ISSN 2473-4901
v7 n5582

Teaching Case

The Building Blocks of Requirements: A LEGO©-Based Activity for Introducing Requirements Engineering and the System Development Lifecycle

Michael Scialdone
mscialdone@mays.tamu.edu
Information and Operations Management
Texas A&M University
College Station, TX 77843, USA

Amy J. Connolly
conno3aj@jmu.edu
Computer Information Systems & Business Analytics
James Madison University
Harrisonburg, VA 22807, USA

Abstract

We describe an original in-class exercise that was designed with LEGO© Serious Play™ to both introduce students to requirements engineering and, more broadly, begin acquiring hands-on understanding of the system (or software) development lifecycle. Our exercise emphasizes information transfer to create an experience which addresses why requirements engineering is critical to the development lifecycle and how easily it is prone to communication problems. Further, we discuss the evolution of the exercise, lessons learned, and potential refinements for future instantiations.

Keywords: Requirements engineering, active learning, systems analysis and design, project management, software engineering, serious play

1. INTRODUCTION

The Telephone Game is an activity, usually played in elementary school, where a message is given to someone, then whispered to the next person in a line or circle of individuals. As the message moves from one person to the next, it is usually misheard, misunderstood, and misinterpreted. When the message reaches the final individual, it has shifted in form, substance, and meaning, often rendering it indecipherable.

This game's main lesson is that the transfer of ideas is problematic depending on how we send and receive messages. Although one might be tempted to write off The Telephone Game as superficial or juvenile, its emphasis on how easy it is for messaging to go awry has real and practical implications far beyond the schoolyard playground. Indeed, effective communication, that is, the extent to which the transfer of ideas maintains a high degree of fidelity as it moves between individuals, is paramount to success in contexts where that success is predicated on cooperation and collaboration.

One such context is that of requirements engineering, a cornerstone of software and systems development. In fact, of the many challenges associated with this activity, "the lack of proper communication and knowledge transfer between software stakeholders is among the most important" (Ghanbari, Similä, & Markkula, 2015). Given that an accurate determination of requirements is critical to the success of any system (Davey and Parker, 2015), students learning about the System Development Life Cycle (SDLC) need to understand *why* requirements engineering is important and *how* it is prone to communication problems like those highlighted by the Telephone Game.

Regardless of the specific development methodology appropriated in practice (i.e., agile, waterfall, etc.), requirements engineering is a vital task because mistakes made at this point affect future work, and if not caught, result in higher costs to correct them (Chakraborty, Baowaly, Arefin, & Bahar, 2012). Therefore, students need to recognize the potential for errors rooted in communication between the individuals typically involved in this task. Furthermore, as the SDLC is inherently a collaborative effort that may include, but is not limited to, groups that include management, end-users, developers, engineers, UX designers, technical writers, and others; effective communication becomes a vital concern across the entire array of tasks in any instantiation of a lifecycle.

To this end, we designed an in-class, hands-on exercise that introduces requirements engineering to students learning about the SDLC, serving to stress the importance of effective communication. This exercise demonstrates that transferring ideas (namely, requirements) between individuals is no simple task, but rather, demands intentionality and even strategic planning. It teaches that *what* and *how* we communicate will ultimately impact the extent to which outcomes of the SDLC are successful.

As a note of clarification, we use the acronym SDLC in this paper to refer to either the Software Development Life Cycle or the System Development Life Cycle. As explained by Ruparelia (2010), "System development life cycle models have drawn heavily on software and so the two terms can be used interchangeably in terms of SDLC, especially since software development in this respect encompasses software systems development". This activity we present is applicable to both software or systems. We begin with a brief review of literature to establish the importance of and problems associated with requirements engineering and some of the challenges in teaching it, as well as examples of similar exercises that have been reported on. We then present our exercise in which students construct an artifact out of LEGOs©, which stresses information transfer between peers and calls attention to what requirements engineering is and why doing it well is critical to SDLC. We then discuss anecdotal evidence about the utility of this exercise, how it evolved, lessons learned, and thoughts on applying it to modified contexts for other instructors.

2. BACKGROUND

Requirements Engineering is a key activity of the SDLC and "involves using various fact-finding techniques, such as interviews, surveys, observation, and sampling" to identify requirements for software and/or a system under development (Tilley, 2020, p. 465). Properly communicating requirements between individuals involved in the SDLC is paramount to success in IT projects but also incredibly difficult to do well. Many of the top reasons that IT projects fail relate to poor requirements elicitation and communication issues (Hughes, Rana & Simintiras, 2017). Therefore, IS students must have a healthy respect for the difficulty in communicating and eliciting requirements as such will serve to underscore the importance of information documentation and transfer throughout the SDLC.

Software and IT systems are amorphous, dynamic, and complex. These traits increase the difficulty for organizational stakeholders to accurately and easily explain what the system should do and what it should look like. Cutting-edge technologies combined with outsourcing and virtual teams amplify these problems. Any IS course that teaches how to communicate and work in a team to engineer software systems needs to stress the importance of strong communication and include applied ways to show students how to do it.

Here, we present an exercise in which students assume one of two roles that are instrumental to the earliest stage of requirements engineering, which is the elicitation of requirements (thus, relying heavily on communication). The first role is that of a stakeholder, often defined as anyone affected by an organization's performance, which includes but is not limited to employees, suppliers, customers, and shareholders. We associate the term primarily with those future end-users of a system who try to articulate specific requirements they need. Our second role is that of an engineer, those professionals concerned with developing software/systems from inception, to operation, to ongoing maintenance (Sommerville, 2016). We chose this terminology as the exercise was developed in a Software Engineering course (see our explanation in the next section), however, we do not see our activity as limited only to the context of that specific course.

This exercise is based on LEGO© Serious Play™ (LSP), rooted in the pedagogical philosophy of Constructionism, which posits that when students construct artifacts, the process of creating requires engagement with classroom content, resulting in a more meaningful learning experience (Papret, 1991). LSP is a technique where "individuals build three-dimensional models using a special mix of LEGO© bricks designed to inspire the use of metaphors and story-making" (Rasmussen, 2006, p. 59). Although our exercise was not intended to inspire metaphor or to fabricate fiction per se, it nonetheless draws on similar benefits. Namely, we use construction blocks as a flexible medium for hands-on, experiential learning that allows students to solve problems and communicate through LEGO© builds (Peabody & Noyes, 2017).

Using LEGO© to help teach concepts in systems analysis and design is unorthodox but not entirely new. For example, Freeman (2003) used them in simulation and roleplay exercises. There, LEGO© simulated currency and stood in as building blocks. In Freeman's example, volunteers performed in front of the entire class, three per round in multiple rounds.

Furthermore, we are not the first to deploy LEGO©-based lessons for concepts related to system or software development. For example, Paasivaara, Heikkilä, Lassenius, and Toivola (2014) created a SCRUM-simulation game where Masters students learned about different roles and agile concepts by constructing a LEGO build©. Similarly, Gama (2019) reported on using LEGO© for hands-on, SCRUM-oriented lessons (adapted from industry-trainings) on understanding requirements and project planning and management. Meanwhile, Kurkovsky (2015) piloted five case studies utilizing LSP to teach lessons related to requirements engineering, software architecture, design patterns, socio-technical systems, and dependability dimensions.

Kurkovsky, Ludi, & Clark (2019) note that because of the array of possible approaches one can take to solving Software Engineering problems, "it is sensible to rely less on lecture and focus more on active learning experiences mimicking what students may encounter" in their future occupations (p. 218). Furthermore, additional research has supported that while LSP activities may seem simple on the surface, it has the potential to support coverage of numerous Software Engineering subjects (López-Fernández, Gordillo, Ortega, Yague, & Tovar, 2016). Indeed, one group of scholars reported that "the students learned more than we expected regarding requirements management and customer collaboration, effective teamwork and the Scrum roles" (Paasivaara et al., 2014, p. 390).
Although it may share similarities, our exercise differs from those referenced above in that it gives students experience in assuming key roles (engineers and stakeholders), and it focuses on highlighting communication between these roles. Given that building software (or even system design in general) is steeped in ambiguities (because software is not a discretely defined tangible good), stakeholders and engineers usually face unique challenges in communicating to one another about the product to be built and the nature of the problem the product is intended to solve. The artifact that the students build serves as a centerpiece around which they have this experience, and as a launching point for the discussion about communication that takes place following the exercise.

As explained in our introduction, this exercise served both as a gateway into the subject of requirements engineering, and as a lesson about the importance of doing it well (specifically, requirements elicitation). In particular, we sought to leverage our LSP Requirements activity as a means to impress upon students that requirements engineering is an activity which requires effective communication to solve problems. How we do this is now addressed in the details of the exercise.

3. THE LEGO-BASED EXERCISE

While we believe that this exercise may be applicable to a range of situations, our description of it is embedded within the Software Engineering course in which we developed and used it. Accordingly, we begin this section with background about the course to help the reader better understand its relevance as a learning activity. We then provide some fundamental background about the activity, in particular, explaining how applying LSP was relevant to a topic about problem-solving, communication, and constructing an artifact within the context of the SDLC.

In the remainder of this section, we present the instructions for our LSP exercise. Our description here is specific so as to explain how to conduct the exercise in the classroom. We include discussion questions to facilitate reflection on the exercise. In the next section (Discussion), we then suggest how others may adapt this based on their needs, audience, materials, constraints, etc.

A Software Engineering Context

This exercise was tested in an upper-level, undergraduate Software Engineering course that included Computer Science and Computer Information Systems students. While coding experience was assumed, the course itself was about the non-technical aspects of creating software, namely, the activities emphasized in the SDLC besides coding. The SDLC in this course consisted of specification, development, validation, and evolution (Sommerville, 2016). While these terms may differ from those of stages typically identified in the systems approach to the SDLC, many of the tasks, goals, and outcomes of both flavors of the SDLC are consistent. Notably, this includes requirements engineering.

Our LSP exercise took place about the fourth or fifth week of the semester, by which time the students had learned about Software Engineering and process approaches in general, namely traditional waterfall and agile. The remainder of the semester drilled down into various activities and processes embedded in the SDLC.

Because the LSP exercise served as an introduction to specification, it was the starting point for hands-on experience with SDLC activities as at this point, students were only familiar with the broad themes associated with each stage. Specification begins with requirements engineering, which is also a common term for a parallel set of tasks in the analysis phase of Information Systems development. Analysis consists of techniques and processes to elicit functional and non-functional requirements from organizational stakeholders (Sommerville, 2016; Tilley, 2020). Additional LSP activities related to SDLC concepts were also developed and employed in the same Software Engineering course over several semesters. However, for the sake of parsimony, this paper focuses only on this one activity related to Requirements Engineering.

We provide instructions for the exercise as an amalgamation of best practices developed based on implementing the exercise in the classroom. Over the multiple semesters it was done, the parameters were changed each time, with some versions proving more fruitful than others. More detail about alterations we've made to the exercise are explained following the directions. These adaptations may prove informative to those who wish to implement the exercise in their own instruction.

Overview of Exercise

Each package of LSP comes with booklets depicting five semi-abstract builds (hereto referred to as SABs) that can be created with the LEGO© included in the kit. We describe them as *semi-abstract* because they may resemble a known object or creature (such as a crocodile or bird), but they also include difficult-to-define characteristics that don't as well align with our understanding of these known objects or creatures. Four examples from the booklet are shown in Figure 1 for a sense of context. Of note, the numbers associated with each example correspond with instructions provided in the LSP booklets and are irrelevant to our exercise.

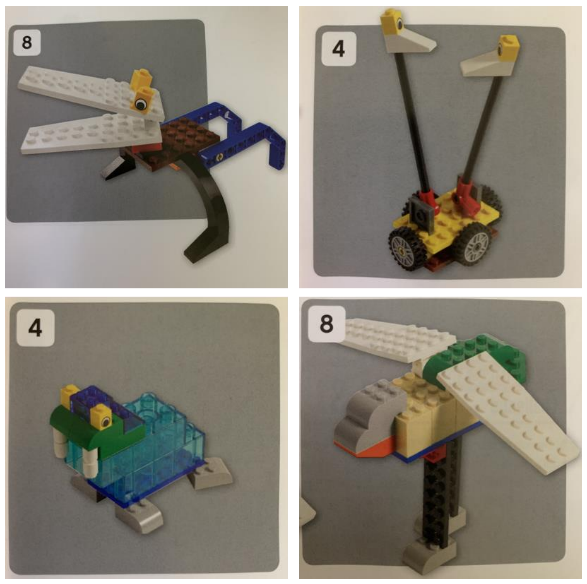

Figure 1: Semi-Abstract Builds (The LEGO Group, 2009)

By having a student familiar with a SAB direct another who is unfamiliar with the SAB to reconstruct it as accurately as possible, the "fuzzy" nature of these SABs serves as a surrogate to communicating requirements about because (like software/system requirements) they are not neatly defined and easily described. That is, we find an opportunity to highlight communication challenges that are present in requirements engineering by having students try to negotiate the particulars of reconstructing a SAB.

Exercise Directions
At the start of the class session, students are told that they'll participate in an activity requiring them to assemble in pairs, with one playing the role of a stakeholder (simply referred to here as Stakeholder), and the other playing a software engineer (or simply, Engineer). They decide which student assumes which role. They're told that the exercise relates to the topic of requirements engineering, but that how it relates will be explained following the activity. Students are then given the following specific instructions:

For this exercise, you'll be working together to replicate an assigned LEGO© build as accurately as possible. Whomever you decide is the Engineer will do the actual construction of the build, while the Stakeholder will provide instruction about what that build should look like. Throughout this exercise, only written and verbal communication may be exchanged meaning no drawings or photographs may be shared in any way.

In a few minutes, you and your partner will be separated. Engineers will remain in this room at their seats, and will each receive one LSP kit to familiarize yourself with until you get instructions about what to do next. Meanwhile, Stakeholders will be sent to a nearby room where they will receive an image of a build they'll ask their partner to reconstruct. They will have a few minutes to write, but not draw, instructions for their partners to reconstruct it (such as describing what to build and/or how to build it).

Pairs will meet initially in the hallway (halfway between rooms) for two minutes to exchange instructions or additional information. Then, over the next 45 minutes, the instructor will periodically (every 10 minutes or so) call a meeting where you can briefly reconvene in the hall to ask questions, provide additional written instruction, or pantomime about the build.

After 3 to 4 iterations, the exercise will end and Stakeholders will be invited back to the room with their Engineer partners to see how well they were able to reconstruct the build. A short discussion about the exercise will follow, the lessons from which will underlie our next few class sessions.

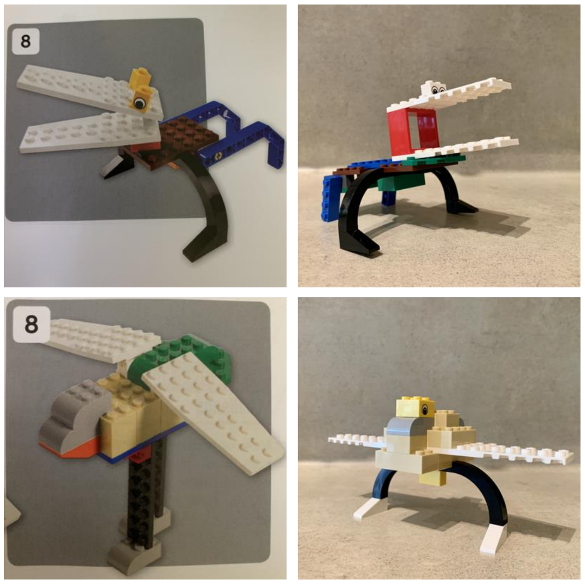

Figure 2: Comparison of LSP Builds

Exercise Outcomes
The result of the Engineers' efforts typically bore an approximate likeness to the SAB, but was imprecise in detail. For example, they may have created a creature with extended wings and long legs, or a crocodile-like four-legged animal (as shown in Figure 1) but done so without accuracy as seen in Figure 2. Figure 2 illustrates the original SABs on the left, and examples of what student outcomes typically resembled on the right. Of note, we reconstructed these student builds and photographed them for this paper as the authors had not previously documented actual student builds following the exercise.

In-Class Discussion
Students playing the role of Stakeholders tend to be quite enthusiastic about seeing the extent to which the Engineers were able to accurately reconstruct their build, while Engineers are keen to learn what image the Stakeholder was working from. As a result, when the class reconvenes at the end of the exercise, the instructor should give the students a few minutes to discuss amongst themselves how they think the exercise went, and how/why their results were as they were.

Instructors do need to ensure that they have at least 5 minutes at the end of class to emphasize that the LEGO© activity was intended to be a first-hand experience to relate to as the following class sessions go into various aspects of requirements engineering, and in particular, requirements elicitation. This activity is meant to be a launching point, and so before the students are dismissed from the class session, the point needs to be stated explicitly: "*you just experienced that* *bringing someone else's vision to life is inherently a difficult task, and one that is inextricably bound to successful communication."*

The discussion about the exercise itself may take place, if time permits, during the same class session or during the following one if there is not sufficient time remaining. Regardless, some relevant questions to pose to students should be similar to what follows.

First, we asked students *to what extent did you find that the build constructed by Engineers reflected the picture of the build held by Stakeholders?* The purpose of this question is to draw attention to the fact that despite the incredible unlikelihood that any pairs got their final product to match the abstract build image, they likely were able to achieve notable similarities.

This first question is important because it shows that communicating ideas isn't, in and of itself, difficult (as evidenced by most teams achieving a rough approximation); but that precise and accurate transfer of detail about an idea from one person to another, is! This is something that the second question seeks to explicitly draw out further by asking *which factor or factors do you believe most influenced discrepancies between the picture and the final product*? Here, students will likely articulate that less obvious concepts like brick size, color, and specific arrangement were points that were less likely to be thought of as relevant facets of communication, whereas the larger picture received most of the focus.

Third, to Engineers, *what did you find most challenging about putting your build together*? This question is meant to probe students' sense about their awareness to the fact that describing and communicating an abstract idea presents unknown unknowns, that is, that the Stakeholders don't necessarily have an idea about the sorts of tools and constraints that are very much present for Engineers, and therefore, are unlikely to account for potential misunderstandings.

This is similar to our next question, which is posed to Stakeholders, *what would have been most helpful to you to better communicate the nature of your vision to Engineers*? Notably, we know from experience that most absent from the exercise is useful feedback to Stakeholders as Engineers work. As such, this question brings that point to the foreground, which in turn serves to underscore the same point from question three. That Stakeholders and Engineers work with fundamentally different vocabulary, concepts, and in varying circumstances. This is a point that is revisited time and again not only in the lessons on requirements engineering, but throughout the remainder of the course.

Through this exercise, students have practiced taking on a role (Engineer or Stakeholder) and have had the ability to observe how their role's position has impacted their teammate's role. Now, through discussion sparked by questions such as the above, the instructor has the opportunity to highlight the importance of communication in respect to not only asking the right questions and providing the clear answers, but also in being as specific as possible. The role of detailed documentation is also a point of relevance to establish to minimize variances in potential interpretations.

4. DISCUSSION

In this section, we discuss concerns related to the exercise including anecdotal evidence of student engagement, multiple variations we've tried (and challenges faced), and some suggested tips that may be useful for future iterations. All of this is intended to assist the reader in determining its applicability to their teaching.

Evidence of Outcomes

Although capturing data on, analyzing, and reporting the learning outcomes of this exercise is beyond the scope of this paper; we do have some anecdotal evidence supporting the value of our LSP activities in respect to Requirements Engineering and other SDLC-related concepts.

First, the instructor of this course consistently noted that students were particularly enthused on days in which LSP activities were a part of the course curriculum. In fact, a reputation developed around the use of LEGO© in this course as one student wrote in their end-of-term (EOT) evaluation, *"coming into the course, I wasn't sure how I really felt about having to do all of the class activities I heard about, but in the end they were very entertaining and helpful for the class."*

This enthusiasm and engagement around LSP activities was not simply limited to this course as another student wrote*, "I really enjoyed the Lego activities it made the concepts of software engineering fun and I wish my other computer classes could be more hands on like this."*

This theme of not only being engaged, but recognizing the practical utility of the exercise, emerged in other comments made by students in the EOT evaluations, with one stating, "the in-class activities were the best part of the course, and allowed us to tie what we learned during lecture into the whole scheme of things," and another writing*, "the in class activities were not only really fun but really helpful with understanding the material."* Figure 3 is a photo taken from one of the in-class activities. While these comments are not necessarily solely focused on the LSP exercise described in this paper, they do lend support to the value of this or similar hands-on, non-technical activities related to teaching the SDLC.

These instructor observations and comments reflect those that have been reported by other scholars utilizing LEGO© for similar purposes. Among these is that such activities are highly-popular and allow students to reflect meaningfully on course concepts (Paasivaara et al., 2014); that class sessions which include LEGO© tend to be more interesting (Gama, 2019); and that they assist students in achieving learning goals (Kurkovsky et al., 2019). Additionally, these activities elicit creativity and imagination around subject matter (Kurkovsky, 2015) which may otherwise be difficult to present in engaging ways.

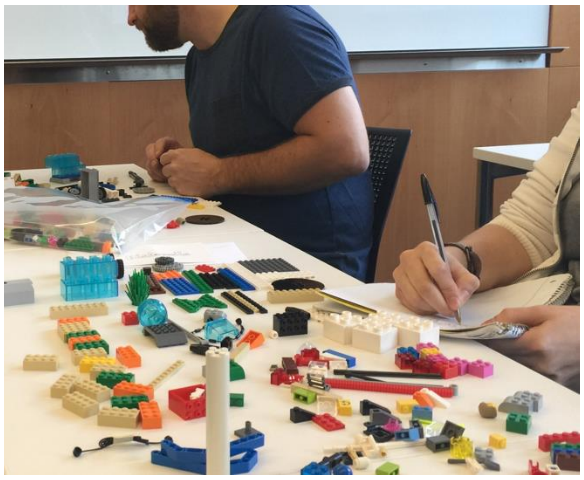
Figure 3: Software Engineering Students working with LEGO© Serious Play™ in class

Arguably, another point of convergence between our anecdotal observations and comments that was also remarked upon in the literature is that "many students reported looking forward to more" (Kurkovsky, 2015, p. 218) LEGO© activities and that students find these "highly fun and motivating and very useful" (López-Fernández et al., 2016, p. 10).

Past Variations
As stated in the previous section, our exercise has gone through iterations in which we've made adjustments and alterations to what is presented in the directions. One major point we've struggled with is if Stakeholders should be allowed to create images or provide Engineers with any types of visual information. We allowed this in at least two iterations, only to find that Stakeholders were trying to either trace the final build, or to draw it out as realistically as possible. Such approaches defeat the spirit of the exercise which is to illustrate the difficulties inherent in describing, explaining, or otherwise conveying semi-abstract ideas to others. In the end, it also deemphasized verbal communication which is often the primary form of ideation for Stakeholders.

Similarly, we tried one iteration in which Stakeholders could sit next to Engineers so they could see the build in-process, and so that communication could be constant between them. Stakeholders tended to, again, subvert the spirit of the exercise by stating specifically bricks to pick up and where to place them; treating the Engineer almost as a surrogate rather than as a conduit through which their information could be transformed into a build. In this approach, the semi-abstract nature of the builds was lost in translation as there was little need to try to explain them from Stakeholder to Engineer.
We also played around with the extent to which Stakeholders were allowed to see the builds in progress. One iteration saw the Stakeholders welcomed into the room where the builds were happening, periodically for a minute or two. This allowed them to give overly specific verbal instructions (as when they were allowed to sit side by side), albeit in a far briefer period of time.

Given the exercise's emphasis on demonstrating the difficulties, first-hand, in translating and transmitting a semi-abstract idea from one person to another; we believe it works best to minimize feedback to the Stakeholder about the Engineer's progress. This then puts the onus on the Engineer to try to verbally describe their build back to the Stakeholder, which, again, puts the spotlight on communication as key to the success of requirements elicitation.

Modifications and Additional Applications
Although this exercise was conceived of, and constructed around LSP, other instructors may adopt materials that are more suitable to their purposes. One author of this paper invested in a dozen and a half LSP kits following their introduction to it at a conference, and the SABs pictured in the manuals included in each kit

presented a ready-made centerpiece to this exercise. However, we can certainly envision that similar outcomes and lessons may be had if different materials were deployed provided that (1) Engineers have all of the materials to build that which Stakeholders direct them to, and (2) that the builds Stakeholders instruct Engineers to construct are semi-abstract in nature.

While we executed this activity in a Software Engineering course, because the SDLC is taught (albeit with different foci) in both Computer Science and Information System oriented disciplines, the LSP exercise is just as suitable in any course where understanding or documenting requirements is an integral lesson. We can see this activity being adopted in courses in Systems Analysis and Design, Project Management, and even User-Experience Design.

5. CONCLUSIONS

In the game of telephone, one might start with a phrase and end up with an entirely non-sensical salad of words. In our LSP exercise, we start with creations that are semi-abstract and end up with builds that, more often than not, resemble their originals tangentially with the finer details lost. In both cases, we find that communication between senders and receivers is easily muddled with the clarity of meaning lost in translation.
Given that requirements engineering is an early and key activity in the SDLC, it is important that students understand how vital communication between stakeholders and developers are. This paper presented a LEGO©-based activity intended to highlight just how easy it is for ideas to get misinterpreted when engaging with individuals who have different perspectives but have the same goal in mind. It is our hope that instructors will adopt, adapt, and apply this exercise in ways that work best for them in their classroom.

6. REFERENCES

Chakraborty, A., Baowaly, M. K., Arefin, A., Bahar, A. N. (2012). The Role of Requirement Engineering in Software Development Life Cycle. *Journal of Emerging Trends in Computing and Information Sciences(3)*, 5, 723-729.

Davey, B. & Parker, K. R. (2015). Requirements Elicitation Problems: A Literature Analysis. *Issues in Informing Science and Information Technology*, 12, 71-82.

Gama, K. (2019). An Experience Report on Using LEGO-based Activities in a Software Engineering Course. *SBES 2019*, September 23–27, 2019, Salvador, Brazil.

Ghanbari, H., Similä, J., & Markkula, J. (2015). Utilizing online serious games to facilitate distributed requirements elicitation. *The Journal of Systems and Software*, 109, 32-49.

Hughes, D. L., Rana, N. P., & Simintiras, A. C. (2017). The changing landscape of IS project failure: an examination of the key factors. *Journal of Enterprise Information Management(30)*, 1, 142-165.

Kurkovsky, S. (2015). Teaching Software Engineering with LEGO Serious Play. *ITiCSE'15*, July 6–8, 2015, Vilnius, Lithuania

Kurkovsky S., Ludi, S. and Clark, L. (2019). Active Learning with LEGO for Software Requirements. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, February 27-March 2, 2019, Minneapolis, MN, USA. New York: ACM.

The LEGO Group (2009). LEGO© Serious Play™: Imaginopedia for Core Processes (booklet). The LEGO Group, Denmark.

López-Fernández, D., Gordillo, A., Ortega, F., Yague, A., and Tovar, E. (2016). LEGO® Serious Play in Software Engineering Education. *IEEE Access(4).*

Paasivaara, M., Heikkilä, V., Lassenius, C., and Toivola, T. (2014). Teaching Students Scrum using LEGO Blocks. *ICSE Companion'14,* May 31 – June 7, 2014, Hyderabad, India.

Papret, S. (1991). Situating Constructionism. In S. Papert & I. Harel (Eds), *Constructionism: Research Reports and Essays, 1985-1990* (pp. 1-11). Norwood, NJ: Ablex Publishing Corporation.

Peabody, M. A. & Noyes, S. (2017). Reflective boot camp: adapting LEGO© SERIOUS PLAY™ in higher education. *Reflective Practice(18)*, 2, 232-243.

Rasmussen, R. (2006). When You Build in the World, You Build in Your Mind. *Design Management Review*, 17, 56-63.

Ruparelia, N. B. (2010). Software Development Lifecycle Models. *ACM SIGSOFT Software Engineering Notes(35),* 3, 8-13.

Sommerville, I. (2016). Software Engineering, 10th Edition. Boston: Pearson Education Limited.

Tilley, S. (2020). Systems Analysis and Design, 12th Edition (Shelly Cashman Series). Boston: Cengage Learning.

Zowghi, D. & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum, & C. Wohlin (Eds) *Engineering and Managing Software Requirements* Berlin, Heidelberg: Springer.