Active Learning with LEGO for Software Requirements

Stan Kurkovsky
Central Connecticut State University
New Britain, CT
kurkovsky@ccsu.edu

Stephanie Ludi
University of North Texas
Denton, TX
stephanie.ludi@unt.edu

Linda Clark
Central Connecticut State University
New Britain, CT
lclark@ccsu.edu

ABSTRACT

Case studies are one of the best active learning approaches for teaching software engineering in general and requirements engineering in particular. They offer an effective approach to understanding a system or a phenomenon that are too large or too difficult to represent in a lab setting. As a tangible manipulative, LEGO works well to support designing hands-on case studies that mix studying software engineering concepts with the elements of team building and playful creativity. In our project, we study the role that LEGO has in engaging students in software engineering through a set of activities that leverage the case study and play aspects. This paper presents our design approach for requirements engineering activities, as well as current results from classroom testing with a focus on student engagement with the subject matter.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → *Requirements analysis*;

KEYWORDS

Software requirements, software engineering, active learning, LEGO

ACM Reference Format:
Stan Kurkovsky, Stephanie Ludi, and Linda Clark. 2019. Active Learning with LEGO for Software Requirements. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287444

1 INTRODUCTION

Software engineering (SE) is often called a 'wicked problem' because the resulting software is full of complex interdependencies due to incomplete, contradictory, and volatile requirements [6]. Students often find it difficult to work on problems that do not have a single well-defined solution, where problem statements are ambiguous and vague, or when classroom examples have to be applied to different scenarios or domains [15]. For these reasons, it is sensible to rely less on lecture and focus more on active learning experiences mimicking what students may encounter in "real-world" SE.

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02…$15.00
https://doi.org/10.1145/3287324.3287444

As a discipline within SE, requirements engineering (RE) may be a particularly wicked problem [4, 15, 24, 27]. Wicked problems cannot be solved by applying a single standard approach or a single set of common procedures from a textbook. In order to learn how to solve RE problems, students need hands-on experience with practical scenarios, which could come from participating in actual software projects, simulations, role-playing, case studies, or other experiential learning activities [8, 13]. These activities provide a safe environment for students to learn from their own failures [28]. When it comes to learning activities in RE, it is especially important to focus on team-based exercises, which come closest to mimicking RE processes in real-word projects. Most importantly, building such experiences in a team setting creates an environment well-suited for open-ended problems offering a wide variety of possible scenarios and solutions informed by a diverse range of individual experiences, opinions, and emotions [24].

Case studies are one of the best active learning tools for teaching SE, offering an effective approach to understanding a system or a phenomenon too large or too difficult to represent in a lab setting. Whether case studies are text-based or hands-on, they help answer detailed *how* and *why* questions, which offers a deeper insight into chains of cause and effect concerning the system's behavior [25].

As a tangible manipulative, LEGO works well to support designing hands-on case studies that mix learning SE concepts with the elements of team building and playful creativity. While LEGO is associated with constructing concrete structures such as buildings, cars or even robots, the dynamic nature of LEGO also allows it to represent more abstract concepts [9]. For example, LEGO pieces in isolation or when assembled into small builds can represent lines of code, subsystems or parts of a software development process [14].

Play in a classroom environment offers many important features to aid in the learning process. Play provides a safe environment where failure is acceptable because "it's just a game." Play is voluntary and nobody is forced or coerced to participate. Play is fun and it is something that people want to do. Play is improvisational and provides an avenue to express creativity and imagination, which is especially important in design-centric computer science courses.

In our project, we study LEGO's role in engaging SE students through a set of activities that leverage the described case study and play aspects. Activity and resource development is part of this project[1], whereby faculty teaching SE or related courses can use the activities in isolation or as a set to engage students in the material. SE concepts represented include architectural design, change management, dependability, and object-oriented interfaces. This work describes our design approach for RE activities, as well as current results from classroom testing with a focus on student engagement with the subject matter.

[1] http://web.ccsu.edu/lego-se/

2 REQUIREMENTS ENGINEERING IN THE CLASSROOM

The importance of RE in software engineering practice, research, and education has long been recognized [16, 19, 20]. However, according to a recent survey of software RE education literature covering 79 papers published prior to 2015 [20], most students are not only unaware of modern RE practices, but are also disinterested in studying them [5, 12, 26]. This could be due to the prevailing student interest towards technical aspects of SE, as opposed to its social dimension exemplified in the RE discipline. This trend is extremely worrisome because as a discipline, RE encompasses understanding the dependencies among the stakeholders, processes, and systems in order to gather, analyze, prioritize, document, review, and manage the requirements for a software system throughout its complete lifecycle [29].

RE is a highly creative process. Whether considering requirements for an existing, new, or a future system, RE process requires a significant degree of insight, experience, and creativity [17]. Creativity can be defined as "the interplay between ability and process by which an individual or group produces an outcome or product that is both novel and useful as defined within some social context".

Many SE novices, most students included, do not take RE seriously enough because it does not immediately result in program code [7, 12]. A traditional approach to computing education essentially encourages this kind of neglectful approach to software requirements. Computer science and SE students begin by acquiring a lot of technical knowledge, which almost always starts with learning how to program. Even in the assignments given in later courses, students are often rewarded mostly for the correctness of implementation. In fact, RE may seem entirely unnecessary because many assignments and projects given to students may already include all requirements spelled out for them. Consequently, many students perceive RE activities as not particularly important because they are almost always less technical than other SE activities such as design, implementation, and testing. Consequently, the importance of software requirements and the RE process may not start to emerge until students begin facing complex problems.

We believe that RE should be studied in a constructive setting that promotes practical application of theoretical knowledge. An active learning environment that prominently features group work with plenty of peer learning opportunities may be a suitable setting to achieve that goal. Ideally, we would like students to test the textbook ideas in a hands-on setting against the backdrop of their practical experiences accumulated thus far. In this approach, students will be able to build new mental models and develop new concepts and attitudes based on their previous knowledge and experience. This approach underscores the fact that although knowledge construction is personal in nature, knowledge sharing is inherently a collaborative and social endeavor, which is achieved through reflection on decisions and actions in order to internalize the newly acquired knowledge [3]. In-class group activities based on the preceding lecture material is a key mechanism to solidify the corresponding course material through peer learning. Ensuing discussions or role-playing can be helpful for students to apply new material in a dynamic way, to challenge themselves and peers, and provide constructive criticism to each other. Well-designed group exercises can allow all students to participate in a safe and non-intimidating way by relating the main topic of the exercise to a well-known or an easy to understand system or context.

Current research indicates that using games in education improves student engagement, helps reinforce conceptual elements through their practical application, leading to a better retention of the course material. Educational games help promote friendly competition, as well as student emotional engagement with the course content [29]. Emotions generated by playing an interesting game can help students create conceptual models providing internalized meaning to the new concepts [2]. Additionally, people tend to be engaged by the novelty of experiences, making them likely to retain information associated with a delivery method that breaks the monotony of a traditional lecture.

A number of reports on using games to teach SE indicate that successful strategies include using a variety of activities that combine practice of new concepts with recall of previous material. Game-based activities are especially well-suited for reinforcing knowledge learned earlier and when it is combined with other instructional approaches [28]. Team-based setting is optimal because it supports collaboration, encourages everyone's participation, reduces the risk of embarrassment, and provides plenty of opportunities for peer learning [2, 12, 28]. Any such activity must culminate in an instructor-led debrief, during which students are typically asked to comment on the ways that the activity reinforces the concept, reflect on their own experiences, and hear the others' thoughts [26, 28]. Open-ended questions help students strengthen the connections between the activity and the course material, ensuring a deeper understanding and better retention of the material [18].

According to several recent surveys describing various efforts to use games and other active learning exercises for SE education and training [21, 28], RE appears to be the most neglected area with the smallest number of documented experience reports. At the same time, it is clear that graduates entering the software industry must possess a strong set of both technical and soft skills, while RE exemplifies an intersection of these skill sets. Many researchers and practitioners argue that strengthening the skills of SE professionals in the area of RE is a key success factor in most SE projects [10, 20].

SE educators and practitioners must not forget that SE's social dimension is integral to the discipline, but is frequently overshadowed by technical aspects of practice [1, 4, 27]. Of all areas of knowledge and practice comprising SE, RE is perhaps one area that impacts this social dimension the most. Although interpersonal communication and teamwork are key enabling factors in all software process activities, they usually involve interactions on technical topics with like-minded technically-inclined individuals. RE, however, calls for an effective communication with a broad range of people who have different backgrounds and goals and who may not always have a clear idea about their wants and needs. RE addresses social dimension issues by emphasizing two aspects of communication: documenting and understanding. On the one hand, RE focuses on documenting the customer needs and desires, and expressing them in the form understandable by software engineers. This implies that customers know what they want, and that information can be expressed using some mutually acceptable formalisms. On the other hand, any documentation produced as a result of RE activities aims to support and improve software engineers' understanding of

the customers' needs. Furthermore, RE activities should also aim to help customers crystallize their understanding of their own needs. The process that leads to developing this mutual understanding is one of the main reasons why social dimension of RE is so important. Modern software processes, such as agile methodology, de-emphasize documentation in favor of close involvement of and frequent interactions with the customer. This focus on understanding underscores the need for instilling social competencies into the modern software professionals' skill set.

3 REQUIREMENTS ENGINEERING AND LEGO

This paper focuses on RE, whose activities transcend the process model used (e.g. waterfall, agile), and include requirements elicitation, analysis, documentation, and validation. Conceptually, RE activities can be challenging for students as many introductory SE courses provide students with the project requirements. Thus, students can feel as if the activities are partially completed for them. The other issue that can arise is that activities involving interaction with users/customers makes some students uncomfortable as such interaction is so different from their typical project work.

This section discusses RE modules, as well as observations in conducting the corresponding LEGO-based activities in class. While this set of modules focuses on RE, each offers a different perspective, as well as focus on specific concepts within requirements. An instructor would not use them all at once, but instead weave them into appropriate parts of the course, especially if agile or incremental process models are used as part of the course.

Each module is outlined in a consistent manner and includes an overview, learning outcomes, materials & supplies, an outline for conducting the activity, as well as instructor notes/observations.

3.1 Module 1: Requirements Elicitation and System Refactoring in an Agile Environment

This module offers a concise experience on the importance of requirements elicitation via stakeholder interaction, while illustrating the role of reflection and refactoring in agile software process.[2] In this module, students work on small teams to build a system prototype, manifesting as a house. The stakeholder is the instructor, who has a set of requirements in mind that the house should possess. Students are instructed to devise effective questions for the stakeholder, whereby the responses are used to prototype the system based on the derived features. The activity can be conducted as 2 or 3 sprints. At the end of each sprint, each team presents their system to the class and the stakeholder offers feedback on requirements coverage. At the beginning of the second and third sprint, the stakeholder changes a requirement that would require refactoring of the prototype.

The learning objectives associated with Module 1 are:

- Devise a set of appropriate elicitation questions to gather requirements from a stakeholder;
- Generate a set of requirements based on stakeholder interaction;
- Describe the importance of reflecting on team effectiveness and the role of refactoring in an agile environment;
- Validate a set of requirements for completeness and accuracy.

The module is designed for a class size of 20 and upwards of 70, where the students work in teams of 5. Some flexibility in terms of duration also is designed into the module, where the minimum time is 50 minutes (2 sprints) to a maximum 70 minutes for 3 sprints.

Each team has its own LEGO kit, either based on the LEGO Serious Play kit or a compendium of classic LEGO bricks, windows, doors, trees/flowers, and some minifigures. The teams can work around their respective tables, where each team should have a LEGO baseplate to build their prototype. Additionally, sticky notes and pens or access to a whiteboard is useful for sharing the requirements that their prototype represents.

To help scope the prototype, the following initial requirements are provided to the instructor: house shall have 3 rooms, where each room has a window and a doorway; house shall be single story, with a single (front) door; doors and doorways shall be sized to fit a minifigure; a single outer wall shall be a solid color; the house shall have landscaping. Note: this is intentionally vague as these parts may not be consistent across kits.

The module is conducted in 2 sprints (50 minutes) or 3 sprints (70 minutes). The 2-sprint module is presented here, the 3rd sprint is a repetition of sprint 2 with additional reflection. The first sprint focuses on requirements elicitation and delivering an initial prototype. In the 30-minute sprint, students prototype a system (a house). Students, working in small teams, are instructed to devise effective questions for the stakeholder in the simulation of elicitation, where the responses are used to prototype the system based on the derived features. Each team must decide what 3-4 questions will be the most effective to gather requirements. Teams write their questions on post-its or on their whiteboard. The team raises their hands when they need to interact with the stakeholder. The stakeholder needs to respond to the question as it was written, i.e. if a team wrote a question as a yes/no question then the response needs to be yes/no.

The student teams then build their prototype based on the stakeholder responses. The teams document the requirements they elicited on sticky notes. At the end of the sprint, each team briefly explains their prototype in terms of what requirements it represents. Each team places their sticky notes on the board for everyone in the class to view. Next, the stakeholder shares the initial requirements that they wanted (setting aside these sticky notes or circling them to call them out), and asks teams to reflect on how effective their questions were at eliciting these requirements.

During Sprint 2, the teams focus on refactoring based on a new requirement given by the stakeholder. Suggested requirements are: a loft needs to be added to the house; the solid color wall needs to be striped with 2 alternating colors; the roof needs a skylight.

For the 20-minute sprint, the teams have time to refactor their system to meet the new requirement. They also need to meet any prior requirements that were missed. At the end of the sprint, each team briefly explains their prototype in terms of what feature it represents and how the prototype was refactored to meet the revised requirements. Also, how did the team work to refactor the prototype efficiently? What could have been done to be more efficient? The team and stakeholder do a final check of the requirements

[2] http://web.ccsu.edu/lego-se/modules/RequirementsValidation.html

for coverage and accuracy. Students discuss any ambiguities or accuracy issues.

If a third sprint is conducted, students are able to reflect further on how their refactoring strategies evolved, as well as any requirements coverage/accuracy issues. The design choice to designate an optional third sprint is to provide flexibility since some classes may meet for 50 minutes while others may meet for 80 minutes.

The instructor notes provide some tips for the instructor, based on initial module testing and general observations by the authors:

- In a large class, keeping track of who asked what questions or what the instructor responded is required. The instructor can do so via a notepad, their tablet, or some other mechanism.
- In a class that meets for 50-minutes, time management is particularly needed though having a mechanism such as a timer or clock will help with keeping track of the time regardless of the class meeting duration.

3.2 Module 2: Requirements Elicitation and Analysis

In this module, students work as a team to prototype a system competing against a well-known social media system (e.g. Facebook, Twitter, Instagram, SoundCloud) or a well-defined hardware system (e.g. Roomba vacuum, Nest learning thermostat, or Amazon Echo).[3] Students brainstorm and identify a set of features that will implement the core functionality and help the new system distinguish itself from the competition.

The learning objectives of this module include:

- Synthesize a core set of system requirements;
- Generalize a set of requirements by identifying related and overlapping requirements;
- Distinguish between functional, non-functional, and domain requirements;
- Validate requirements for completeness and consistency.

The prerequisites of this module include basic familiarity with software requirements, as well as prior exposure to the principles of LEGO Serious Play [11, 14]. This activity is about 60 minutes long and works well for a single team of up to 25 participants. It has been offered successfully with team sizes ranging from 5 to 25.

Materials and supplies necessary to run this activity include a LEGO set for each student, such as a LEGO Serious Play 'exploration bag' or any cheap ($5) LEGO Classic, LEGO City, LEGO Friends, or a similar set. Alternatively, all students can share one large LEGO set with a diverse combination of bricks. A set of minifigures with accessories may also be useful. Some sticky notes and markers will also be needed. Each student requires a separate small building space with one additional table to place all models.

The module consists of two parts: requirements elicitation (using LEGO Serious Play for ideation) and analysis (not using LEGO).

Requirements elicitation begins with explaining the problem to the students: we are building a competitor to a well-known social media system or a well-defined hardware system. We need to brainstorm and come up with some features that will implement the core functionality and help us distinguish ourselves from the competition. It is important to emphasize the word 'features,' not requirements or functionality. It is important to use a well-known system as a reference, so that students would have enough general knowledge about it while providing enough room for interpretation and extension. The instructor asks an open-ended question: what features would you envision in the system? In response to that, each student builds a LEGO model representing a single feature of the system. Then, following the principles of LEGO Serious Play, students take turns to briefly explain the ideas behind the LEGO models they built and the system features they represent. Everyone takes turns to speak keeping the stories related to their LEGO models. The learning point is that many students may come up with similar ideas, but everyone has a different perspective. It is important to consider everyone's input to obtain a holistic unbiased picture. In requirements elicitation, all stakeholders have an opportunity to contribute their input to the list of requirements. Right after presenting their features of the system, each student writes down a few words on a sticky note describing their main idea. All sticky notes are collected and placed on the board where everyone can see them. This allows the students to make their ideas more concrete and formulate them in a concise form. This ideation process may be repeated, because a single round may not be sufficient to generate all important ideas. Only after seeing the results of one round on the board, students may start to come up with really interesting features.

Once the ideas for the proposed system features are in place, the elicited requirements can be analyzed. The instructor can ask the class to suggest some logical groupings of the requirements. It is possible that some requirements may be similar or overlapping, while clusters of requirements could help us identify and focus on the core functionality of the system. We should also reject those requirements that don't make sense. A learning point here is that it takes a collective effort to see the 'big picture' in order to identify what is important/relevant and what is not. At this point, the requirements can be divided into three categories: functional, non-functional, and domain requirements. Following that, the requirements can be checked for completeness. The instructor could prompt students with these questions: what are the most important requirements, did we miss anything important, are there any assumptions that must be identified as separate requirements? The resulting set of requirements should also be checked for consistency: are there any ambiguities or contradictions?

Running the activities of this module with a group of students, it is important to keep in mind a few practical points:

- It's OK to propose similar requirements during the first phase. In fact, it is very helpful to have such similar/duplicating requirements to make the learning points stronger.
- It is very important to keep a close eye on the clock during the second part of the exercise to ensure that there's enough time for every part of this activity. As with all such activities, the key to achieving the learning points is in the discussions, all of which are concentrated in the second part.

3.3 Discussion

Two of the three co-authors work at two different institutions. One of the authors is at Central Connecticut State University, a medium-sized public university with classes consisting of 25-30 students.

[3] http://web.ccsu.edu/lego-se/modules/RequirementsElicitation.html

Another co-author works at University of North Texas, a large public university with class sizes ranging between 45-70 students.

Although both learning modules described here are conducted in a team setting, the purpose is to engage each student with the material and to reflect on the concepts/process. As such, the stopping points where teams are asked to reflect are critical. However, it is important to note that each student on the team must contribute to the activity, reflect of what was learned, and share with the class as a whole. Without this, some students may not be engaged and feel that the activity is a diversion. To mitigate this risk, the use of the appropriate terms as they are being introduced/used in class helps solidify that connection.

These modules can be scaled up/down to fit various class sizes. For one of the authors, SE class sizes are typically between 45 and 67 students. As such, conducting modules in teams is helpful in managing resources and time. Even so, piloting module 1 has provided the author with some observations to help in devising tips for replication elsewhere with similar class sizes, as well as feedback for future development. For example, module 1 has been run successfully with larger class sizes, though the challenge is having enough time to facilitate discussion with large numbers of teams (8 or more). Module 2 has been successfully run in various configurations: one large team of up to 20 students, or two or more smaller teams of 8-10 students working in parallel. For larger classes, a TA can serve as an additional. Teams of 4-5 students works well for large classes as it allows each person to participate, while balancing the time needed by the instructor to interact with the teams.

4 EVALUATION

This study took place at a regional master's public institution located in the Northeastern US. During the Fall 2017 semester, participating students enrolled in a senior SE course required for all undergraduate computer science students. All 21 of the students enrolled in the course consented to participate in the study.

4.1 Materials

The Motivated Strategies for Learning Questionnaire (MSLQ) [22] is a 44-items questionnaire with a 5-point Likert scale response ranging from "strongly disagree" to "strongly agree." The MSLQ measures student motivational orientation, cognitive engagement, and self-regulation in the classroom. The cognitive variables and self-regulation are hallmarks of students who choose to be more cognitively engaged [22]. The MSLQ has five subscales: test anxiety, self-regulation, self-efficacy, cognitive strategy use, and intrinsic value. The MSLQ lends itself to subscale comparisons or global composite scores. Empirical validation of the subscales includes significant relationships with prior academic performance [22] and later academic performance [23]. Open-ended questions listed in the Appendix included constructs relevant to the process of building the model, abstract thinking, and the use of LEGO models as representations of abstract ideas.

4.2 Procedures

During the second week of class, the course instructor invited the researcher to meet with the students during a class period. Students were informed of the study and consent to participate was obtained.

Table 1: Descriptive Statistics for the Pre-Module Continuous Variables

	Control M (SD)	LEGO M (SD)	Total M (SD)
Self-Efficacy	3.66 (.72)	3.65 (.62)	3.66 (.65)
Intrinsic Value	4.36 (.37)	4.27 (.40)	4.31 (.38)
Cognitive Strategy Use	3.55 (.27)	3.53 (.43)	3.54 (.35)
Self-Regulation	3.37 (.32)	3.34 (.37)	3.36 (.34)
Test Anxiety	2.77 (1.05)	2.73 (1.08)	2.75 (1.04)
GPA	3.13 (.38)	3.13 (.42)	3.13 (.39)

Table 2: Descriptive Statistics for Post-Module MSLQ Scores

	Control M (SD)	LEGO M (SD)	Total M (SD)
Self-Efficacy	3.47 (.60)	3.80 (.61)	3.64 (.61)
Intrinsic Value	3.84 (.43)	4.22 (.42)	4.04 (.46)
Cognitive Strategy Use	3.56 (.36)	3.60 (.47)	3.58 (.41)
Self-Regulation	3.31 (.68)	3.42 (.42)	3.37 (.55)
Test Anxiety	3.23 (.85)	2.82 (1.11)	3.02 (.99)
Total Score	3.52 (.34)	3.66 (.33)	3.62 (.34)

During this class, a paper and pencil version of the MSLQ was administered.

Students were matched on the MSLQ subscale scores, MSLQ overall score, and cumulative grade point averages. One student from each pair was randomly assigned to one of two groups, LEGO or control. With uneven numbers, the remaining student was assigned to the LEGO group. During the module, the two groups met in two separate rooms. One room included the LEGO group and the primary course instructor; the control group met in a separate room with a second instructor. Each instructor followed identical module scripts, with the exception of working with LEGO versus working through the example on the whiteboard.

The following class meeting, the LEGO group met with the researcher and responded to a series of open ended questions in a focus group format. At the same time, the control group had a debrief with the control group instructor.

Four weeks following the module, a paper and pencil mid-term exam was administered to the entire class. Two items embedded in the exam measured the ability of students to identify system requirements, resolve ambiguities, and express them in the form of user stories. The mid-term items were blindly graded by the course instructor using a scale of 0 to 10, with 10 being a perfect score.

At the end of the semester, a second paper and pencil administration of the MSLQ was given to both groups by the researcher.

4.3 Results

4.3.1 Quantitative Data. Of the 21 students, 3 were female and 18 were male. The control group of 10 students included one female, and the LEGO group 2 females. Because the groups were matched on the MSLQ and GPA values, the two groups did not differ significantly on any of these measures, as shown in Table 1.

Both groups share the same trend among the MSLQ subscales on the initial administration, for all subscales. Overall intrinsic value is the highest construct (M=4.31, SD=.38), followed by self-efficacy (M=3.66, SD=.65). Cognitive strategy use (M=3.54, SD=.35) and self-regulation (M=3.36, SD=.34) were rated lower, however means for both subscales fell into the "agree" Likert category. The lowest subscale was test-anxiety (M=2.75, SD=1.04), both groups tending not to rate themselves as anxious.

These similarities were maintained after the intervention, yielding non-significant differences both between the pre- and post intervention, as well as between the two groups using bootstrapped confidence intervals and a significance level of .05 (Table 2). Specifically, intrinsic value for the LEGO group (M=4.22, SD=42) and control group (M=3.84, SD=.43) were again the highest values. The non-significant differences for LEGO group members for self-efficacy (M=3.80, SD=.61), cognitive strategy use (M=3.60, SD=.47), and self-regulation (M=3.42, SD=.42) were similar to control group means (M=3.47, SD=.60), (M=3.56, SD=.36), and (M=3.31, SD=.68) respectively. While overall test anxiety rose slightly after the intervention (M=3.02, SD=.99), LEGO group mean scores of (M=2.82, SD=1.11) were similar to control scores of M=3.23, SD=.85.

As a measure of academic performance, the two mid-term items reflected the module content. Both groups of students exhibited similar non-statistically significant strong performances on these items, with means above 9.0 for both groups on both items, no student receiving less than a score of 7 on either item.

4.3.2 Qualitative Data. Of more interest were the open-ended comments from participants in the LEGO group. Overall, the students expressed positive comments about the activity. Students liked the interpersonal interaction, *"discussing in small group, learn from each other, better than working individually."* While this the activity was challenging, it was also rewarding, *"trying to represent that idea with LEGO was initially confusing but doing so helped me visualize and further develop that idea."* Several reported the impact that working with LEGO had the problem-solving process. *"LEGO did help to an extent to see the software development process from another perspective besides UML modeling or staring at a terminal."* One student went so far to describe the thought process used while engaging in the activity: *"By thinking about how to build the model to properly show my feature, I thought of which features of the model would contradict my feature and cause issues with the model, and then I was able to abstractly consider why a model would fail from those features in order to apply it to the context of the social media site, which helped me to explain what I meant."*

Finally, the students reported that using LEGO moved their thinking beyond concrete conceptualizations to abstract. This higher-order thinking is a pedagogical challenge for most faculty, but from the student comments, LEGO may be tapping into this cognitive function *"by having LEGO to describe my example I had to go abstract and think about what is the fundamental idea behind the design pattern rather than thinking about just the example."*

Overall themes from the qualitative remarks touch on important pedagogical constructs in engaging post-secondary students in STEM education, active learning, problem solving, and abstract thinking. Within the student development process, moving students to interactive classroom activities can be met with resistance. The use of LEGO created an environment in which students interacted with peers and the instructor. The students reported an increased ability to problem solve, a key component particularly within STEM applied skills. These results are consistent with creating a learner-centered environment to maximize instruction of undergraduate STEM majors.

4.3.3 Threats to Validity. While the administration of the module is a pilot, a one-time experience with LEGO may not be sufficient to elicit changes in cognitive motivation as measured by the MSLQ. Additionally, the student comments along with the literature related to play suggest that abstract thinking as opposed to motivation may be a more appropriate measure of the impact of LEGO on cognitive function. A final consideration for the validity of the results reflects the study design. Because the two groups were from the same class, the students may have discussed the experience between groups, hence diluting the uniqueness of the groups. These factors likely contributed to the lack of significant findings within the quantitative results. It may be possible that working with LEGO does not impact motivation, but instead, the ability to engage in abstract thinking. Hence, along with the limitations of the study design, the construct of interest may have been incorrectly identified.

5 SUMMARY

This paper outlined our ongoing work to develop active learning modules that reinforce a broad range of SE concepts. RE is an integral component of the SE discipline, which, unfortunately, often gets overlooked by many students, educators, and practitioners. Here, we detailed two active learning modules focusing on RE. Both of these modules have been tested a number of times with different groups of students and at different institutions. Our evaluation of the impact on student learning shows that students are enthusiastic about using LEGO-based learning modules in the classroom, while their feedback indicates a number of tangible benefits helping students achieve their learning goals.

We aim to expand the assessment to both institutions in order to study the effectiveness of the activities on different groups of students, including diverse student samples in large classroom settings (60+ students per section). At the University of North Texas, the SE course will soon become a 3rd-year course; the impact on student learning under the new course constraints will also be of interest. Our goal is to provide activities that can be used in different instructional contexts, so the evaluation will provide evidence to support the aim.

As assessment of these modules continues, the researchers are currently investigating the impact of the LEGO modules on abstract thinking. In order to ensure validity of the intervention, the LEGO activates will occur in two separate classes and include three modules over the course of an academic semester. It is anticipated these adjustments in the assessment methods will enhance the validity of the interventions and result in gaining more knowledge about the cognitive changes brought about by the LEGO play.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation IUSE program awards 1611905 and 1709244.

REFERENCES

[1] B. Al-Ani and N. Yusop. 2004. Role-playing, group work and other ambitious teaching methods in a large requirements engineering course. In *Proceedings. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004.* 299–306.

[2] J. Beatty and M. Alexander. 2008. Games-Based Requirements Engineering Training: An Initial Experience Report. In *2008 16th IEEE International Requirements Engineering Conference.* 211–216.

[3] Mordechai Ben-Ari. 1998. Constructivism in Computer Science Education. *SIGCSE Bull.* 30, 1 (March 1998), 257–261.

[4] J. A. Bubenko. 1995. Challenges in requirements engineering. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on.* 160–162.

[5] David Callele and Dwight Makaroff. 2006. Teaching Requirements Engineering to an Unsuspecting Audience. *SIGCSE Bull.* 38, 1 (March 2006), 433–437.

[6] Peter DeGrace and Leslie Hulet Stahl. 1990. *Wicked Problems, Righteous Solutions.* Yourdon Press, Upper Saddle River, NJ, USA.

[7] J. M. Fernandes, R. J. Machado, and S. B. Seidman. 2009. A Requirements Engineering and Management Training Course for Software Development Professionals. In *2009 22nd Conference on Software Engineering Education and Training.* 20–25.

[8] K. Garg and V. Varma. 2007. A Study of the Effectiveness of Case Study Approach in Software Engineering Education. In *20th Conference on Software Engineering Education Training (CSEET'07).* 309–316.

[9] David Gauntlett. 2014. The LEGO System as a tool for thinking, creativity, and changing the world. In *LEGO Studies: Examining the Building Blocks of a Transmedial Phenomenon*, Mark Wolf (Ed.). Routledge, New York.

[10] Hubert F. Hofmann and Franz Lehner. 2001. Requirements Engineering As a Success Factor in Software Projects. *IEEE Softw.* 18, 4 (July 2001), 58–66.

[11] Alison James. 2013. Lego Serious Play: a three-dimensional approach to learning development. *Journal of Learning Development in Higher Education* 9, 6 (2013).

[12] E. Knauss, K. Schneider, and K. Stapel. 2008. A Game for Taking Requirements Engineering More Seriously. In *2008 Third International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games.* 22–26.

[13] D.A. Kolb. 1984. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice-Hall.

[14] Stan Kurkovsky. 2015. Teaching Software Engineering with LEGO Serious Play. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15).* ACM, New York, NY, USA, 213–218.

[15] Connolly Thomas M., Stansfield Mark, and Hainey Thomas. 2007. An application of games-based learning within software engineering. *British Journal of Educational Technology* 38, 3 (2007), 416–428.

[16] C. J. Neill and P. A. Laplante. 2003. Requirements engineering: the state of the practice. *IEEE Software* 20, 6 (Nov 2003), 40–45.

[17] Lemai Nguyen and Graeme Shanks. 2009. A Framework for Understanding Creativity in Requirements Engineering. *Inf. Softw. Technol.* 51, 3 (March 2009), 655–662.

[18] Scott Nicholson. 2015. *A RECIPE for Meaningful Gamification.* Springer International Publishing, Cham, 1–20.

[19] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements Engineering: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE '00).* ACM, New York, NY, USA, 35–46.

[20] Sofia Ouhbi, Ali Idri, José Luis Fernández-Alemán, and Ambrosio Toval. 2015. Requirements Engineering Education: A Systematic Mapping Study. *Requir. Eng.* 20, 2 (June 2015), 119–138.

[21] Oscar Pedreira, FÃĺlix GarcÃıa, Nieves Brisaboa, and Mario Piattini. 2015. Gamification in software engineering âĂŞ A systematic mapping. *Information and Software Technology* 57 (2015), 157 – 168.

[22] Paul R. Pintrich and Elisabeth V. De Groot. 1990. Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology* (1990), 33–40.

[23] Paul R. Pintrich, David A. F. Smith, Teresa Garcia, and Wilbert J. Mckeachie. 1993. Reliability and Predictive Validity of the Motivated Strategies for Learning Questionnaire (Mslq). *Educational and Psychological Measurement* 53, 3 (1993), 801–813.

[24] Gil Regev, Donald C. Gause, and Alain Wegmann. 2009. Experiential learning approach for requirements engineering education. *Requirements Engineering* 14, 4 (20 Jun 2009), 269.

[25] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case Study Research in Software Engineering: Guidelines and Examples* (1st ed.). Wiley Publishing.

[26] R. Smith and O. Gotel. 2008. Gameplay to Introduce and Reinforce Requirements Engineering Practices. In *2008 16th IEEE International Requirements Engineering Conference.* 95–104.

[27] Hans van Vliet. 2006. Reflections on Software Engineering Education. In *Software Engineering Education in the Modern Age*, Paola Inverardi and Mehdi Jazayeri (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–10.

[28] C. G. von Wangenheim and F. Shull. 2009. To Game or Not to Game? *IEEE Software* 26, 2 (March 2009), 92–94.

[29] D. Zowghi and S. Paryani. 2003. Teaching requirements engineering through role playing: lessons learnt. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* 233–241.

A APPENDIX: OPEN-ENDED QUESTIONS

Question 1: Comment about your thoughts when you were actually building your LEGO model. For example, were you thinking about how to represent the actual feature, the fit of the feature into the system, the function of the feature?

Question 2: You had to build a LEGO model representing a software feature you'd like to propose. Did LEGO help you crystallize your idea? If so, how? If not, why do you think it didn't help?

Question 3: Before you heard what your classmates said about their LEGO models, what were some of your impressions when you saw your classmates' LEGO models?