major motivating factor.	Orientation

Table 4: Empirical Evidence of Usefulness of GEs in STEM Education (continued)

Paper ID	Study Result	*Course / Subject*
P31	Authors observed that the enrollment rate was up, dropouts were down, and grades were noticeably improved. Subjective comments from the students suggested a greater interest in software engineering course.	Software Engineering

In this dissertation, a systematic literature review study has been conducted to analyze the use of gamification in STEM courses. The scope of this SLR was gamification in STEM wide discipline, thus only education or serious games were considered. The study identify the most commonly used GE in different STEM courses. The researcher also evaluate the primary studies, seeking to analyze how the application of game mechanics in an educational context affect aspects such as user motivation, user engagement, user participation, etc. were impacted. The results show that integrating gamification elements into STEM education helps in achieving positive educational outcomes. Particularly, the addition of gamification elements into the online learning material has been shown to act as a motivation factor for students to become more active in learning.

The results obtained during the analysis of the primary studies show that the existing literature on gamification is more utilized in computer science level one courses (CS1). Most of the studies examined look at programming within the CS1 curriculum. This show up an important gap that must be addressed since other STEM courses (example chemistry and physics) and other areas of computer science (such as software testing and software engineering) have not been studied to their full extent.

Another shortcoming identified was that most of the studies make use of the most common GE, which are; leaderboard, badge and points. This reflects that researchers do not focus on

intrinsic motivation such as psychological and behavioral variables. The use of GE which is too simple can lead to some students not finding gamified learning activity engaging hence enthusiasm for utilizing the platform diminished after two or three days.

One other gap that we were able to identify is the nonexistence of empirical study that shows the impact GE has on gender. The researcher would like to see study carried out on this topic to see the difference between female and male performance as well as to highlight different ways in which gender are motivated. In our opinion, the researcher would like to also see ways in which gamification can be used to pursue STEM studies and retain women in the STEM field, particularly computer science.

Results are intended to be applied in the area of gamification in computer science / software engineering and will be used to improve the design and usability of an online learning environment by extending the range of STEM courses supported in the online environment. While there are some studies underway, the researcher plan to report the results and conduct additional studies guided to build a larger body of evidence on the usefulness of gamification in CS/SE education.

The researcher invite researchers to conduct more studies that focus on other STEM courses outside of computer science. The researcher foresee that within the next decade, there will be more empirical evidence in STEM to further support its application.

4. VALIDATION OF CODE REVIEW IMPLEMENTATION IN CS1/CS2/SE COURSES ACROSS MULTIPLE INSTITUTION

This chapter describes the framework of the research. This chapter describes a series of three controlled studies (one building on another) that evaluated different aspects of code review (at individual and in collaboration with peers) in different courses (CS1, CS2, Software engineering courses) across different higher education institutions (NDSU and Miami University). The studies focused on developing a checklist that students can use when performing code reviews with the underlying goal that the code review sessions can help students become better programmers and lifelong learners. Section 4.1 explains the study that evaluates the impact of Learning Objects availability within SEP-CyLE on CS1 student learning at North Dakota State University. A study was conducted with CS1 students to gain insights into the relationship between PCR and code development to help CS1 students understand programming concepts and improve their programming skills, which is describe in Section 4.2 Section 4.3 explains the study that was conducted to evaluate the usefulness of guided (checklist) PCR on CS2 students' learning of programming concepts. Section 4.4 describes the study that evaluated the usefulness of guided (checklist) PCR on CSE students' learning with respect to programming concepts and coding standards.

4.1. Study 1: Evaluating the Impact of the Frequency of LOs in SEP-CyLE

SEP-CyLE enables instructors to provide students with access to LOs depending on the way SEP-CyLE is integrated with in-class course meetings. In the past, at NDSU, instructors have used SEP-CyLE where LOs are assigned at the beginning of the course or randomly and students can access them at their perusal or interest. Based on the feedback collected during past studies, it was reported that, if LOs can be more closely integrated with the content being covered during the

class lectures and lab assignments, students can use LOs to self-assess and identify areas they need improvement. This study was intended to analyze the impact of *frequency of LO assignment* (e.g., assigning LO's every week vs assigning them randomly whenever the instructor believes that extra resource support would help the students to better understand the concept) on student learning and understanding of programming concepts in an introductory programming course. The study was conducted across two different sections of an introductory programming course (CS1) taught by two different instructors during the same semester. Each section represented a different experimental condition. The study utilized a pre- and post-test instrument to measure the impact associated with using SEP-CyLE's resource support on student learning outcomes.

4.1.1. Study 1: Study Goal

This study had two primary objectives. The essential goal was to analyze the impact of Learning Objects availability within SEP-CyLE on student learning, which formally is phrased as follows:

Investigate the effect of availability of digital learning objects included in SEP-CyLE on students' acquisition of software programming and testing conceptual knowledge, tools and techniques in an introductory computer programming course

The second goal of the study is to evaluate the overall satisfaction of students on different features of SEP-CyLE as well as its usability in the programming course and to enhance its future usage.

4.1.2. Study 1: Participating Subjects

The study was conducted across two different sections of an introductory programming course at North Dakota State University, taught by two different instructors. A total of 26 students participated in the study.

4.1.3. Study 3: Study Procedure

The study was conducted in one CS2 class over a period of four weeks during the summer of 2019. PCR sessions were conducted once per week and lasted approximately thirty-five minutes. During each PCR session, students were given a different piece of code to review. Each piece of code contained 20-70 source lines of code (SLOC) and was written in Java. Each code fragment was seeded with multiple errors representative of most commonly committed errors by students at North Dakota State University. No line of code exhibited multiple errors. The code examples were developed in consultation with two CS2 instructors at NDSU and included topics that students in CS2 had previously discussed in lecture including doubly linked lists, array sorting, trees traversal, Breadth First Search in graphs). The study procedure included the following major steps:

Step 1: Introduction to PCR: Students were introduced to guided PCR and the purpose of the study.

Step 2: Individual Peer Code Review: Students were given individual error checklists, example code, and error sheets for them to record errors in the code. Students were first asked to review a piece of source code individually using the checklist. A sample of questions used to guide code review is shown in Figure 6.

1. Are all loops correctly formed, with the appropriate initialization, increment and termination expressions?
2. Do Boolean functions return the correct value?
3. Has correct syntax used for logical operators?
4. Are values assign to the correct variable?
5. Are there any typographical error such as = = instead of = or != instead of !
6. Are variables mathematical calculation done correctly such as adding instead of subtracting?

Figure 6: Sample Java Checklist

The researchers in consultation with the course instructor developed four pieces of code for error abstraction, with each piece of code becoming progressively more difficult, in terms of both programming concepts covered and the number of SLOC. In each piece of code, five categories of errors were seeded with on average of 20 errors in total; at least one error belonging to each category was seeded in each piece of code. The seeding of errors was done with input from instructors who have identified major errors committed by students in CS2 programming course at NDSU. Table 5 shows the category of errors. The output of this step was 16 "individual error sheets" (one per student) that described the errors (including Line # in source code and error description) found during each code review session.

Table 5: Category of Errors Seeded in Code

Category of Errors	**Definition of Errors**
Initialization Declaration Error	Not creating or declaring an identifier used in program
Method call / Definition Error	Not naming or invoking a method correctly
Arrays, linkedList and trees Error	Not indexing an array correctly, not creating a node correctly or defining pointers correctly, assigning a value to the wrong variable
Output Error	Grammatical errors in displayed output or improper statement termination or incorrect output formatting
Flow of control Error	Improper looping, incorrect value return for Boolean function or incorrect use of syntax for logical operators, adding numbers instead of subtracting

Step 3: Group Peer Code Review: Students were randomly assigned to pairs (groups of two) to find and record any additional errors on a new error sheet (group error sheet) using the checklist. Students were asked to first discuss their individual results (found during Step 2) and only record new errors (not found by either of the students belonging to the pair). The goal of this step was to identify the importance of team based review activity. The output of this step was eight "group error sheets" (one per group) that described the new errors.

Step 4: Code Reflection sheet: After students finished reviewing each piece of code, they were asked to fill out a reflection sheet for that code fragment that described actual seeded errors. Each reflection sheet (a total of four for each PCR session) described actual seeded errors in the code, the location of each error, asked students to comment if they did (or did not) locate the error, and to provide comment if they did not agree it was an error.

Step 5: Discussion: After each reflection period, and prior to subsequent guided PCR session, researchers discussed the seeded errors in the code artifact and used the reflection to encourage discussion of common programming errors.

Step 6: Survey: At the end of the four-week study, a survey was administered to gather students' feedback on the usefulness of checklist in PCR exercise and its impact on their improved understanding of programming concepts. Students responded to a set of questions using a 5-point Likert scale.

4.1.4. Study 1: Study Procedure

The study procedure included five major steps:

Step 1-- Pretest: A pretest was conducted at the start of the semester prior to introducing SEP-CyLE in order to determine the student's baseline knowledge associated with programming concepts.

Step 2 -- Introduction of SEP-CyLE: The students were provided with information about using SEP-CyLE. They were trained on how to browse learning objects and video tutorials, how to track their efficiency, how to change their profile details and socialize with their peers within the cyber learning environment, and how to post in the discussion forums.

Step 3 Assigning LOs to the students: For the experimental group, students were assigned an LO each week based on course content being discussed that week. For the control group, all the required LOs were assigned to students at the beginning of the semester. The students were expected to complete each LO and were graded based on the results of LO quizzes. All of the LOs used, were developed by the SEP-CyLE development team or by researchers specifically for use in the course.

Step 4 -- Posttest: Towards the end of the semester, the students were reevaluated using a posttest instrument which had a set of questions similar to those in the pretest instrument. This was done to understand the impact of additional resource support on improvement in the conceptual knowledge of the students.

Step 5 -- Survey: Towards the end of the study, students participated in a focus group analysis to gather the feedback for enhancing SEP-CyLE for future courses.

4.1.5. Study 1: Data Collection and Evaluation Criteria

The main data source is the student's responses to the pre- and post-tests described previously. The students also completed a survey where they responded to questions using a 5-point Likert scale. The study also collected SEP-CyLE data for all students in terms of the number of LOs attempted, number of LOs passed, and time spent on SEP-CyLE, and the number of virtual points the student earned. The researcher also analyzed students' performance in terms of their assignment scores, exam scores and end of semester grades.

Following that the researcher evaluated the relationship between the different variables to evaluate the impact of SEP-CyLE usage metrics on the student's performance in different sections. The independent variables and the dependent variables are listed below.

4.1.6. Independent and Dependent Variables

Independent Variables:

- Number of LO's attempted: This is measured as the Number of LOs that students attempted from the assigned ones in Sep-CyLE.
- Number of LO's passed: This is measured as the number of LOs completed by students with at least 80% questions correctly answered.
- Total time spent on SEP-CyLE: Sum of the total amount of time spent on all LOs.
- Virtual Points earned: For each LO or activity completed, students earn virtual points.

Dependent Variables:

- Average assignment scores: Average of all the assignment scores of students.
- Average exam scores: Average of all the exam scores of students.
- End-of-semester course grades: These are the grades that are assigned to students at the end of the semester.

4.1.7. Study 1: Summary of Results and Analysis

This section describes the results that have been found in this study.

To evaluate the impact of SEP-CyLE on the student's understanding of the programming concepts and their proficiency associated with tools and techniques, a comparison between pre- and post-test was performed. The result from paired t-test showed that the increase was statistically significant (at $p < 0.01$) or both groups. From Figure 7, there was an increase in post-test scores for both of the sections. The result from paired t-test showed that the increase was statistically

significant (at $p < 0.01$) for both groups. The highest increase was for the experimental section that had enabled allocating LOs on a weekly basis.

Additionally, the researcher divided the pre/posttest question into following categories:

- Memory Management: 1, 2,11 and 15
- Basic Programming Concepts: 3,4,7,8,9 and 21
- Arrays: 4, 5, 10, 11
- Basic Operators: 6 and 20
- Methods: 7
- Pointers: 10
- OOPs Concepts: 12, 13, 14 and 15
- Software Testing Concepts: 16, 17, 18, 19 and 20

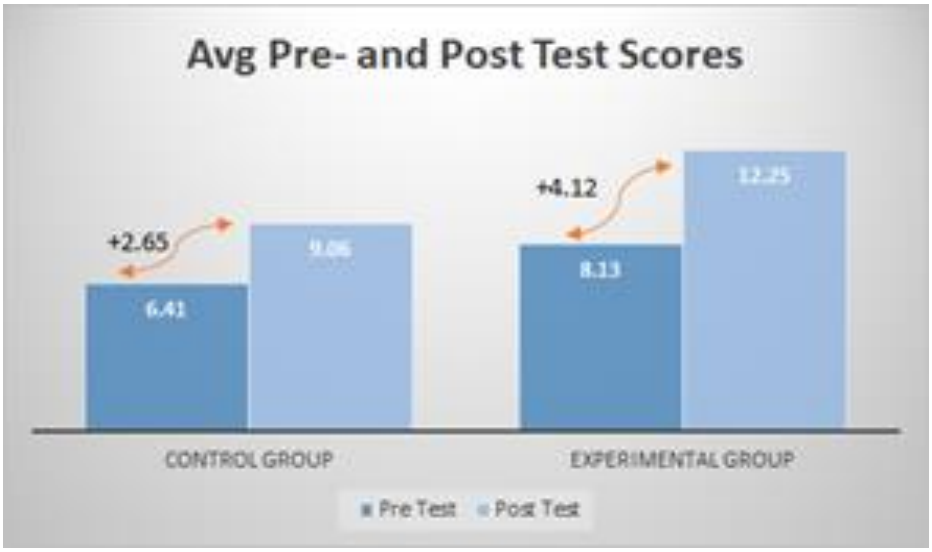


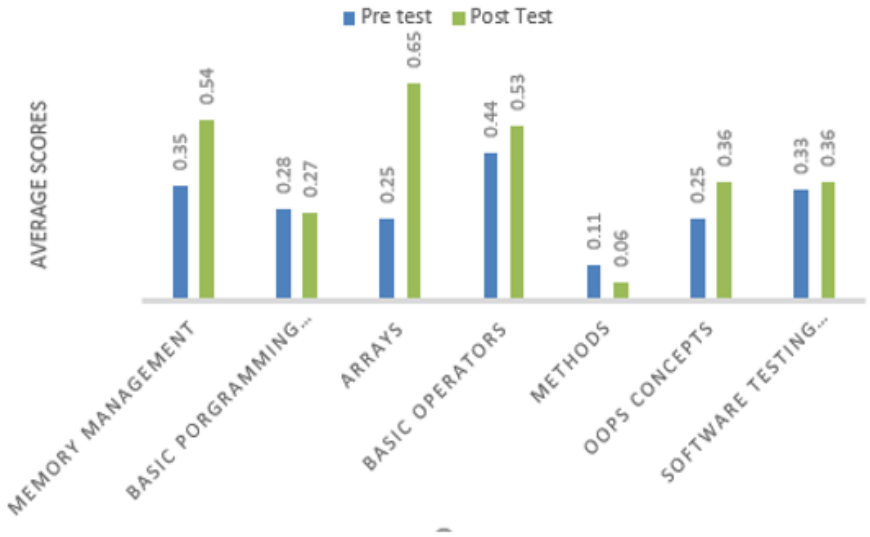
Figure 7: Pre and Post Test Scores of Control and Experimental Group

The study compared the average pre/posttest scores across the both sections to assess whether the improvement in post-tests was measurably noteworthy for all programming concepts. Based on the results, there has been improvement in the post test scores of both sections, which is shown in Figure 6. Surprisingly, the experimental group students did not answer questions related to methods concepts. This may be because they are still struggling to understand the concepts of methods. Even when we examine the control group, there is no improvement in their posttest scores in methods area, which can be also looked as a knowledge deficiency of students in that particular area.

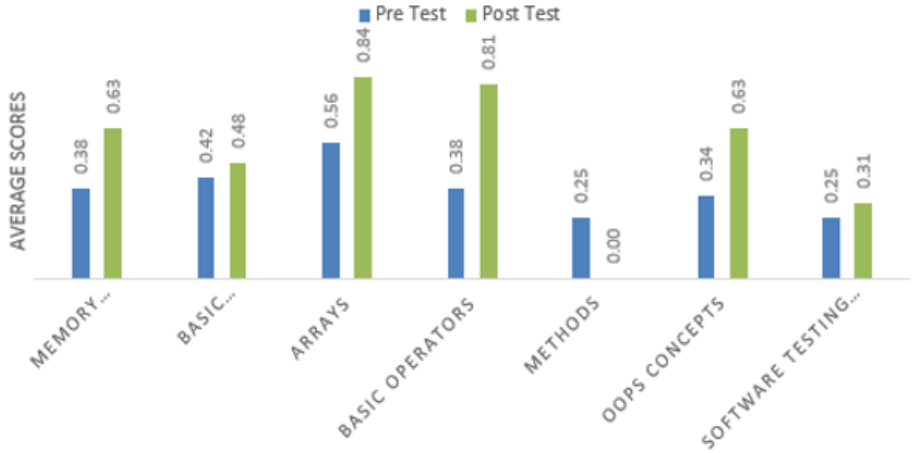
Table 6: Relationship between SEP-CyLE Usage Metrics vs. Student Performance

	Control Group			**Experiment Group**		
Vs.	Average assignment scores	Average exam scores	End-of-semester course grades	Average assignment scores	Average exam scores	End-of-semester course grades
# of LOs attempted	r= 0.412	r = 0.231	r= 0.392	r = 0.685	r = 0.283	r = 0.645
	p =0.101	p =0.371	p =0.119	p =0.06	p =0.497	p =0.08
#Los passed	r = 0.412	r = 0.231	r.= 0.392	r = 0.829	r = 0.516	r = 0.872
	p =0.101	p =0.371	p =0.119	p =0.01	p =0.190	p =0.005
Time Spent	r = 0.275	r = -0.192	r = 0.157	r = 0.667	r = 0.477	r = 0.717
	p =0.285	p =0.459	p = 0.547	p =0.07	p=0.233	p =0.05

Table 6 provides the Pearson correlation coefficient ‘r’ (strength of correlation) and p-value ‘p’ (statistical significance) between these independent and dependent variables. The experiment group has stronger positive correlation (significant at p-value <0.1) Also, providing extra resource support can positively contribute to the overall success of students in an introductory computer programming course. Based on these results, the SEP-CyLE LOs had a positive impact on students’ performance.



Control Group



Experiment Group

Figure 8: Mean Pre/Post Test Scores for Different Categories of Programming Concepts

4.1.8. Survey Results

In terms of student feedback, there was positive feedback of SEP-CyLE with regards to how the students felt about the Learning objects and overall satisfaction with the website, its ease and clarity of information as shown in Figure 9 & 10. Students rated their responses on a 5-point Likert scale [1= Strongly Disagree; 2= Disagree; 3= Neither Agree nor Disagree; 4= Agree; 5=

Strongly Agree]. Starting with the satisfaction of the website and its ease of use, SEP-CyLE had positively impacted students positively. Most of the students strongly agree that SEP-CyLE is pleasant. With regards to Learning Objects, survey results from Figure 10 indicated that students are satisfied with the learning objectives content and most of the students were positive about how the learning objects helped them in understanding the programming concepts.

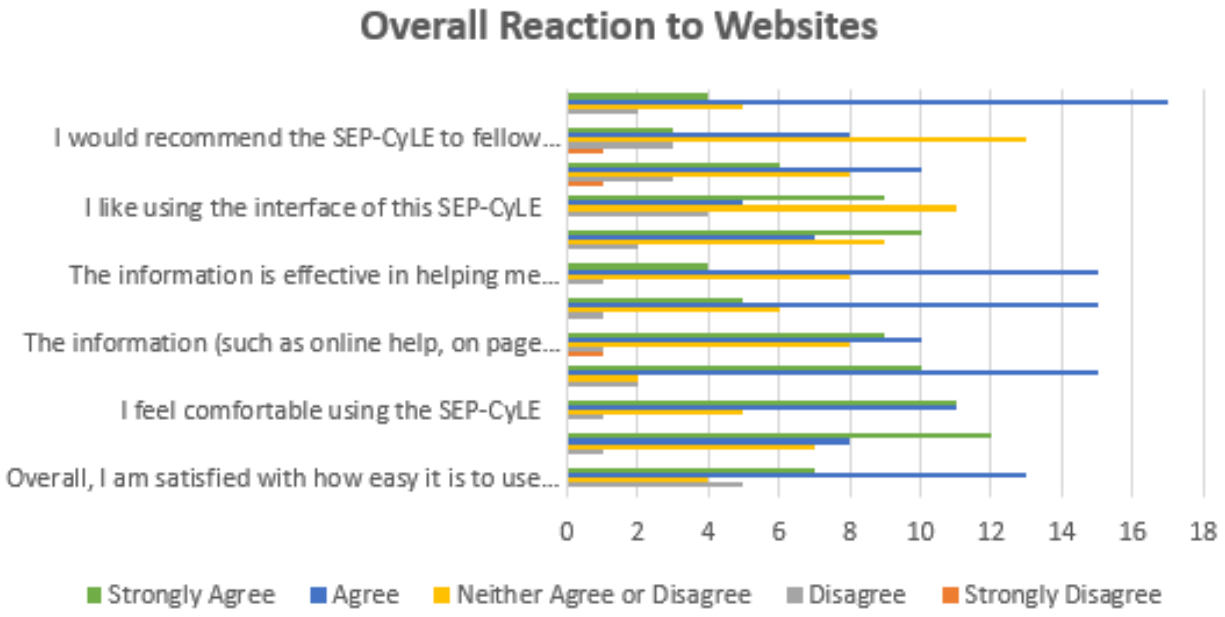


Figure 9: Survey Results of Overall Reactions on the Website

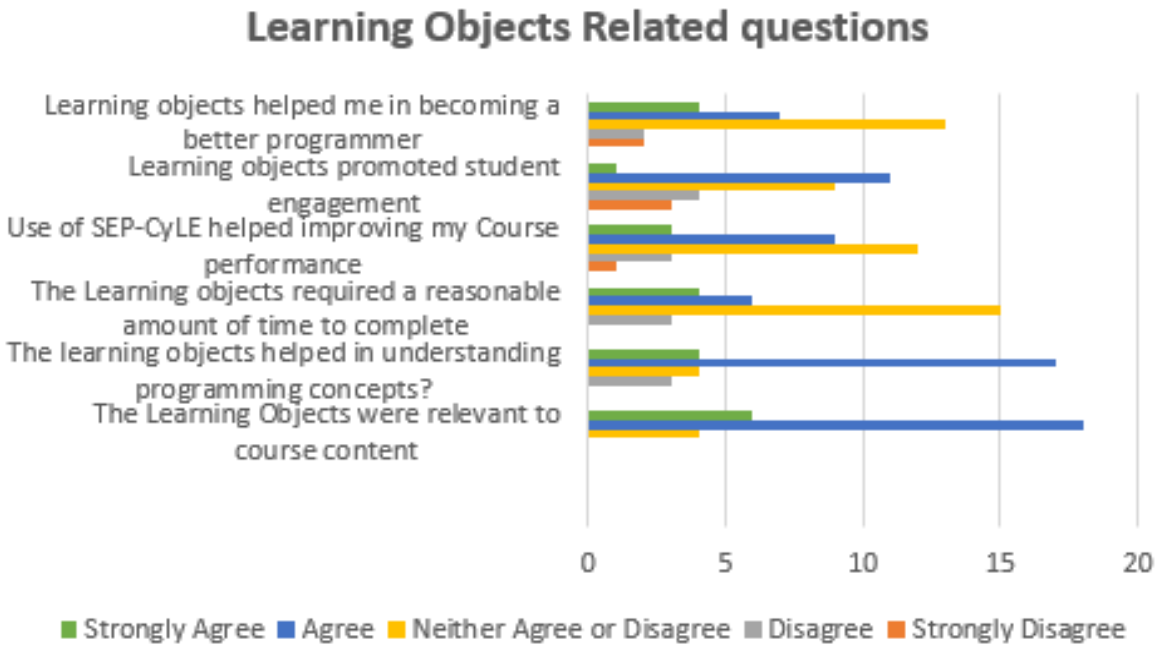


Figure 10: Survey Results on Learning Objects Related Questions

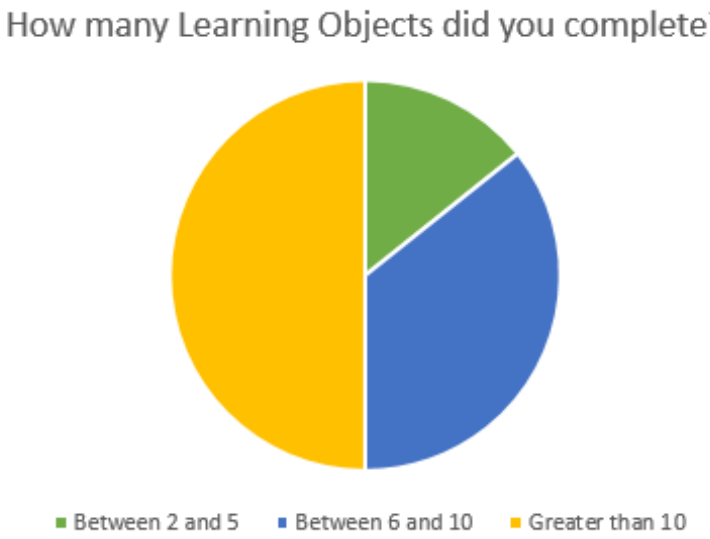


Figure 11: Number of LOs Completed

When students were asked about how many learning objects the students completed during their term of course half of the students indicated that they had completed more than 11 learning objects, which suggests that the students are more motivated to use LOs (Figure 11)

Whereas Figure 12 provides us the details of the number of LOs that are completed by students as part of their coursework. All the survey results indicated that the students are satisfied and positive with the usage of SEP-CyLE. Also, to get a better understanding of their opinion about different features of SEP-CyLE, the researcher asked them some set of sub questions like how virtual points of SEP-CyLE affected their individual and team performance, did use of SEP-CyLE improved their course performance and so on. The researcher categorized these questions and are presented below.

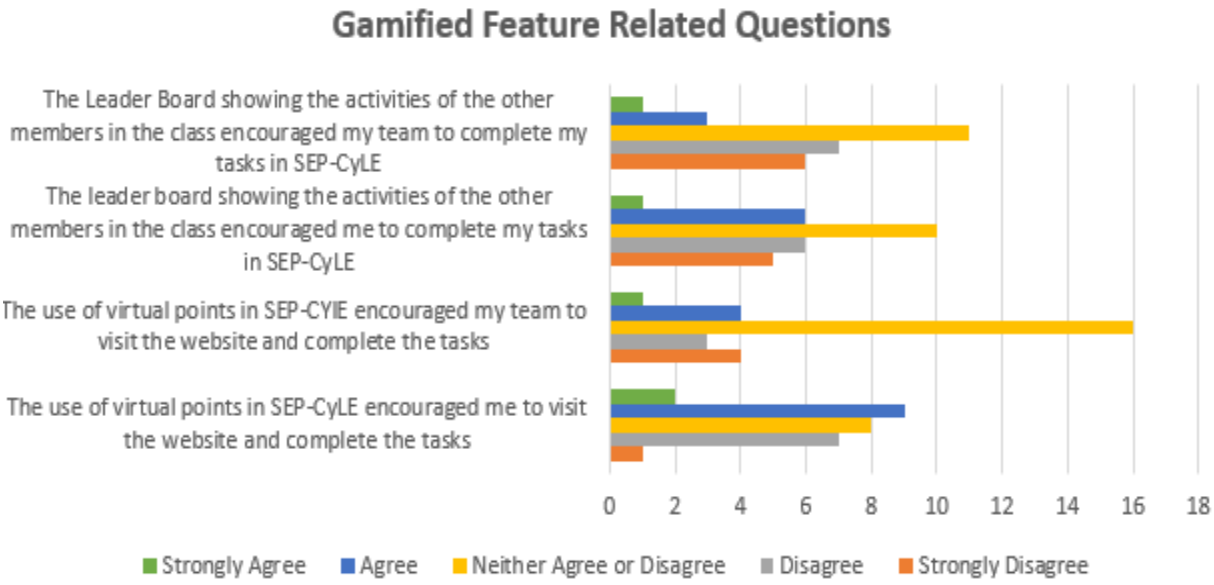


Figure 12: Survey Results of Gamification Features Related Questions

The study also asked students to describe how motivated they are to use SEP-CyLE and finish the assigned tasks and whether gamification had an impact on the students' behavior. Figure 12 depicts the survey responses regarding gamification features of the website. And there was a highest positive response that the virtual points have helped the students to perform the task and visit the website frequently.

Discussion of Results: The outcomes from this study revealed the promise of making use of SEP-CyLE with continuous resource support and learning engagement strategies included to provide better results in teaching software programming and testing concepts for introductory computer programming courses. Based on the correlation results, providing students extra resources based on what is taught during that period of time would enhance their understanding of the concepts clearly. From this study, it is evident that there is significant improvement in students' performance when they have been assigned the respective learning material. Assigning one LO on a weekly basis that covers major topics, for e.g., Arrays and white box testing, is needed for supporting continuous student learning. As the results from this section have also shown that students are struggling with Methods concepts, this could be because they were not able to apply the concepts that they learnt or may not be able to recall the topics that were taught during the semester.

From this study, it is observe that some students were motivated by leaderboard and virtual points as it provides instant feedback and allows students to continuously strive to improve their place in the rankings. However, taking a look at Figure 11 closely, other students are not motivated by these elements. This could be because they might have not enjoyed the element of competition that leaderboards and virtual points have introduced into the learning environment. Also, students are still struggling with methods concepts even after completion of their score, which is a significant observation that the researcher found in this study.

It was noticed that a digital LO should be allocated after a topic is introduced in the class but before in-class quizzes or exams to improve student performance. Survey results showed that students would like to work on LOs in groups using SEP-CyLE. LOs allowed students to evaluate and improve their understanding before attempting major assignments.

Threats to Validity*:* This section attempts to address a few of the threats to validity that could possibly have influenced the results of the research. A major threat is the small size of subjects taking part in this study, which limited the data analysis. Second, the students might not have taken the posttest seriously because it was voluntary (i.e., it did not have an impact on their grades) and they might not have been motivated/inspired to perform well on it. Another validity threat would be that both the sections were taught by different instructors and their teaching strategy might be different. However, all the students were taught the same material.

In this dissertation, the study described our approach of providing continuous resource support with SEP-CyLE in conjunction with the use of gamification elements. The study talked about the current design and the outcomes of the experimental study in order to determine the impact of SEP-CyLE with continuous resource support on the students' learning. Our results have indicated that while the use of SEP-CyLE with gamification elements can positively impact students' performance and engagement in an introductory programming course. In future studies the researcher intend to use SEP-CyLE in upper level CS courses and assess how continuous resource support along with the learning engagement strategies affect students outside of introductory programming courses.

4.2. Study 2: Peer Code Review CS160

As mentioned earlier, SEP-CyLE is currently limited in ways students can collaborate on programming tasks that would help improve their conceptual and practical understanding of programming. Before these collaborative features can be implemented in SEP-CyLE with other LESs, it is important to understand the ways collaboration can be effectively used in the context of CS/SE classrooms. On that end, at NDSU, pair programming has found to be an effective pedagogy in CS1/CS2 classrooms. This study is extending that idea by implementing peer code

review (where students review an externally developed code) and learn best practices at developing their own code and at comprehending someone else's code (both skills needed after graduation). Therefore, the researcher conducted a controlled study to gain insights into the relationship between peer code review (PCR) and code development to help CS1 students understand programming concepts and improve their programming skills. This goal aimed at evaluating whether PCR can assist students' at improving their programming skills in CS1 courses. The secondary goal is to understand how PCR can be used as a pedagogical tool to help students improve their conceptual understanding of programming. To accomplish these goals, a pretest-posttest experiment was conducted in CS1 class wherein students participated in PCR of externally developed code samples seeded with realistic faults. After each PCR session, students were asked to reflect on their review results, discuss errors, and ways to avoid them when developing their own programs. Post PCR sessions, students developed their own programs. At the end of study, students provided feedback on the usefulness of PCR in an introductory programming course.

4.2.1. Study 2: Study Goal

The major goal of this study is to gain insights into the relationship between PCR and code development to help CS1 students understand programming concepts and improve their programming skills.

4.2.2. Study 2: Research Question

The study formulated the following three research questions to accomplish our research goal:

- **Research Question 1 (RQ1):** *What major insights regarding student learning of programming concepts can be gained from using peer code review in CS1 classroom?*
- **Research Question 2 (RQ2):** *Does peer code review leads to an improvement in programming abilities of students?*
- **Research Question 3 (RQ3):** *How can PCR be integrated into CS1 curriculum to improve student learning and engagement?*

4.2.3. Study 2: Participating Subjects

This study was conducted in an introductory programming course at North Dakota State University. A total of nineteen students elected to participate in the study. The student participants were made up of CS and non-computer science majors, most of which are in their freshman year and had almost negligible prior programming experience.

4.2.4. Study 2: Study Procedure

The study was conducted over a period of three days and lasted fifty minutes each day. The study procedure included following major steps:

Day 1: Students were provided with three different code-snippets (each piece of code contained 30-60 SLOC and was written in Java). These code-snippets were executable Java code and were seeded with errors representative of most commonly committed errors by introductory programming learners. To establish a baseline of students' programming knowledge, students were asked to predict the output of these faulty code-snippets. These code snippets included topics that students in CS1 were recently taught and that students struggle at comprehending (including loops, arrays, exception handling). The code snippets were developed in consultation with two CS1

instructors at our university. The evaluation of students' ability to predict the expected output is referred to as 'pre-test' in the remainder of this section.

Day 2: PCR - Next, students were asked to review 'code snippets" (a piece of source code with 30-60 SLOC). Three pieces of codes were created by researchers in consultation with the course instructor. These code samples were developed based upon the topics that were recently taught in class. Each snippet was progressively more challenging, both in terms of programming constructs and the length of the programming activity. In each code snippet, six categories of errors were seeded (18 errors in totality); one error belonging to each category was seeded. Errors were seeded in consultation with two CS1 instructors that had identified major errors categories identified in CS1 programming course deliverables. These error categories are listed in Table 7.

Table 7: Categories of Errors Seeded in Code Snippets

Category of Errors	Definition of Errors
Declaration Error	Not declaring an identifier used in program
Exception Error	Event that will disrupt the normal flow if instructions
Loop counter Error	a mistake in a program's source code that results in incorrect or unexpected behavior
Instantiation Error	Occurs when an application tries to use the Java new construct to instantiate an abstract class or an interface.
Library Error	library that cannot be located by the Java Runtime Environment
Reference Error	Trying to use a variable before it is defined.

First, students were instructed to review each code-snippet. After students finished reviewing the first code sample, they were asked to fill the reflection sheet for that sample (that described actual seeded errors) before attempting subsequent code-snippets. Each reflection sheet

(a total of three for each code sample) described errors in that code, the location of the error and asked students to report if they were able to identify each error (Yes/No)? If they reported it? and the reason if they noticed but did not report. After each reflection and prior to the subsequent PCR session, researchers and CS 1 instructor discussed the faults that were seeded into the code artifact and used the reflection to foster discussion of common programming errors.

Day 3: **Posttest**- During the final day, students were asked to write three new programs in java (that were similar in terms of code artifacts reviewed during the prior step). These programs were subsequently evaluated by researchers to evaluate the impact peer-code review had, on the quality of code produced by the students. At the end of the study, a survey was administered to gather students' feedback on the usefulness of PCR exercise and its impact on their improved understanding of programming concepts. Students responded to a set of questions using a 5-point Likert scale. From Figure 13, it is evident that student's knowledge has been increased from code snippet 1 exercise to code snippet 3 PCR exercise. Prior to the introduction of code snippet 1no intervention in the form of discussion was provided to students. After code snippet 1, the researcher had a five minutes discussion with students about the reflection sheet. This intervention helped students to identify the types of errors and they were able to identify more errors in code snippets 2 and 3.

4.2.5. Study 2: Data Collection

The researcher collected quantitative data during the PCR sessions and when evaluating the quality of the code produced by students post PCR. Specifically, collected data regarding the categories (Table 6) and number of errors reported during each PCR exercise, the number of errors left undiscovered during each PCR exercise, the type and number of errors made when students developed their own code.

Additionally, the study collected data regarding the students' feedback on code review and code development activity. These questions provide insights regarding the students' overall satisfaction with the PCR. The feedback from students allowed researchers to evaluate the usefulness of the peer code review and the code development activity in an introductory programming course.

4.2.6. Study 2: Data Analysis and Results

In this section, initially the PCR, pre- and posttest results were analyzed and later the correlation analysis and survey results are described.

Study 2 (RQ1): What major insights regarding student learning of programming concepts can be gained from using peer code review in CS1 classroom?

This RQ aims at gaining insights on using PCR technique while teaching CS1. To answer this question, the researcher analyzed students' output on Day 2 (PCR sessions) in terms of the number of actual errors found by students when reviewing each code snippet. This was done to identify errors that students were able to identify, errors that students were not able to identify, and the impact of discussion/reflection had on their peer-review performance.

To provide an overview of the results, Figure 13 compares the percentage of errors reported by students belonging to each of six categories of errors seeded in each code snippet. The results are presented in terms of percentage of errors reported by students for each error category and for each code snippet.

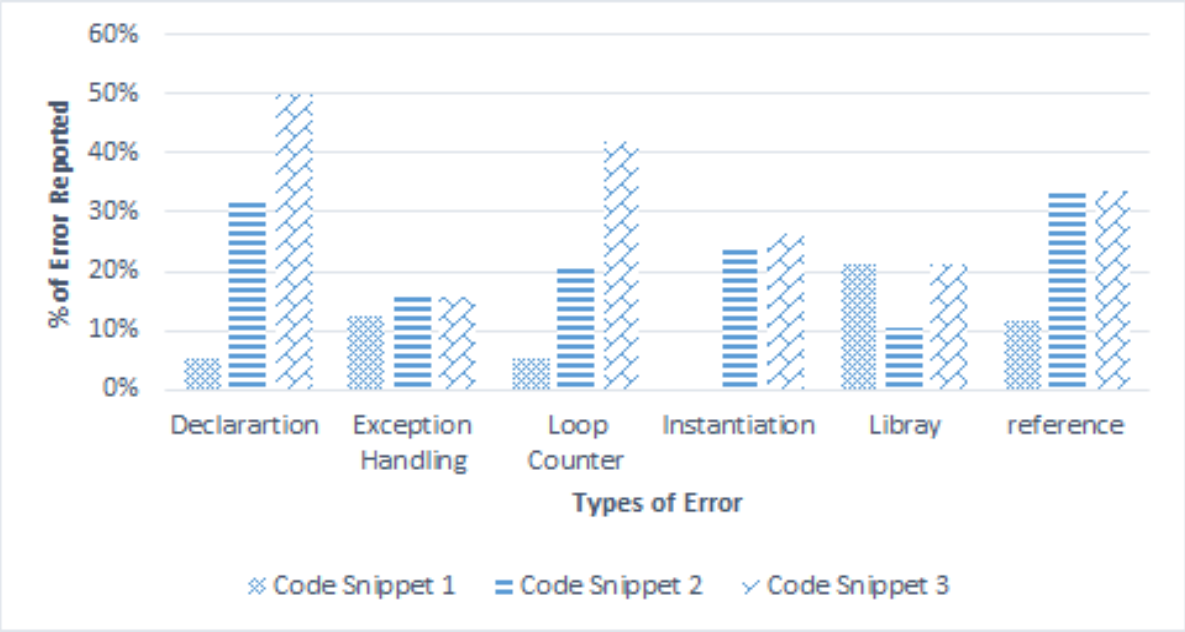


Figure 13: Percentage of Error Reported by Category

The major observations are discussed below:

Most and Least reported errors: When comparing the average values across all three code snippets, Declaration (an average of 89.47%), loop (an average of 68.42%), and reference (an average of 78.95%) were most commonly reported errors. Similarly, Exception Handling (an average of 43.86%), library (an average of 42.11%) and instantiation (an average of 50%) were least reported errors. Based on these values, students did not seem to have a good understanding on the correct usage of libraries and instantiations in coding. Researchers and CS1 instructors speculate that students are not able to report exception handling errors because of their dependence on IDE (e.g., Eclipse). Students tend to rely on trial and error (trying to figure out what each line in the code is actually doing) when trying to report the error.

Improvement in student output across code snippets: The researcher also noticed an improvement in the review performance of students. Students found an average of 5.5 errors when reviewing the first code-snippet 1, 13.5 errors when reviewing the second code-snippet, and 16.5

errors during the last review session. This is consistent across each error category (expect 'library' error types) as well. Students' ability to report library errors does not follow the same trend (e.g., students reported fewer library errors when reviewing the second code snippet). Based on the reflection reports, students did not pay attention to incorrectly spelled imported libraries which was discussed post inspection which shows improvement during the third code review. Researchers noted that using reflection to discuss the actual errors seeded between each peer review session helped students improve their understanding of errors and is reflected in their improved ability to report those error types in subsequent review sessions.

Analysis of reflection sheets: The researcher also analyzed students' reflections (post PCR session) to gain insights into the reason they were not able to report particular types of errors. A summary of the students' thought process is listed in Table 8. One of the things that emerged from students' responses is "overlooked / assumed wasn't an error" which would explain a significant improvement (145% increase in performance from first to second code review). One of the major insights gained from the reflection reports is that students tend to "power browse" (and not read each line of the code under review). As a result, they did not report errors in the first code snippet due to lack of attention to detail. However, as students progressed from one code snippet to the next, the researcher noticed that their readability and their attention to detail improved as they were able to find and report errors that were previously undetected.

Table 8: Reasons Students did not Report Errors

Category of Errors	Comment (Why did you not locate the error)
Declaration Error	Did not pay attention to detail, overlooked
Exception Error	Not as familiar with using files, did not remember, IDE would catch error, did not recognized, did not know it was an error, forget, did not the variable was a string, did not look at it close enough, only use file once and forget the correct syntax, not a lot of practice with files, not enough experience with java, was not looking for the error, overlooked.
Loop counter Error	Missed it, did not have iteration in mind, have never seen such a case, did not know it was an error, did not recognize it, and did not understand the error.
Instantiation Error	Assumed it was not required, Missed, overlooked.
Library Error	Use to automatically import libraries, rely on the compiler to find error, did not check, IDE would catch error, forgot, did not recognize, missed it, did not look at libraries, though it already had it, overlooked it, was nervous, attention to detail was lacking.
Reference Error	Was not sure about the format, did not know, did not see error, missed, did not pay attention to detail, overlooked, did not think about it, assumed to be there, did not think it was an error, overlooked, forgot.

Study 2: (RQ2): Does peer code review leads to an improvement in programming abilities of students?

The purpose of this question is to assess the impact of peer code review (Day 2) on students' improved understanding of programming concepts (Day 3) when compared against their baseline understanding (Day 1). To answer this question, the researcher compared student results during the Day 1 (pretest - students' understanding of programming concepts based on their ability to correctly predict program output in an externally developed programs with seeded faults) vs. during the Day 3 (posttest - type of naturally occurring errors made by students when developing

their own programs). Figures 14 and 15 show pre and post-test results respectively. Based on the results from the correlation analysis, the researcher found that students who were not able to predict the output of the code in the pretest committed fewer errors in the posttest. The Correlation between pre and posttest produces an r-value of 0.6196 and p-value = .004664. This is because of the PCR session that took place on Day 2 of the study.

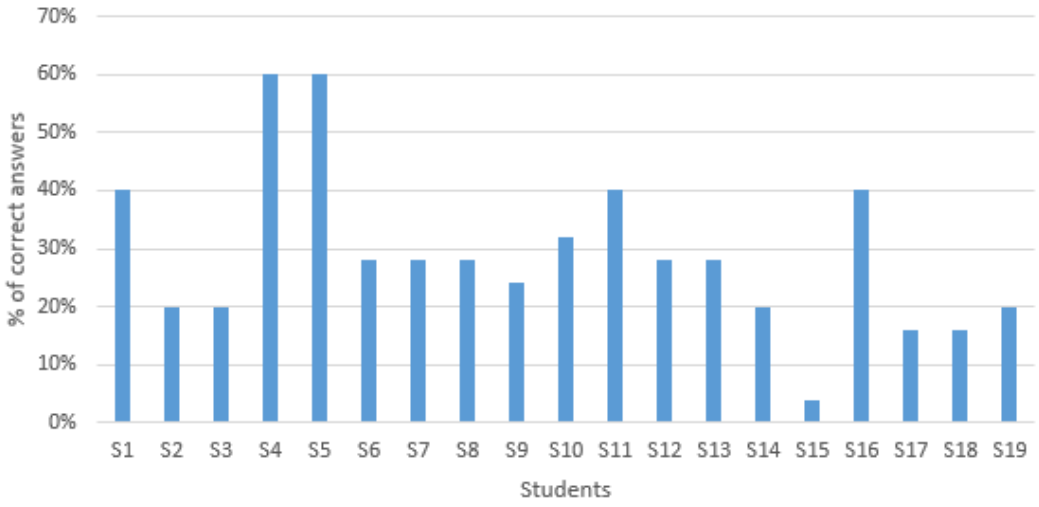


Figure 14: Pretest-Percentage of Correct Answers

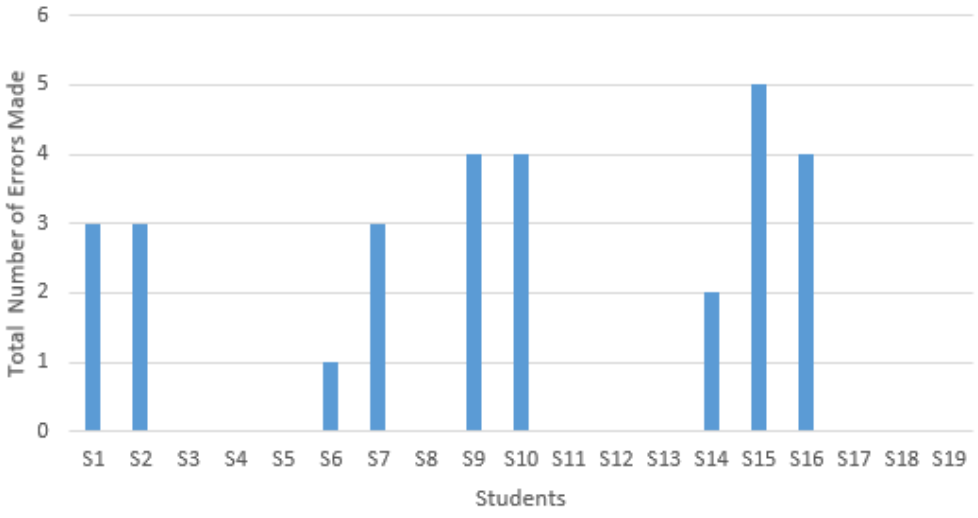


Figure 15: Posttest-Total Number of Errors Made

The researcher also performed correlation analysis between the number of errors identified during PCR (Day 2) and their Posttest (Day 3) performance. The result showed a strong negative correlation between the # of errors identified during the PCR activity and # of errors committed during their code development ($r= 0.88$, $p<0.001$). This result is interesting in the sense that, students that are able to identify errors when reviewing someone else's code are able to retain that information and are less likely to make errors when developing their own code. This correlation analysis was done separately for each error type (i.e., # of error of each type detected during PCR and # of errors of each type made during their own code). The results showed negative relationships for each error type but significant results ($p<.01$) were observed for Loop Counter, Instantiation and Reference error types. Again, this indicates that the more errors students identify during PCR exercise, the less likely they are to make those errors when developing their own code.

This also provides evidence that PCR can be an effective pedagogical intervention at helping students discover common error types (that their peers often commit). Instructors can use PCR to foster class discussion and help students understand ways of avoiding making those errors when developing their own code.

Study 2: (RQ3): *How can PCR be integrated into CS1 curriculum to improve learning?*

To answer this question, we analyzed qualitative data provided by students from the reflection sheet (PCR sessions on Day 2) and student rating on a 5-point interval scale from the end-of-study questionnaire. Only the most relevant study results are discussed below.

First, the researcher analyzed students' responses to questions on PCR exercise and its impact on code development. Based on the results, PCR was viewed as the most useful method at understanding common programming errors (an average usefulness rating of 4.26) and for code development (an average usefulness rating of 4.3). Students also reported that they felt PCR

improved their ability to review code and find real faults (an average usefulness rating of 4.1). Instructors involved in CS1 also mentioned that reflection instruments could foster discussion of common programming errors because everyone is reviewing the same code samples and reflecting on the same set of errors seeded in the code samples.

Next, the researcher analyzed students' responses in terms of their satisfaction with PCR and code development exercises. The results indicate that students were highly satisfied with both PCR exercise (an average satisfaction rating of 4.0) and code development exercise (an average satisfaction rating of 4.15). Next, the researcher collected students' qualitative data (written responses) to understand their perception of PCR exercise. The word cloud in Figure 16 was constructed from students' responses and highlights that students find PCR helpful and a good learning tool. Students specifically commented that they felt that PCR helped them improve their programming skills and that they will be able to retain knowledge of programming errors when developing code in future.

Figure 16: Students Perspective on PCR Exercise

The researcher also elicited student responses in terms of concepts that they still find challenging. Figure 17 shows that students struggle to understand problems relating to nested loops, two-dimensional arrays, class and object instantiation.

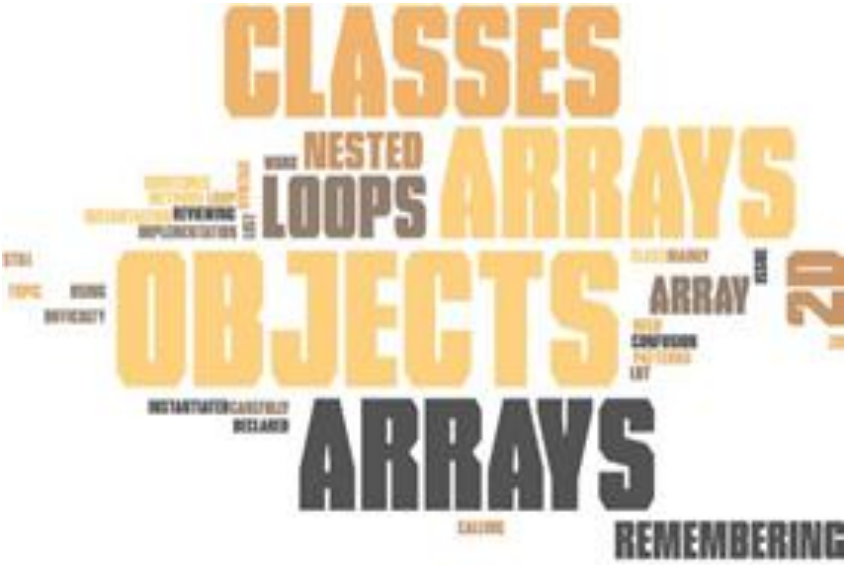


Figure 17: Difficult Topic Areas in CS1

Based on the results from this study, PCR sessions and reflection instruments can allow instructors to help students understand common programming mistakes, analyze most challenging CS1 topics, and align course materials to improve student understanding of those hard-to-comprehend topics. PCR can assist instructors at highlighting common mistakes made by beginning programmers by having student review code samples from previous students. PCR can also help students improve their understanding of basic program design when reviewing multiple solutions of the same program specifications. A larger collection of peer code snippets for different topic areas can aid critical thinking skills and engagement in CS1 class. Authors of this dissertation are interested in exploring if PCR can be integrated into a cyber-learning environment (specifically, SEP-CyLE that some of the authors are part of the development team [9]) to assist

students in knowledge acquisition of introductory programming and software engineering concepts.

4.2.7. Study 2: Discussion

This section provides additional insight of results presented in earlier sections and future plans for evaluating effective implementation of PCR in CS1 classrooms.

One of the major insights gained from the study is that PCR can be used to systematically improve students' understanding of programming concepts. Using PCR as a teaching approach helps students and instructors in the following ways:

- It encourages students to critically analyze programs in terms of program design as they try to either predict output or when trying to discover errors present in the code;
- It provides instructors a means of making students self-discover most common programming errors;
- It reinforces students to think about the errors they may have discovered in someone else's code when developing their own code which in turn can help students become better programmers;
- It also allows students to understand different ways in which a programming task can be solved and exposes them to different ways of algorithm designs;
- PCR process is a common practice in industry especially when students begin their jobs in industry. This exercise can also help students learn important skills on how to comprehend someone else's code and learn how to develop code that is easy to comprehend for others as well;
- PCR helps students to be more actively engaged in their own knowledge acquisition and can be expanded to be more collaborative if instructors choose to do so.

- The researcher gained some important insights into the CS1 challenging topics (that included multidimensional arrays, objects and class, nested loops) that instructors can use in future PCR exercises to improve students' understanding of these concepts. It must be noted that not all students found these topics difficult to learn. Additionally, during the PCR sessions, students became more comfortable with looping based on loops over ranges of integers but, in some cases, were still unable to complete problems that contained nested loops in 2-dimensional arrays. This information is very useful when designing future PCR exercises. The researcher also noticed that as students were able to retain information gained during each PCR session and were able to identify errors (that they had missed during the prior PCR sessions) even if the code being reviewed was more complex (compared to the code used in previous session). This is a very meaningful result that students are able to not only learn but also retain and use it when developing their own code (as evidenced by high quality code produced during the post-test).

Since this was an initial investigation, the researcher are planning to conduct future research that will allow us to develop a large enough repository of code artifacts with seeded faults that can be seamlessly integrated at different points in the CS1/CS2 programming course. The long-term goal of this research is to use code-reading techniques to develop a taxonomy of common syntactic and logical programming errors made in CS1 and to provide students with resources that will help them prevent students from committing most commonly occurring errors.

The researcher are also planning to incorporate PCR into our cyber-learning environment, called SEP-Cyle (Software Engineering and Programming – A Cyber Learning Environment) [9]. SEP-CyLE is an online repository that contains small chucks vetted learning objects (LOs). It is developed to assist instructors in teaching and to improve students' fundamental understanding of

introductory computer programming and software engineering concepts. SEP-Cyle uses a combination of learning engagement strategies (LES) to get students motivated to be more involved in learning programming concepts contained in LOs. One of the LES is collaboration, which facilitates interaction among individuals. Studies have shown that collaboration has positive results across different levels of education, ranging from young children doing their school projects like craft work in teams to university students working on development projects [12] [13]. the researcher are planning to include PCR as an individual and collaborative learning tool for students. This will be done by providing students with online PCR tutorials, which will comprise of a variety of software programming concepts and common programming errors. Students can then browse through LOs and tutorials; the researcher will then analyze their understanding of those concepts through quizzes.

Threat to Validity: This section attempts to address some of the threats to validity that may have affected the results of the research. A major threat is the relatively number of participating subjects, which limits the impact of results. The researcher plan to conduct additional studies to address this threat. Second, students may not have taken the posttest seriously because it was voluntary (i.e., did not have an impact on grades) and they might not have been motivated/inspired to do well on it. While our pretest was not consistent with the posttest, both pretest and posttest codes were similar in structure and complexity. Furthermore, the defects seeded in the code snippets were done manually may not be representative of naturally occurring faults. The researcher plan to conduct further studies to address these threats.

The current study showed that PCR is a useful way at helping students understand programming concepts and at improving their programming skills. While some concepts continued to be troublesome for the students through the end of the study, the researcher also saw evidence

of improvement over the course of the experiment. the researcher strongly recommend the integration of peer learning techniques, like the PCR discussed in this study, for reinforcing programming concepts and improving programming skills throughout the semester.

4.3. Study 3: Guided Peer Code Review CS161

Our initial study supports the idea that PCR can be used to support programming pedagogy [86] in higher education. Based on a feasibility study, it was found that finding errors in someone else's code reduced the likelihood of committing similar errors when students went on to develop their own code. However, students were not reporting many errors when they did not have a list of questions that guided them to identify errors, which resulted in students not pay attention to detail when reading someone else's code. Students indicated that they needed more guidance on what types of errors to look for when performing the code review. Including some set of expectations that can help students map those expectations to the source code under review.

This study aimed to empirically investigate whether guided PCR could be effective in teaching CS2 students programming concepts. Additionally, the researcher investigated the possibility of helping students to retain information (from PCR sessions) to times when they develop their own code. Furthermore, the researcher also wanted to investigate if pair-based review can help with knowledge retention or sharing among students. To accomplish this, guided PCR sessions were conducted where students were asked to follow a provided checklist in order to systematically check for programming errors when reading source code on a line-to-line basis.

4.3.1. Study 3: Study Goal

The essential goal of this research was to evaluate the usefulness of guided (checklist) PCR on CS2 students' learning with respect to the programming concepts of data structures and algorithms. More specifically, the researcher wanted to understand if checklist-based PCR could

guide students in understanding and overcoming common programming mistakes identified in the CS2 course. To accomplish these goals, four sessions of guided PCR study were conducted in one CS2 class wherein code samples were externally developed and seeded with faults (in consultation with the CS2 instructor). Students were asked to review code samples and log faults using the PCR checklist. Following each guided PCR session, students were asked to reflect on their review results, discuss errors, and ways to avoid them when developing their own programs. At the end of study, students provided feedback on the usefulness of guided PCR in an introductory programming course.

4.3.2. Study 3: Research Questions

The following research questions were formulated to accomplish the research goal:

- **Research Question 1 (RQ1):** *Which type of errors do students identify during guided PCR and why?*
- **Research Question 2 (RQ2):** *What is the effect of guided PCR on students learning of programming concepts in CS2 course?*
- **Research Question 3 (RQ3):** *What perceptions do students have on guided PCR and using it in CS2 course?*

4.3.3. Study 3: Participating Subjects

This study was conducted in a CS2 programming course at North Dakota State University. Sixteen students elected to participate in the study. The study sample was made up of CS and non-CS majors, most of whom completed CS1 or some other previous introductory programming course.

4.3.4. Study 3: Data Collection

The researcher collected both quantitative and qualitative data during the study run. The quantitative data included the number and category of errors found both by students individually and by students working in pairs during each of the guided PCR sessions. False positive errors were removed prior to the analysis by comparing the reported errors against the true errors that were seeded in each piece of code. The researcher also collected data regarding the number and type of errors overlooked during guided PCR sessions, students' performance on assignments and quizzes pre-, post- and in-between PCR sessions. These quantitative data were collected to allow us to analyze whether guided PCR had any effect on CS2 students' learning of data programming concepts (RQ1). In addition, it allowed us to determine if there was any improvement in students' performance throughout the CS2 course (RQ1).

The qualitative data collected during the study included student reflection reports, and responses on the end-of-study survey (completed at the conclusion of all PCR sessions). The qualitative data will allow us to answer RQ2.

4.3.5. Study 3: Results and Analysis

This section presents the analysis of the data collected from the study and is organized around three research questions listed on the experiment design section of this study.

Research Question 1 (RQ1): *Which type of errors do students identify during guided PCR and why?*

This research question aimed at evaluating the use of guided PCR as a teaching technique in CS2 classroom.

The researcher examined the number of true errors reported by students when using guided PCR to review each piece of code. This was done in order to identify errors students were able to

report, errors students were not able to report, and the effect the reflection / discussion had on subsequent peer review performance. Figure 18 provides an overview of the results. To normalize the performance across each PCR session and error category, Figure 18 shows a comparison of the percentage (%) of error reported by students (when working individually) for each category of errors in each piece of code. Some of the major observations based on the analysis of data collected during the study run are discussed below:

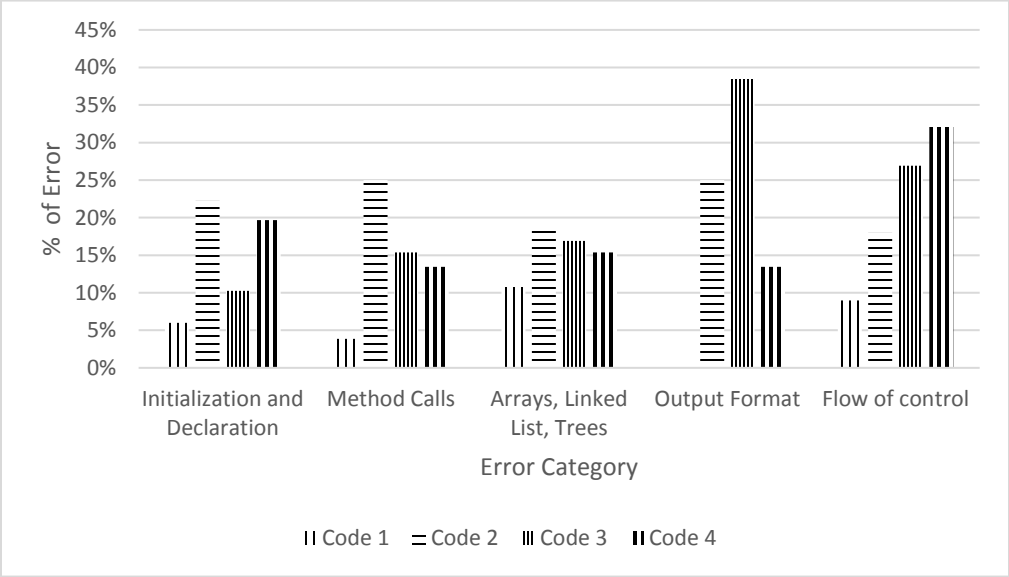


Figure 18: Percentage of Errors Reported by Category

Observation 1: The graph shows low error reporting percentage for Code 1 (doubly linked list). This was because the instruments for the pilot session of guided PCR did not accurately capture the error description or the location of the error reported by students. While these instruments include the error reporting sheet and the reflection sheet, students mentioned that they needed more guidance on how to use the checklist to find and record errors. The researchers used this feedback from students to improve the design of the previously mentioned instruments before conducting subsequent guided PCR sessions. In addition, based on students' comments from the error and reflection sheets, students found it hard to conform to the code review process and needed

pointers on how to read checklist questions and compare them against the code under review. This study is referring to the first PCR session as the pilot run. For subsequent PCR sessions, the study included extensive training and developed more clear directions to ensure that participants were able to use the checklist to perform the code review.

Observation 2: The error categories with the highest total percentage (> 60%) of error reported by students were: arrays/ linkedList/Trees, Output and flow of control. Initialization, declaration and method calls were the least reported error with 58% of the total errors being reported for each category by students. From Figure 18, the researcher notice that students did not report any errors in the pilot study (Code 1) for the output format error category. Because this is the first time students were exposed to a checklist, students did not thoroughly conform to the process of using the checklist in the pilot study. Flow of control category was the only error category where students maintained a progressive percentage increase in reporting errors. The researcher noticed a visible improvement in students' ability to find flow of control errors, which covers logical errors (something an integrated development environment (IDE) cannot find). It shows that students are able to critically analyze individual pieces of code from a logical perspective.

Observation 3: Based on the results in Figure 18, the researcher also noticed that the performance of students across each piece of code fluctuates for the majority of the error category with the exception of flow of control error type. This is because the complexity of each piece of code differs, which provided a different level of challenge for the students to recognize and report the errors. In addition, the topic being covered in each piece of code was progressively more challenging than the previous topic. There was a significant increase in the percentage of output format error reported for code 3 because students mentioned (in their reflections) that they seemed

to have a better understanding of this particular topic area. This is interesting because it allows students to self-assess their knowledge based on their code review performance and can also guide instructors at identifying a good baseline knowledge at an individual and a class level. Flow of control error category maintained a steady increase in the overall percentage of errors reported by students. Majority of the students who participated in the study have completed CS1 and are familiar with using an IDE. Some errors that would normally be caught by the compiler, students may have grown accustomed to having the IDE report those and therefore be less prone to identifying them manually. This lead to students progressively reporting more flow of control (logical) errors than they do others.

Observation 4 (Based on reflection sheet output): The researcher analyzed student comments gathered from the reflection sheets to gain insight on the reasons students were unable to locate particular types of errors. Across each category of errors, more than 90% of the responses agree that they "*did not notice the error*"; in other words, they were unable to report those errors. Based on the survey questionnaire administered at the end of the guided PCR code sessions, the researcher also discovered that some students thought the checklist needed clearer guidelines as some of students abandoned it mid-way and used an ad-hoc approach. This is not surprising but does inform how improvements can be made to the checklist. In the future, to help students make better use of the checklist, the researcher will make the steps more specific, and provide training or a training document on guided PCR.

Research Question 2 (RQ2): *What is the effect of guided PCR on students learning of programming concepts in CS2 course?*

Observation 5 (Correlational analysis): The researcher performed correlation analysis between the total number of category errors reported per topic during guided PCR and the students'

performance on assignments covering specific topics. Based on the results in Table 9, students who are able to find more errors during the PCR session on a particular topic are able to score better on their lab assignments that covered the same topic. There was a positive (but non-significant, with the exception of Code Trees assignment) correlation across all four PCR sessions and their subsequent assignment performance. This suggests that students who are able to find more errors in someone else's code were able to retain and use that information when developing their own code. This also reinforces that the performance on the PCR session can be a good predictor of students' understanding of a particular topic.

Table 9: Correlation Analysis between Specific Topics Covered in Guided PCR and Specific Assignment

PCR 1 vs lab1 Assignment	PCR 2 vs lab2 Assignment	PCR 3 vs lab3 Assignment	PCR 4 vs lab4 Assignment
r = 0.1427	r = 0.486	r = 0. 6381	r = 0.522
p = 0.5848	p = 0.047935	p = 0.005846	p = .031607

Research Question 3 (RQ3): *What perceptions do students have on guided PCR and using it in CS2 course?*

At the end of the four-week guided PCR sessions, a feedback survey was conducted. The goal of the survey was to gain insights on how useful the error checklist was in PCR and how impactful it was on students' understanding of programming. Students rated their responses on a 5-point Likert scale [1= Strongly Disagree; 2= Disagree; 3= Neither Agree nor Disagree; 4= Agree; 5= Strongly Agree]. On examining the survey (see Figure 19), the researcher found that over 68% of the respondents indicated that PCR assisted in avoiding errors during their own code development, and 75% of respondents agreed that PCR helps in learning programming concepts. When asked if PCR is useful in verifying code quality, 100% of the students unanimously agreed.

There was positive feedback on the use of checklists in PCR with respect to how students felt about its application when developing their own code, when discussing results with peers, and when performing individual code reviews. Most of the students strongly agree that the checklist was useful when performing PCR. Figure 19 shows the topic areas students were able to improve their understanding of programming concepts when using guided PCR.

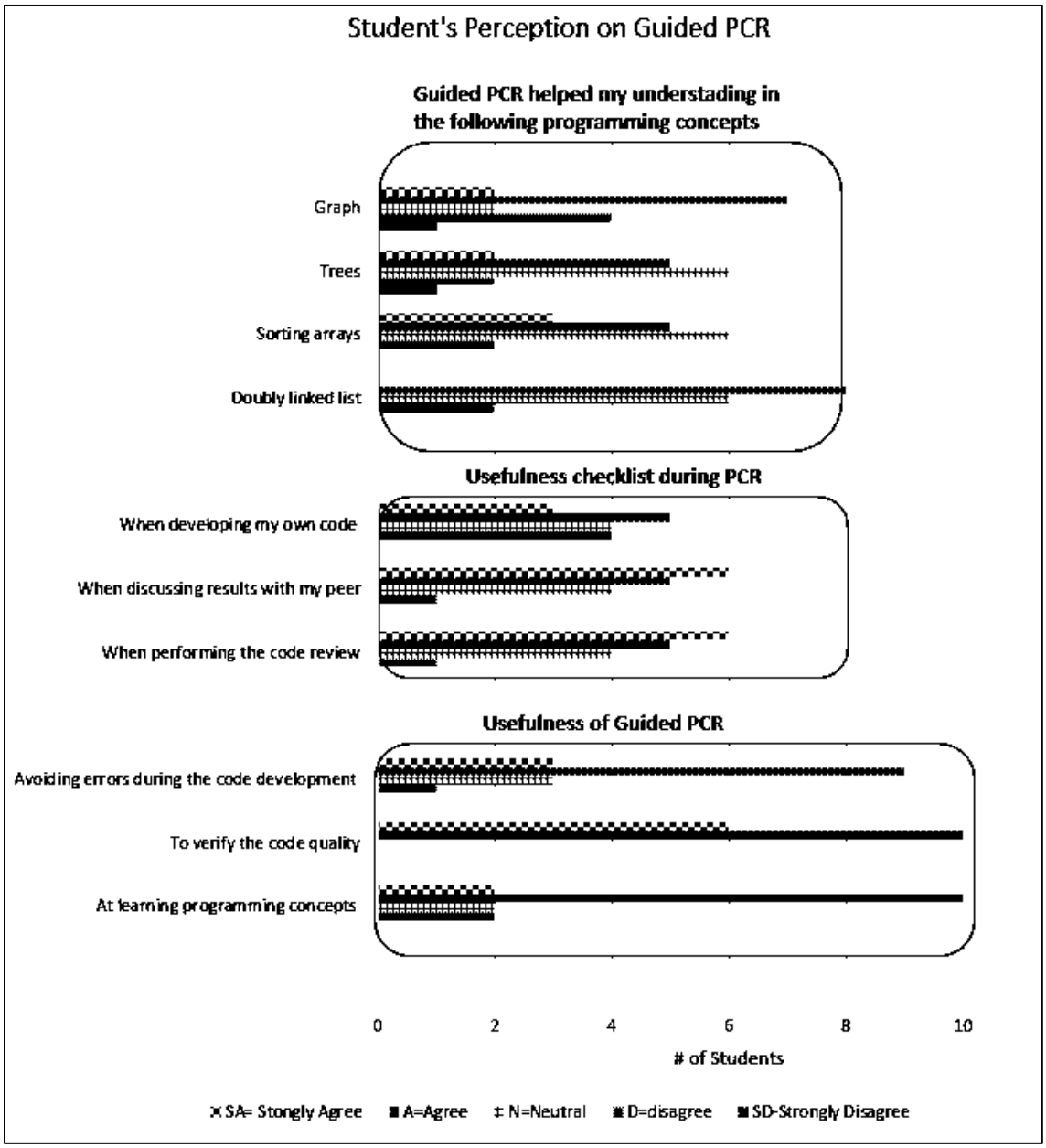


Figure 19: Survey Results on Students' Perception of PCR

Overall, the study showed an improvement in students understanding CS2 programming concepts, specifically in the areas of graph searching, trees traversal, and array sorting. Survey results show that over 80% of the students would recommend using PCR in CS2 classroom.

4.3.6. Study 3: Discussion

The researcher performed independent t-test analysis on two separate CS2 courses. One course was administered in summer 2018 and the other in summer 2019. The same instructor taught both. The purpose of the test was to determine if with guided PCR students receive higher grades on their final exam (Summer 2019) versus the traditional teaching method that was implemented in the CS2 course in the previous years (summer 2018). The result shows that students in guided PCR achieve higher grades (The p-value = 0.016711) on their final exams compared to the previous years (when PCR was not used). There was an overall 21.666% percentage increase in students' final exam for guided PRC compared to a CS2 course that did not participate in the experiment. While not all of this may be attributable to the use of guided PCR, the improvement in the performance is noteworthy and warrants further investigation in a controlled study.

The researcher also conducted a group review session and analyzed the new errors that were reported during the group review step of the PCR session. The researcher tabulated the average number of errors reported by students during the individual review vs. the average number of new errors reported during the group review (when students were paired). This was done for all four guided PCR sessions and the results are shown in Table 10. Based on these results, students though may be able to exchange their individual error performance with each other, are not able to find a large number of new errors (that none of the pair members had previously discovered).

Table 10: Individual vs Group Errors Found

	PCR Code 1	PCR Code 2	PCR Code 3	PCR Code 4
Individual	23	78	72	72
Group	13	10	3	3

At the end of the summer 2019 survey, the researcher asked students to provide additional comments. The comments provided by students gave us additional insights on the usefulness of guided PCR in CS2 course. Below are some of the comments students provided followed by insights:

A. Students' feedback: Group Discussion helps understand how each piece of code functioned.

Insights: Students mentioned that they were able to think critically on each piece of code when working in pairs because they were able to discuss their perspectives and share knowledge. While this may not have resulted in finding a large number of additional errors, students enjoyed discussing their individual performances with their peers. This activity also helped students at discuss different things to look for when reviewing code and ways in which they can solve data structures and algorithms problems.

B. Students' feedback: Guided PCR was more helpful when code was more complex in design

Insights: This exercise exposes students to the experience of performing code review irrespective of the code complexity. Students are able to see the benefit of using a checklist when they are given a more complex program to review versus a less complex code. This experience student can take with them to the software industry. Additionally, instructors can use this feedback from students to determine more difficult to comprehend topics that they should target when using guided PCR in CS2 course.

C. Students' feedback: Error checklist needs improvement

Insights: The researcher discovered that some students were unable to follow the error checklist thoroughly because steps were unclear and some of the students found the checklist distracting. The researcher plans to make improvements on the error checklist prior to conducting further experiments. The researcher will also need to conduct more extensive training that will communicate to the students the purpose of the checklist in PCR and demonstrate to the students the correct way to use the checklist.

D. Students' feedback: *It would be good to keep doing it as it helped us make fewer errors when writing own code*

Insights: Students find guided PCR exercise valuable in teaching CS2 programming concepts. Students felt that they could benefit from participating in additional PCR sessions because they are able to self-assess their understanding of computer programs when reviewing their performance against the true list of seeded errors. This when done over a longer period of time can help students become better programmers as they are more mindful of mistakes they have discovered previously.

E. Students' feedback: Incorporate more CS2 topics

Insights: Students commented that PCR could have benefitted their understanding of other CS2 topics. Some of these include teaching the concepts of static variable and polymorphism. This shows that students are experiencing difficulty in understanding those mentioned programming concepts and guided PCR can be used in the future to improve students' understanding of these concepts.

In this dissertation, the researcher described how guided PCR was utilized to assist in students' knowledge acquisition of programming concepts in CS2 course. This dissertation have

provided empirical data to support the increase in students' performance on assessments post guided PCR. The study shows that the current research was able to achieve the following:

- Assist students in learning data structure and algorithms programming concepts
- Improve students' performance on lab assignments and final exam
- Facilitate communication among peers

Threat to Validity: Factors that may have affected the results of the research includes-: The relatively small number of participating subjects, which limits the impact of the results. The researcher plan to conduct additional studies to address this threat. Second, the defects seeded in the code snippets were done manually may not be representative of naturally occurring faults. The researcher plan to conduct further studies to address these threats.

The researcher plan to improve on the current error checklist and incorporate a training process that covers the following:

- Convey to the students the importance of guided PCR
- Demonstrate to the students the proper way of using guided PCR

The researcher also plan to conduct further investigation on the usefulness of guided PCR in a controlled study.

Overall students had a positive experience using guided PCR and recommended frequent use of this exercise in CS2 course by instructors.

4.4. Study 4: Team-Based Guided Peer Code Review CSE201

In this study, the researcher aimed to empirically investigate whether guided PCR could be effective in teaching computer software engineering students (CSE) programming concepts and reinforces coding standards. Furthermore, the researcher aimed to identify the types of errors made by students in CSE classroom. Additionally, the researcher also wanted to investigate if pair-based

review can help with recording more errors. To accomplish this, a guided PCR session was conducted where teams of students were asked to follow a provided checklist in order to systematically check for programming errors and programming standards when reading source code developed by the team (self-review).

4.4.1. Study 4: Study Goal

The essential goal of this research was to evaluate the usefulness of guided (checklist) PCR on CSE students' learning with respect to the programming concepts and coding standards. More specifically, the researcher wanted to understand if checklist-based PCR could guide students in finding errors and missed coding standards in CSE course. To accomplish these goals, a session of guided PCR study was conducted in three CSE classes taught by the same instructor. Students developed Java code in teams of two using a requirement document given by the instructor. Students were asked to review code developed by their team and log programming and coding standards errors using the PCR checklist.

4.4.2. Study 4: Research Questions

The following research questions were formulated to accomplish the research goal:

- ***Research Question 1 (RQ1):*** What errors do student programmers make when developing code while working in teams?
- ***Research Question 2 (RQ2):*** How effective are Software Engineering students at finding errors when using PCR checklist?

4.4.3. Study 4: Participating Subjects

This study was conducted in a CSE course at Miami University, Ohio. 70 students elected to participate in the study (35 teams of two). The study sample was made up of CS and non-CS majors, most whom completed CS1 or some other previous introductory programming course.

4.4.4. Study 4: Study Procedure

The study was conducted in three sections of CSE classes and lasted for approximately 50 minutes. During the PCR session, teams were given their own code they developed to review. The source lines of codes (SLOC) for each Java file varied by team. The Java codes were developed by teams of students from a lab requirement document that was given by the instructor of the three CSE sections and included OOP concepts such as arrayList, strings, file handling, string handling, string matching and substring matching. The study procedure included the following major steps:

Step 1: Introduction to PCR: Students were introduced to guided PCR and the purpose of the study.

Step 2: Team-based Peer Code Review

4.4.5. Study 4: Data Collection

The researcher collected quantitative data during the study run. The quantitative data included the number errors found both by teams during each of the guided PCR sessions and the number of errors found that correspond to each PCR question. These quantitative data were collected to allow us to analyze whether guided PCR was effective in helping students to find errors, learning programming concepts and coding standards (RQ2). The qualitative data collected during the study included student reflection reports. The qualitative data will allow us to answer RQ2.

4.4.6. Study 4: Results and Analysis

This section presents the analysis of the data collected from the study and is organized around two research questions listed on the experiment design section of this study.

Research Question 1 (RQ2): *What errors do student programmers make when developing code while working in teams?* This research question aimed to evaluate the types of errors computer software engineering students make when developing code in teams. The researcher examined the number of true errors recorded by a grader when using the requirement document to grade team lab assignment. Figure 20 provides an overview of the results.

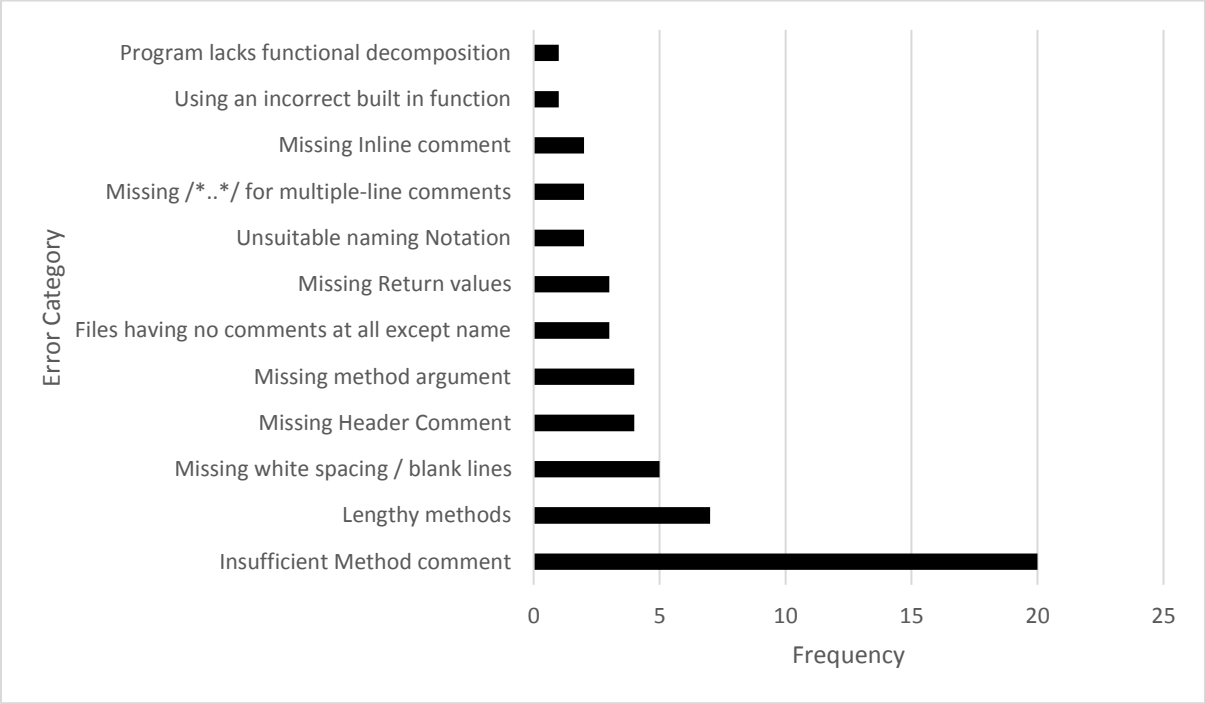


Figure 20: Errors CSE Students Make When Developing Code in Teams

Discussion: The graph shows that most of the errors teams made when developing their own code related to coding standards. This was because CSE 201 students would have completed CS1 programming courses and had a good grasp on concepts such as OOP (e.g. arrayList, strings, file handling, string handling, string matching and substring matching). The error category with the highest frequency was insufficient method comments. The data showed no recorded errors for any form of logical errors relating to arrays or the program itself.

Research Question 2 (RQ2): *How effective are Software Engineering students at finding errors when using PCR checklist?*

This research question aimed to evaluate the use of guided PCR as a teaching technique in CSE2 classroom. The researcher examined the number of true errors reported by the grader when grading team lab projects and the number of errors reported by teams when using guided PCR to review their own lab project. This was done in order to identify the number of errors students were able to report, and the number of errors students were not able to report, Figure 21.

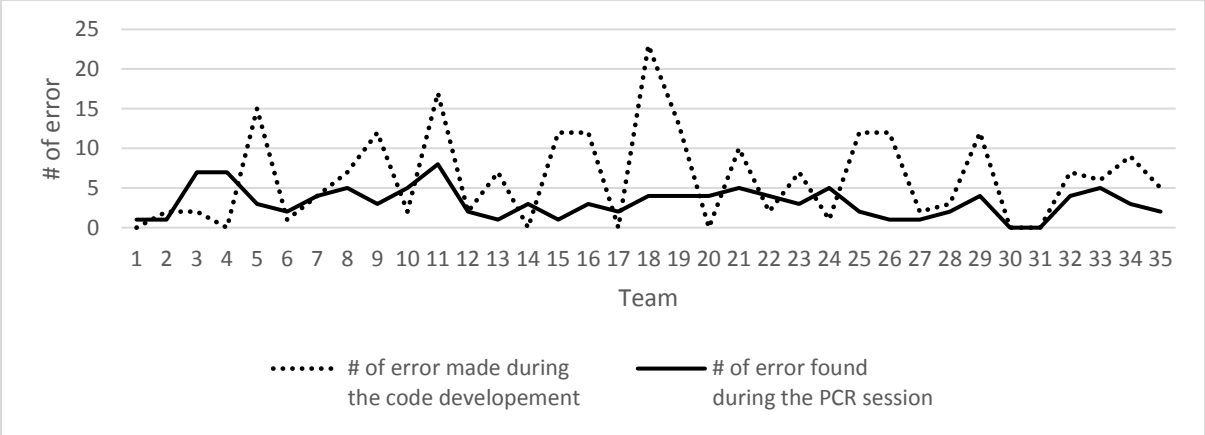


Figure 21: Number of Errors Made during Code Development vs. Number of Errors Found during PCR session

Discussion: From Figure 21 the researcher noticed that the number of errors teams reported during guided PCR session was in most cases significantly lower than the number of errors made during their code development. In a few cases some teams (that is, team 3,4,10, 17, 20, 22, 24) were able to identify more errors than what was recorded by the grader after their code development. This is because the guided PCR contained more questions than the lab requirement document.

5. DISCUSSION

The studies conducted in this dissertation provided evidence that active learning engagement strategies embedded in a cyber-learning environment (SEP-CyLE) improved students' engagement and learning. It was found that using SEP-CyLE in CS1 classrooms could help motivate students to be more involved in their own learning. Additionally, evidence suggests that PCR can be adapted to foster collaborative learning through peer-to-peer interaction and collaboration through social constructs.

The systematic literature review (SLR) results revealed that Gamification had a positive impact on student learning. This dissertation identified most commonly used GEs in different STEM courses and provided recommendation for adapting them to SEP-CyLE infrastructure. The SLR process also evaluated primary studies, analyzed the application of game mechanics in an educational context and its impact on user motivation, engagement, participation, etc. The results show that integrating gamification elements into STEM education helps in achieving positive educational outcomes. Particularly, points and badges when used in SEP-CyLE has a positive impact on student engagement and learning.

The results obtained during the analysis of the primary studies show that the existing literature on gamification is more utilized in computer science level one courses (CS 1). Most of the studies examined CS1 programming courses. This shows an important gap that must be addressed since other STEM courses (example chemistry and physics) and other areas of computer science (such as CS2 and software engineering) have not been studied to their full extent. Another shortcoming identified was that most of the studies make use of the most common GE, which are; leaderboard, badge and points. This reflects that researchers do not focus on intrinsic motivation such as psychological and behavioral variables. The use of GE which is too simple can lead to

some students not finding gamified learning activity engaging hence enthusiasm for utilizing the platform diminished after two or three days.

These shortcomings of previous studies motivated the design and implementation of follow-up studies to investigate the impact of learning engagement strategies on student learning across different courses and institutions. A variety of learning engagement strategies were developed and analyzed across multiple studies at North Dakota State Universities and across different universities. The results from these studies showed that combining gamification with collaborative learning helps students' learning and engagement.

Study 2 revealed that introducing collaborative learning in CS1 classroom as an active teaching technique helps in the improvement of students' knowledge of programming concepts. In addition, collaborative learning helps students and instructors in the following ways:

1. It encourages students to critically analyze programs in terms of program design as they try to either predict output or when trying to discover errors present in the code
2. It provides instructors a means of making students self-discover most common programming errors
3. It reinforces students to think about the errors they may have discovered in someone else's code when developing their own code which in turn can help students become better programmers
4. It also allows students to understand different ways in which a programming task can be solved and exposes them to different ways of algorithm designs
5. Collaborative learning can also help students learn important skills on how to comprehend someone else's code and learn how to develop code that is easy to comprehend for others as well.

Study 3 showed that the implementation of collaborative learning in CS2 classroom had a positive impact on students' final exams compared to the previous years where no collaborative technique was used in the classroom. Furthermore, team collaboration seemed to be more effective for CS2 students because students were able to think critically when working in pairs due to sharing of knowledge. This shows that care must be taken w.r.t the implementation of any collaborative techniques in CS learning material.

The researcher also found that SEP-CyLE in a variety of way helps students improve their conceptual knowledge of computer programming. Collaborative learning proved to be more advantageous to CS1 students; however, CS2 students were more engaged in verbal conversation with peers when working in groups, which led to a better understanding of a program. Digital LOs and PCR helped students' progress in their assignments and also helped at improving their course grades. Based on the results from the studies, SEP- CyLE could be extended to incorporate collaborative learning (PCR) for easier and frequent dissemination of course material. This would help instructors to quickly extra feedback and provide additional course materials to teams of students in a cyberlearning environment that would support peer-to-peer learning.

Motivated by the results of learning engagement strategies shortcoming in SEP-CyLE, the researcher are planning to include more digital learning objects in SEP-CyLE that will focus on students' learning of programming concepts in a collaborative manner. The researcher are also planning to include more GEs for example, badges and levels to conduct further studies on their impact on students' motivation, learning and engagement. Our SEP-CyLE infrastructure can be used by other interested computer science educators and researchers to support pedagogy in their classrooms.

6. CONCLUSION

This section discusses the major contribution of the work described in this dissertation to Computer science and Software Engineering education research and practice. This section also enlists the publications that will be the output of this dissertation work.

6.1. Contribution to Research and Practice

The main goal of this thesis is to enrich computer science education by deepening our understanding of learning engagement strategies: Gamification, and Collaborative Learning. This work will add to the body of knowledge in computer science education by providing a better understanding of the implementation different learning engagement strategies: Gamification, and Collaborative Learning. This work presents detailed insights into ELES in SEP-CyLE that can be used to improve student motivation and will be supported by the results of the studies (described in Chapter 4).

The researcher believe that if these ELEs are successfully used, they can support learning by enhancing students' engagement and motivation in a cyberlearning environment and face-to-face classroom. The results from proposed work will determine the extent of the usefulness of SEP-CyLE in a variety of classroom settings. This work contributes to the computer science pedagogy by providing evidence of the benefits of learning engagement strategies that can be used to support student learning individually and through collaboration with peers. Our SEP-CyLE infrastructure can be used by other interested computer science educators and researchers to support pedagogy in their classrooms.

6.2. List of Publications

This section describes the publications that resulted from the work done for this dissertation.

1 Brown, T., Walia, G., Singh, M., Nassareddygari, M., and Radermacher, A. J. “Effectiveness of Using Guided Peer Code Review to Support Learning of Programming Concepts in CS2 Course: A Pilot Study”, 2020 ASEE Annual Conference & Exposition. June 21 – 24, Montreal, Quebec, Canada. Abstract accepted

2 Brown, T., “Guided Peer Code Reviews in CS1/CS2 and SE Curricula”. IEEE Conference on Frontiers in Education Doctoral Symposium, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA.

3 Brown, T., Nassareddygari, M., Singh, M., and Walia, G. “Using Code Review to Improve Programming Skills of Students in an Introductory CS Course”. IEEE Conference on Frontiers in Education, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA.

4 Reddy, M., Brown, T., Walia, G., “Using Digital Learning Content Embedded in a Cyber Learning Environment to Support CS1 Pedagogy”. IEEE Conference on Frontiers in Education, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA.

5 Kaur, R., Singh, M., Brown, T., Reddy, M., Walia, G., “Using Associated Rule Mining Algorithm to Analyze Patterns in Students Learning Content in CS1 Course”. IEEE Conference on Frontiers in Education, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA.

REFERENCES

[1] Auvinen T., Hakulinen L. & Malmi L. (2015). Increasing Students' Awareness of Their Behavior in Online Learning Environments with Visualizations and Achievement Badges. IEEE Transactions on Learning Technologies (TLT), Volume 8, Issue 3, pp. 261-273.

[2] Barata G., Gama S., Jorge J. A., & Gonçalves D. (2013a): Improving participation and learning with gamification. Gamification 2013: 10-17.

[3] Barata G., Gama S., Jorge J., & Goncalves D. (2013b). Engaging Engineering Students with Gamification. 2013 5th International Conference on Games and Virtual Worlds for serious Applications (VS-GAMES), IEEE

[4] Barnes T., Powell, E., Chaffin A., Godwin A., & Richter, H. (2007). "Game2Learn: Building CS1 learning games for retention". In ITiCSE '07: Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (pp. 121–125). New York, NY: ACM.

[5] Buisman A. L. D., & van Eekelen M. C. J. D.: Gamification in Educational Software Development. CSERC 2014: 9-20.

[6] Burkey D. D., Anastasio D.D., & Suresh A.. (2013). Improving Student Attitudes Towards the Capstone Laboratory Course using Gamification. 120th ASEE Annual Conference & Exposition

[7] Claypool K., & Claypool M., "Teaching Software Engineering Through Game Design", Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, June 27-29, 2005, Caparica, Portugal

[8] Connolly T. C., Boyle E. A., Hainey T., McArthur E., & J. M. Boyle, (2012). A Systematic Literature Review of Empirical Evidence on Computer Games and Serious Games. Computers & Education, 59, 661e686.

[9] Chang-lau R. & Clarke P. J.. Software Engineering and Programming Cyberlearning Environment (SEP-CyLE), 2018. https://stem-cyle.cis.fiu.edu/instances (retrieved Jan. 2018).

[10] Danny P. (2013). The Effect of Virtual Achievements on Student Engagement. In Proceeding of the SIGCHI Conference on Human Factors in Computing Systems. Changing Perspectives, April 27–May 2, 2013, Paris, France, pp. 763-772

[11] Deloitte. (2011), Tech Trends 2012 Elevate IT for Digital Business, Deloitte, available at: http://www.deloitte.com/assets/DcomUnitedStates/LocalAssets/Documents/us_cons_techtrends2012_013112.pdf

[12] Deterding S., Khaked R., Nacke L.E. & Dixon D.. (2011a). Gamification: Toward a Definition. Chi 2011 Gamification Workshop Proceedings, pages 12-15

[13] Deterding S., Dixon D., Khaled R., & Nacke L., (2011b). "From game design elements to gamefulness: defining gamification". In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, pp 9-15.

[14] Dicheva D., Dichev C., Agre G., & Angelova G. (2015). Gamification in Education: A Systematic Mapping Study. Educational Technology & Society, 18 (3), 75–88

[15] Dominguez A., Saenz-de-Navarrete J., de-Marcos L., Fernandez-Sanz L., Pages C., & Martinez-Herraiz J.J. (2013). Gamifying Learning Experiences. Practical Implications and Outcomes. Computer and Education, vol 63, April 2013, pages 380-392. https://doi.org/10.1016/j.compedu.2012.12.020

[16] Faghihi U., Brautigam A., Jorgenson K., Martin D., Brown A., Measures E., et al. (2014). How Gamification Applies for Educational Purpose Specially with College Algebra. Procedia Computer Science, vol. (41), 2014, pages 182-187. BICA 2014. 5th Annual International Conference on Biologically Inspired Cognitive Architectures. https://doi.org/10.1016/j.procs.2014.11.102

[17] Fitz-Walter Z., Tjondronegoro D., & Wyeth P.: Orientation Passport: using Gamification to Engage University Students. OZCHI 2011: 122-125.

[18] Haaranen L., Ihantola P., & Hakulinen L., A. Korhonen, How (not) to Introduce Badges to Online Exercises, Proceedings of the 45th ACM Technical Symposium on Computer Science Education, March 05-08, 2014, Atlanta, Georgia, USA.

[19] Halan S., Rossen B., Cendan J., & Lok B. (2010). High Score!- Motivation Strategies for User Participation in Virtual Human Development. International Conference on Intelligent Virtual Agents 2010, pages 482-488

[20] Hamari J., Koivisto J., & Sarsa H., "Does Gamification Work?–A Literature Review of Empirical Studies on Gamification" in System Sciences (HICSS), 2014 47th Hawaii International Conference on, 2014, pp. 3025–3034.

[21] Harrington B.: TrAcademic: Experiences With Gamified Practical Sessions for a CS1 Course. WCCCE 2016: 25:1-25:2.

[22] Huang W. H-Y., & Soman D.. (2013). A Practitioner's Guide to Gamification of Education, Behavioral Economics in Action Report Series, Rotman School of Management, University of Toronto.

[23] Ibanez M.B., Di-Serio A., & Delgado-Kloos C..: “Gamification for Engaging Computer Science Students in Learning Activities: A Case Study”. IEEE Trans. Technol. 7(3), 291 301(2014).

[24] Jurado J., Fernandez A., & Collazos C. A. (2015) Applying Gamification in the Context of Knowledge Management. In: i-KNOW '15 Proceedings, 43, ACM (2015).

[25] Kapp K. M. (2012). The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education, Pfeiffer, 1 edition

[26] Khaleel F. L., Sahari N., Meriam T. S., & Ismail A. (2015) “The Study of Gamification Application Architecture for Programming Language Course”, ACM IMCOM 2015 - Proceedings. Association for Computing Machinery, Inc, 2015. A17.

[27] Kim E., Rothrock L., & Freivalds A.. (2016). The Effects of Gamification on Engineering Lab Activities. 2016 IEEE Frontiers in Education Conference (FIE)

[28] Kim S. (2015). Team Organization Method using Salary Auction Game for Sustainable Motivation. Sustainability 2015, 7, 14358-14370. https://doi.org/10.3390/su71014358

[29] Kolikant Y. B. (2010). Digital Natives, Better Learners? Students’ Beliefs About How the Internet Influenced Their Ability to Learn. Computers in Human Behavior, 26(6), 1384-1391. doi:10.1016/j.chb.2010.04.012

[30] Krause M., Mogalle M., Pohl H., & Williams J. J.: A Playful Game Changer: Fostering Student Retention in Online Education with Social Gamification. In Proceedings of the Second ACM Conference on Learning@ Scale Conference (L@S’15). Vancouver, Canada: ACM Press

[31] Lambruschini B. B.., & Pizzaro W. G. (2015). Tech – Gamification in University Engineering Education. The 10th International Conference on Computer Science & Education (ICCSE 2015)

[32] Lee J.J., & Hammer J.. (2011). Gamification in Education: What, How, Why Bother? Academic Exchange Quarterly, vol. 15, page 146, 2011

[33] Lehtonen T. & Aho T. & Isohanni E. & Mikkonen T. (2015). On the Role of Gamification and Localization in an Open Online Learning environment: javala experiences. 50-59. 10.1145/2828959.2828973.

[34] Levin D., & Arafeh S., (2002). The Digital Disconnect: The Widening Gap Between Internet-Savvy Students and Their Schools. Washington DC: Pew Internet & American Life Project

[35] Means B., Toyama Y., Murphy R., Bakia M., & Jones K. (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, 2009

[36] Mekler E.D., Bruhlmann F., Opwis K., & Tuch A. N.. (2013). Do Points, Levels and Leaderboards Harm Intrinsic Motivation?: an Empirical Analysis of Common Gamification Elements. In Proceeding of the First International Conference on Gamiful Design, Research, and Applications, pages 66-73

[37] Monteiro B.S., Gomes A.S., & F.M. Mendes Neto. (2014). Youubi: Open Software for Ubiquitous Learning. Computers in Human Behavior, 55(2016), 1145-1164

[38] Morrison B. B. & DiSalvo B. J.. (2014). Khan Academy Gamifies Computer Science. SIGCE ’14: Proceeding of the 45th ACM Technical Symposium on Computer Science Education. ACM

[39] NSF Task Force on Cyberlearning. Fostering Learning in the Networked World: The Cyberlearning Opportunity and Challenge, 2008

[40] NSF Task Force on Cyberlearning & Workforce Development. A Report of the National Science Foundation Advisory Committee for Cyberinfrastructure, 2011

[41] Osborne J., (2003). Attitudes Towards Science: A Review of the Literature and its Implications. International Journal of Science Education, 25(9), 1049–1079.

[42] Ortiz M., Chiluiza K. & Valcke M. (2016). Gamification in Higher Education and STEM: A Systematic Review of Literature. 8th Annual Conference on Education and New Learning Technologies – Edulearn 16 at Barcelona, Spain

[43] Prato A., Best H., & Scurry I. (2016). 4th Family STEM Program: Sports Science. In Integrated STEM Education Conference (ICSEC), 2016 IEEE, pages 102-103. IEEE

[44] Rincon-Flores E.G., Gallardo K., & Fuente J.M. de la . (2018). Strengthening an Educational Innovation Strategy: Processes to Improve Gamification in Calculus Course Through Performance Assessment and Meta-Evaluation. IEJME-Mathematics Education 2018, Vol 13, No. 1, 1-11

[45] Rincon-Flores E.G., Ramirez-Montoya M. S., & Mena J. (2016). Challenge-Based Gamification and its Impact in Teaching Mathematical Modeling. TEEM ’16 Proceeding of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality, pages 771-776

[46] Ruiperez-Valiente J.A., Munoz-Merino P. J., & Delgado-Kloos C. (2017). Detecting and Clustering Students by their Gamification Behavior with Badges: A Case Study in Engineering Education. International Journal of Engineering Education 33(2), pages 816-830

[47] Sepehr S., & Head M.. (2013). Competition as an Element of Gamification for Learning: an Exploratory Longitudinal Investigation. Published in Proceeding of the First International Conference on Gameful Design, Research, and Application, pages 2-9. Toronto, Ontario, Canada

[48] Sillaots M.. (2014). Achieving flow through Gamification: A Study on Re-designing Research Methods Courses,

[49] Simpson R. D., & Oliver J. S., (1990). A Summary of Major Influences on Attitude Toward and Achievement in Science among Adolescent Students. Science Education, 74(1), 1–18.

[50] Villagrasa S., Fonseca D., & Duran J. (2014). Teaching Case: Applying Gamification Techniques and Virtual Reality for Learning Building Engineering 3D Arts. TEEM ’14 Proceeding of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality, pages 171-177. 10.1145/2669711.2669896

[51] Souza M.R. de A., Veado L., Moreira R. T., Figueiredo E. & Costa H. (2018). A Systematic Mapping Study on Game-Related Methods for Software Engineering Education. Information and Software Technology 95 (2018) 201-218

[52] Wallis P., & Martinez M. S.: Motivating Skill-Based Promotion with Badges. SIGUCCS 2013: 175-180.

[53] H. M. Walker, “ACM Retention Committee Retention of Students in Introductory Computing Courses: Curricular Issues and Approaches,” ACM Inroads, vol. 8, pp. 14-16, 2017.].

[54] Ryan Rybarczyk, Lingma Acheson. 2019. Interactive Peer-Led Code Reviews in CS2 Curricula. In Proceedings of the 50th ACM Technical Symposium on Computer Science

Education (SIGCSE '19). February 27-March 2, 2019, Minneapolis, MN, USA, ACM, NY, NY, USA

[55] T. Beaubouef, and J. Mason, 2005. "Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations". SIGCSE Bulletin 37, 2 (Jun. 2005), 103-106.

[56] Beaubouef, T. & J. Mason. Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. SIGCSE Bull. 37, 2 (Jun. 2005), 103-106.

[57] K. Päivi & M.Lauri. (2006). Why Students Drop Out CS1 course? 2006. 97-108. 10.1145/1151588.1151604.

[58] Bagel, A., and Simon, B., Struggles of New College Graduates in their First Software Development Job. Proceedings of the 39th SIGSCE Technical Symposium on Computer Science Education, pages 226-230 Portland, OR, USA – March 12-15, 2008

[59] Brechner, E. (2003). Things They Would Not Teach Me of in College: What Microsoft Developers Learn Later. In Proceedings of OOPSLA '03. ACM.

[60] Barata G., Gama S., Jorge J., & Goncalves D. (2013b). Engaging Engineering Students with Gamification. 2013 5th International Conference on Games and Virtual Worlds for serious Applications (VS-GAMES), IEEE

[61] Sepehr S., & Head M.. (2013). Competition as an Element of Gamification for Learning: an Exploratory Longitudinal Investigation. Published in Proceeding of the First International Conference on Gameful Design, Research, and Application, pages 2-9. Toronto, Ontario, Canada

[62] Lee J.J., & Hammer J.. (2011). Gamification in education: What, How, Why Bother? Academic Exchange Quarterly, vol. 15, page 146, 2011

[63] Kapp K. M. (2012). The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education, Pfeiffer, 1 edition

[64] Deterding S., Khaked R., Nacke L.E. & Dixon D.. (2011a). Gamification: Toward a Definition. Chi 2011 Gamification Workshop Proceedings, pages 12-15

[65] Fitz-Walter Z., Tjondronegoro D., & Wyeth P.: Orientation Passport: using Gamification to Engage University Students. OZCHI 2011: 122-125.

[66] Connolly T. C., Boyle E. A., Hainey T., McArthur E., & J. M. Boyle, (2012). A Systematic Literature Review of Empirical Evidence on Computer Games and Serious Games. Computers & Education, 59, 661e686.

[67] Barata G., Gama S., Jorge J. A., & Gonçalves D. (2013a): Improving Participation and Learning with Gamification. Gamification 2013: 10-17.

[68] Rincon-Flores E.G., Gallardo K., & Fuente J.M. de la . (2018). Strengthening An Educational Innovation Strategy: Processes to Improve Gamification in Calculus Course Through Performance Assessment and Meta-Evaluation. IEJME-Mathematics Education 2018, Vol 13, No. 1, 1-11

[69] Souza M.R. de A., Veado L., Moreira R. T., Figueiredo E. & Costa H. (2018). A Systematic Mapping study on Game-Related Methods for Software Engineering Education. Information and Software Technology 95 (2018) 201-218

[70] Mourya Reddy Narasareddy Gari, Gursimran Walia and Alex David. Radermacher. Gamification in Computer Science Education – A Systematic Literature Review. 125th Annual ASEE Conference, 2018

[71] Peter Navarro and Judy Shoemaker. 1999. The Power of Cyberlearning: an Empirical Test. Journal of Computing in Higher Education. September 1999, Volume 11, Issues 1, pp 29-54

[72] Prabir Bhattacharya ; Minzhe Guo ; Lixin Tao ; Bin Wu ; Kai Qian ; E. Kent Palmer. 2011. A Cyber-Based Cyberlearning Environment for Introductory Computing Programming Education. 2011 IEEE 11th International Conference on Advanced Learning Technologies. Athens, GA, USA

[73] Deterding, S. (2011). Gamification by Design: Response to O'Reilly. Gamification Research Network. Retrieved from http://gamification-research.org/2011/09/gamification-by-designresponse-to- oreilly/

[74] Behnke, K.A. Gamification In Introductory Computer Science

[75] Brindley, J. E., Walti, C., & Blaschke, L. M. (2009). Creating Effective Collaborative Learning Groups in an Online Environment. International Review of Research in Open and Distance Learning, 10(3), 18.

[76] Drave W A (2000) Teaching online LERN Books, River Falls, Wisconsin.

[77] Hiltz, S. R. (1994). The Virtual Classroom: Learning Without Limits via Computer Networks. Norwood, NJ USA: Ablex Publishing Corporation.

[78] Bryant, J., Bates, A. J. (2015). Creating a Constructivist Online Instructional Environment. TechTrends, 59(2), 17-22.

[79] Hurst, B., Wallace, R. & Nixon, S.B., 2013, 'The Impact of Social Interaction on Student Learning', Reading Horizons 52(4), 375-398.

[80] McKinney D and Denton LF (2006). Developing Collaborative Skills Early in the CS Curriculum in a Laboratory Environment. SIGCSE 2006 Technical Symposium on Computer Science Education. Houston, Texas, USA

[81] Tinto, Vincent. Leaving College: Rethinking the Causes and Cures of Student Attrition. Second Edition, The University of Chicago Press, 1993.

[82] MR Narasareddygari, Walia, G., Borchert, O., and Radermacher, A. "Evaluating Learning Engagement Strategies in a Cyber Learning Environment during Introductory Computer Programming Courses-- An Empirical investigation" 125th Annual ASEE Conference, June 24 - 27, 2018, Salt Lake City, Utah

[83] MR Narasareddygari, Walia, G., and Radermacher, A. "Using Gamification and Cyber Learning Environment to Improve Student's Learning in an Introductory Computer Programing Course —an empirical case study" 125th Annual ASEE Conference, June 24 - 27, 2018, Salt Lake City, Utah.

[84] Reddy, M.*, Walia, G., Duke, D., Ramasamy, V., Kiper, J., Davis, D., Allen, A., and Potvin, G. "Evaluating the Impact of Combination of Engagement Strategies in SEP-CyLE on Improve Student Learning of Programming Concepts", The 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019. February 27 - March 2, 2019, Minneapolis, MN, USA.

[85] B. A. Kitchenham and S. Charters. 2007. "Guidelines for performing Systematic Literature Reviews in Software Engineering." Technical Report EBSE 2007-001. Keele University and Durham University Joint Report

[86] Brown, T.*, Reddy, M.*, Singh, M., Walia, G. “Using Code Review to Improve Programming Skills of Students in an Introductory CS Course”, IEEE Conference on Frontiers in Education, FIE-2019. October 16-19, 2019, Cincinnati, OH, USA

APPENDIX. SYSTEMATIC LITERATURE REVIEW RESULTS

A.1. Results of the Research Questions

In this section the researcher present the result of the systematic literature review and Co-Citation Analysis. This Sections describes the result for research questions RQ1-RQ4, respectively.

RQ1: *What are most commonly reported gamification elements in STEM education?*

This section discusses the results of the first research question. Gamification is defined as a technique that reuses game elements in a non-game context, which helps promote better user experience by improving students' motivation and engagement. This SLR focuses on the list of GEs (gamification elements) that have been used in different disciplines of STEM education. Based on our literature review, the researcher identified twenty-two commonly used GEs in the educational contexts. The researcher have provided brief descriptions of each GE below (Figure 4). We counted the number of times (frequency) each of the above GEs appeared in the primary studies. Figure 5 also shows the number of times each of the different GEs was identified in the primary study. Based on the results, leaderboard (appeared in 19 studies) followed closely by badges (appeared in 18 studies) and points (appeared in 17 studies), have been the most widely recognized (and reported) game element in STEM education. Level was reported as the fourth commonly used GE. The remaining 18 GEs had much fewer papers reporting the evidence on their impact of student learning.

A Wordle https://www.jasondavies.com word-cloud was created utilizing the names of the gamification elements that appears in each primary study. The visual word-cloud gives a descriptive overview of the most active gamification element in the literature, see Figure A.1. The word with the largest font is the more common word that appear in each study. In the case of

Figure A.1, the word cloud clearly shows that leaderboard is the most commonly used GE in the studies.

Figure A.1: Visual Word Cloud of the GEs for the Primary Studies

RQ2: *What disciplines in STEM use gamification?*

The researcher identified the subject area of each primary study, Figure A.8 (Empirical Evidence of Usefulness of GEs in STEM Education), and mapped them to the STEM disciplines, Figure A.2. The results shows that Formal Sciences (Computer Science, Mathematics, Software Engineering) is the field with the greater number of primary studies with GEs supporting it (53.3%), followed by Applied Sciences (Engineering, Technology, Medical) (40.0%) The least supported discipline is Natural Sciences (Biology, Biotechnology) (6.7%). Meanwhile, there was no recording of GEs support in Humanities and Social Sciences field of study. Figure A.3 shows the number of publications by course. Within the STEM field computer science leads the bar chart with ten publications having empirical evidence on gamification.

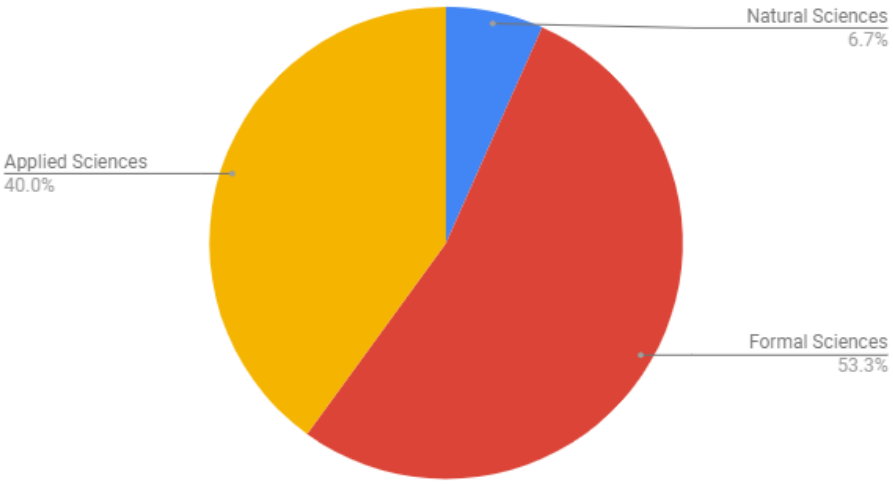


Figure A.2: Number of Publication by Discipline

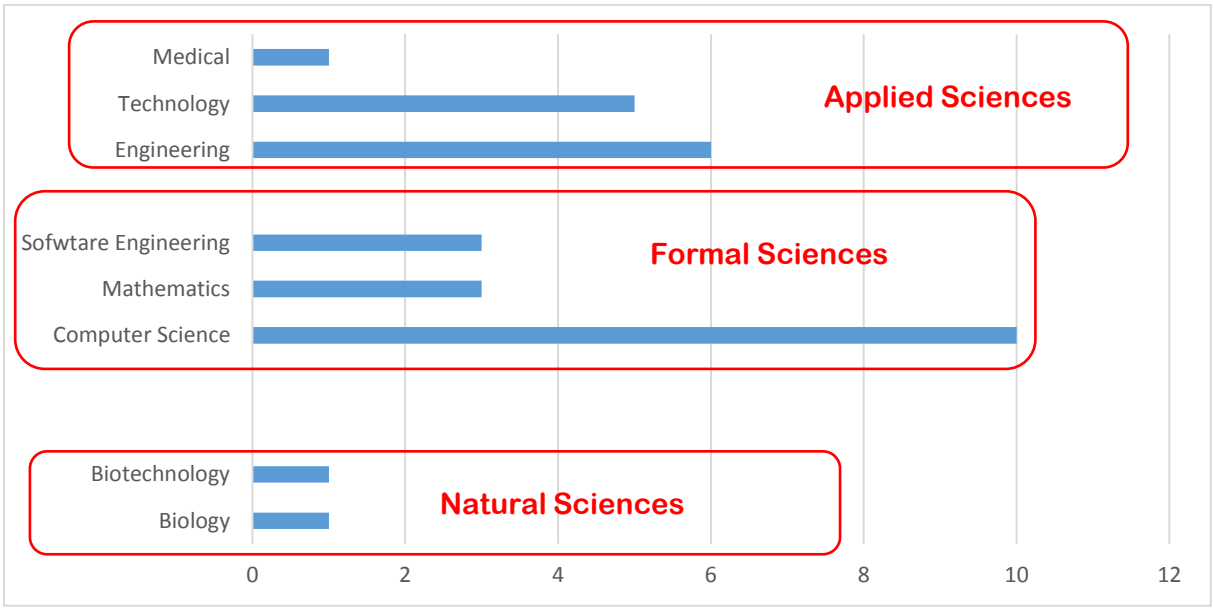


Figure A.3: Number of Publication by Course

RQ3: *What is the evidence of impact of those game elements on student learning and student engagement?*

The effects on gamification in education suggest that it is a promising technique that could be introduced into learning material. Most of the studies have shown that gamification contributes

to improvement in students' motivation, performance and their participation. Listed below is the frequency of the outcome of gamification on students.

Motivation ---------- Frequency of 15

Performance ---------- Frequency of 10

Participation ---------- Frequency of 9

Learning and engagement ---------- Frequency of 7

Competence, immersion and satisfaction ---------- Frequency of 1

- Motivation: The largest positive impact of gamification elements were found on motivation (appeared in 15 primary studies). The primary studies measured motivation in terms of students' willingness to complete the task because of incentives (e.g., points). LB, B, L, P all showed a positive impact on student motivation either when used individually or when used in conjunction.
- Performance: Multiple primary studies (10) reported positive improvement in student grades (a measure of performance) because of gamification. LB and P had the most impact on student performance.
- Participation: Gamification elements (especially Leaderboard and Challenges) showed a positive impact on Students active involvement in completing the task.
- Learning and Engagement: Learning (the ability of students to acquire knowledge) and engagement (students showing commitment to completing the task) were each reported by seven primary studies to have been impacted by gamification elements.
- Competence, Immersion, and Satisfaction: Competence (completing a task correctly), Immersion (Deep but effortless involvement in a task), and Satisfaction (enjoyment when

completing gamified task) only appeared in a single primary study and was limited in terms of empirical evidence.

Follow is the definition for the top eight study outcome.

- Motivation - Students were willing to complete task as a result of the incentives (GEs)
- Performance - Improvement in students' grades
- Participation - Students active involvement in completing task
- Learning - The ability of students to acquire knowledge
- Engagement - Students showing commitment to completing task
- Competence - Student completing a task correctly
- Immersion - Deep but effortless involvement in a task
- Satisfaction - Students enjoy completing gamified task

A Wordle https://www.jasondavies.com word-cloud was created utilizing the impact of gamification elements that appears in each primary studies (study result). In the visualization, common English words (e.g. the, of) have been removed. Similar words have been grouped also (e.g., motivation, motivating). The visual word-cloud word frequency gives a descriptive overview of the empirical evidence of the usefulness of GEs in STEM education, see Figure. A 4. It is clear that the major impact of GEs on students are:

- Motivation
- Participation
- Performance
- Engagement
- Learning

Figure A.4: Visual Word Cloud of the Usefulness of GEs in STEM Education

RQ4: *How can answers to RQ1 and RQ3 be incorporated into the design of cyber learning environments in CS/SE education?*

As previously mentioned, leaderboard, badges, points and levels are the top GEs that have been empirically evaluated in the literature. As SEP-CyLE already incorporates points and leaderboards in its system, badges and levels are the obvious choice for addition, as they have shown positive impact on students learning in most cases. While it is clear that certain game elements such as badges can help to motivate students, incorporation of any gamification element to SEP-CyLE would need to be empirically verified.

Though gamification has shown positive effect in most of the cases, the authors have also identified potential downsides. There were some studies where the students were not motivated and, in some cases it had a negative effect on them by reducing their intrinsic motivation [52]. This should be an important factor to take into consideration while designing the learning material with game elements in it. The researcher must make sure that there is not only extrinsic motivation, but also the intrinsic motivation on the part of the students, though motivating the students intrinsically is not an easy task. It is always important to track if a student is losing interest on the learning material or not. If they are losing interest, then the system should make some intervention to restore their interest on the topic.

A.2. Discussion of the Results

This section discusses put findings and insights about the use of gamification in stem education.

***RQ1**: What are most commonly reported gamification elements in STEM education?*

Overall, the study identified 31 relevant sources that were searched, as shown in Figure A.5. The continent contributing the most papers to gamification in STEM education is Europe followed by North America.

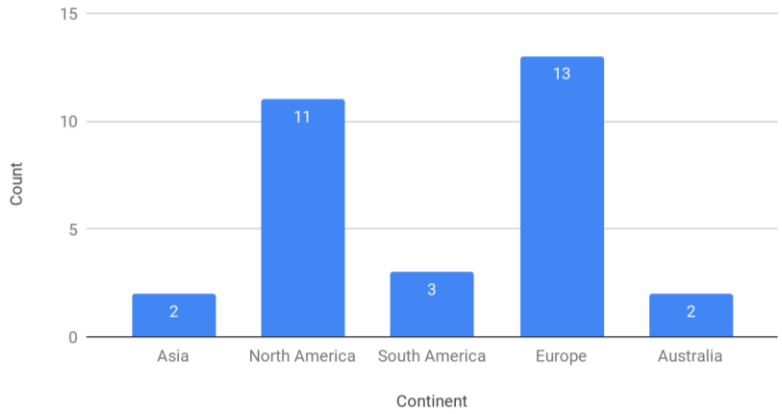


Figure A.5: Distribution of Primary Studies at the Continent Level

From the included 31 study, a total of 22 gamification elements commonly used in in STEM courses were identified. A combination of GEs contributed to the outcome of each primary study. This combination does not allow readers to know exactly which GE is associated with particular effects in students. The top four GEs used in the study incorporated in this SLR are leaderboard, badges, points, and levels. However, studies did not highlight which of the GE had more impact on student engagement and learning or in what percentage. Further study is required in order to isolate the impact of each GE on students learning.

RQ2*: What disciplines in STEM use gamification?*

The result of the study shows that number of publication by discipline was greater in the field of Formal Sciences specifically computer science. There is a minor presence of GEs in areas such as biology, chemistry and physics (that is, the Natural Sciences). These findings agree with previous results reported by Dicheva et al who also observed a tendency to apply gamification to the Formal Sciences (mainly Computer Science (CS) / Information Technology (IT). What does this mean? It means that researchers can learn from this study by implementing some of the GEs from the Formal Sciences to other fields [14]. The lack of instructors' ability to adopt to the digital era or maintaining digital instructional materials could be the reason for these findings [14]. Hence, more online educational tools are needed that involve other STEM areas in order to gather empirical data. For the top five courses (computer science, mathematics, software engineering, engineering and technology), the GEs that is mostly used are as follow:

- Computer science – points, leaderboard and badges all occurred within seven (7) times of the included publications.
- Mathematics – levels occurs three (3) times in the studies
- Software Engineering (SE) – Point was used two (2) times in SE studies
- Engineering – point was used in three (3) of the studies, levels in four (4) of the studies, badge in five (5) of the studies and leaderboard in three (3) of the studies
- Technology – leaderboard and levels tied at four (4), while points was used three (3) times in the studies.

RQ3: *What is the evidence of impact of game elements on student learning and student engagement?*

Gamification is mostly studied to look at the impact on student motivation; thus, also affecting student participation and academic performance. When analyzing the studies in detail most studies mainly focused on the effect of using GEs by students and a system. This might explain why motivation is a dominant dependent variable. The impact GEs have on learning and student engagement was reported positive in 14 of the studies. In addition, gamification was used as a device to motivate students in conforming to desired behaviors, such as being participative in the classroom, students arrived in class on time, reading of course materials [19, 3, 31]. Overall, it is questionable how researchers do not focus on intrinsic motivation such as psychological and behavioral variables. Even though there is more weight on positive outcomes, there are some negative effects of gamification on students. Barata showed that some students did not find the gamified learning activity engaging and enthusiasm for utilizing the platform diminished after two or three days [2]. Additionally, Auvinen reported that neither badges nor heat maps influenced the behavior of majority of the students [39]. Haaranen indicated that badges had an exceptionally negative impact on a student that "died internally" every time he saw the badges [18].

One of the paper revealed that gamification had constructive outcome on women [4]. Lehtonen et al found that the users of Finnish version utilized the framework for a more extended time and completed more activities than that of the English version users [33], which explains the impact of gamification on localization. This localization played an important part and affected the excitement of the user to use the learning environment.

RQ4*: How can answers to RQ1 and RQ3 be incorporated into the design of cyber learning environments in CS/SE education?*

The results from this review can assist the development (or re-design) and subsequent validation of SEP-CyLE. While SEP-CyLE already has the two of the most common gamification elements (Points and Leaderboard) incorporated, the researcher would like to add badges and progress bars into SEP-CyLE and measure how these gamification elements would affect the student learning. Based on the result of the study as summarized in Figure A.8 (Empirical Evidence of Usefulness of GEs in STEM Education), the researcher also propose the re-designing of SEP-CyLE to include more learning object for SE (Software Engineering) course and GEs that have proven to have a positive impact on students learning and engagement in SE. Empirical studies should follow to know which GEs is best suited for students' enrolled SE course. In addition, empirical studies may be done to assess the impact of GEs on motivation, performance and participation.

A.3. Conclusion

In this section, a systematic literature review study has been conducted to analyze the use of gamification in STEM courses. The scope of this SLR was gamification in STEM wide discipline, thus only education or serious games were considered. This study identify the most commonly used GE in different STEM courses. The researcher also evaluate the primary studies, seeking to analyze how the application of game mechanics in an educational context affect aspects such as user motivation, user engagement, user participation, etc. were impacted. The results show that integrating gamification elements into STEM education helps in achieving positive educational outcomes. Particularly, the addition of gamification elements into the online learning material has been shown to acts as a motivation factor for students to become more active in learning.

The results obtained during the analysis of the primary studies show that the existing literature on gamification is more utilized in computer science level one courses (CS 1). Most of the study examined look at programming within CS1 curriculum. This show up an important gap that must be address since other STEM course (example chemistry and physics) and other areas of computer science (such as software testing and software engineering) have not been studied to their full extent.

Another shortcoming identified was that most of the studies make use of the most common GE, which are; leaderboard, badge and points. This reflects that researchers do not focus on intrinsic motivation such as psychological and behavioral variables. The use of GE which are too simple can lead to some students not finding gamified learning activity engaging hence enthusiasm for utilizing the platform diminished after two or three days.

One other gap that the researcher were able to identify is the nonexistence of empirical study that shows the impact GE has on gender. We would like to see study carried out on this topic to see the difference between female and male performance as well as to highlight different ways in which gender are motivated. The researcher would like to also see ways in which gamification can be used to pursue STEM studies and retain women in the STEM field, particularly computer science.

Results are intended to be applied in the area of gamification in computer science / software engineering and will be used to improve the design and usability of an online learning environment by extending the range of STEM course supported in the online environment. While there are some studies underway, the researcher plan to report the results and conduct additional studies guided to build a larger body of evidence on usefulness of gamification in CS/SE education.

The researcher invite researchers to conduct more studies that focus on other STEM courses outside of computer science. The researcher foresee that within the next decade, there will be more empirical evidence in STEM to further support its application.