WIP: Building a Bridge Between Hackathons and Software Engineering Capstones Through Adaptive Expertise

Cecilia La Place, Arizona State University, Polytechnic campus

Cecilia La Place is a first-year Ph.D. student at Arizona State University (ASU) studying Engineering Education Systems & Design. She has received her M.S./B.S. in Software Engineering through an accelerated program at ASU. She began researching hackathons after she joined the Fulton Undergraduate Research Initiative (FURI) in her junior year. This stemmed from her love of learning in hackathons having participated in numerous hackathons from as far west as Southern California to as far east as Pennsylvania.

Dr. Shawn S. Jordan, Arizona State University, Polytechnic campus

SHAWN JORDAN, Ph.D. is an Associate Professor of engineering in the Ira A. Fulton Schools of Engineering at Arizona State University. He teaches context-centered electrical engineering and embedded systems design courses, and studies the use of context and storytelling in both K-12 and undergraduate engineering design education. He received his Ph.D. in Engineering Education (2010) and M.S./B.S. in Electrical and Computer Engineering from Purdue University. Dr. Jordan is PI on several NSF-funded projects related to design, including an NSF Early CAREER Award entitled "CAREER: Engineering Design Across Navajo Culture, Community, and Society" and "Might Young Makers be the Engineers of the Future?," and is a Co-PI on the NSF Revolutionizing Engineering Departments grant "Additive Innovation: An Educational Ecosystem of Making and Risk Taking." He was named one of ASEE PRISM's "20 Faculty Under 40" in 2014, and received a Presidential Early Career Award for Scientists and Engineers from President Obama in 2017.

©American Society for Engineering Education, 2020

Work In Progress: Building a Bridge Between Hackathons and Software Engineering Capstones Through Adaptive Expertise

Abstract

As hackathons become more commonplace and accessible at universities around the world, surges of undergraduate Computer Science and Software Engineering students can be found attending these events to have real world development experiences. Meanwhile, faculty find themselves continuously adapting themselves and their curriculum to prepare students to be adaptive experts, one who leverages prior or similar knowledge to solve new problems in new contexts, when they enter the workforce. Capstones and culminating experiences test students' abilities to be adaptive experts, but students are not always prepared to meet the challenge. Hackathons present a unique but accessible opportunity to gain more adaptive experience prior to and during capstone experiences. In this work in progress pilot study, the hackathon and capstone experiences of graduated software engineering students are compared through an adaptive expertise framework to begin exploring how hackathons can supplement academic experiences.

Introduction

Educators face a difficult problem: teaching students how to solve problems they have never seen before. Despite their best efforts, some students express feelings of unpreparedness when entering the workforce as an intern or new full-time hire. Students in Computer Science (CS) and Software Engineering have begun to leverage coding marathons known as hackathons to ease this concern, believing they are developing real world experience in the process.

In the past decade, hackathons have been on the rise, and CS and software students are eagerly throwing themselves into hackathons. Warner and Guo found that student participants say hackathons give them more learning and networking opportunities than their schooling (2017). The goal of these 36-hour coding marathons is not to encourage malicious activity such as breaking into systems (e.g. hacking into emails). Instead, they encourage developing technical solutions to problems presented or designing projects around themes (Briscoe & Mulligan, 2013). However, few understand what other impacts hackathons have. Even less understand how hackathons impact students. Some work has begun to address knowledge transfer within hackathons, specifically how students are sharing and receiving knowledge ((La Place et al., 2017)). There remains a missing link in understanding what knowledge students bring into hackathons and share with other participants, and how students use software process in their hackathon projects.

This work in progress pilot study looks at a group of students from a project-based undergraduate software engineering program at Arizona State University (ASU). The program was designed to develop each student into an "agile engineer, a lifelong learner with a comprehensive set of skills appropriate to the needs of today and tomorrow" (Roberts et al., 2007) Students in this program have been taught to apply skills learned through project-based courses with the intent of also learning how to contextually apply knowledge to solve different problems (Gary, 2015). For students in this opportunistically structured program, hackathons

present a potentially familiar environment though shorter in duration. The projects developed in each capstone and hackathons will allow for an exploration into a selection of skillsets software engineers bring to hackathons, and the processes used in their projects both consciously and unconsciously.

This work will inspire a series of research following knowledge transfer within hackathons as more domains such as engineering, math, science, and art join the event and shape development processes. Though motivational studies on hackathons are thorough, considering how these motivations play into the projects developed at hackathons may lend to a deeper understanding of student experiences, learning possibilities, and potential career-defining moments that can be leveraged by university courses.

Literature Review

Hackathons

In order to understand knowledge transfer, we must first understand the environment in which it occurs. Hackathons have received varied levels of attention in research, news, and communities. CS and software engineering students flock to hackathons around the world eager to build something with their friends, win a prize, and learn something new (Pedra, 2019). The community is growing too as more engineering and non-engineering disciplines attend (Pedra, 2019). The hackathon culture is rapidly expanding from where it first started in the Northeastern US, becoming a worldwide phenomenon (Swift, 2019).

Research on hackathons have categorized the many different types that exist such as tech and focus centric (civic oriented) (Briscoe & Mulligan, 2013) or as 24-hour, industry, or competition hackathons (Porras et al., 2018). Others have focused on hackathons in industrial settings, identifying hackathons as a new way to innovate by leveraging companies own employees instead of their research teams (Flores et al., 2018; Komssi et al., 2015). At a collegiate level, a majority of hackathon research is centered around discovering what students think of hackathons (Warner & Guo, 2017) or using hackathons as pedagogy to reinvent classroom experiences (Calco & Veeck, 2015; Gama et al., 2018). Hackathons are also being used as a way to develop solutions to existing problems such as encouraging CS interest in college students (Mtsweni & Abdullah, 2015), diversifying tech (Richard et al., 2015), or resolving local community concerns such as homelessness (Linnell et al., 2014) or self-harm (Birbeck et al., 2017). Other community efforts appear in civic hackathons, where governments leverage open source development and its local community of amateur to professional developers (Gama, 2017a). Despite the numerous ways that hackathons can be used to impact other people, classes, and communities, there is little work on how hackathons impact its participants.

The largest accessible group of research participants, undergraduate STEM students, are also the least studied in the literature. Aside from the previous examples, preliminary research in collegiate hackathons has identified small scale knowledge transfer within teams ((La Place et al., 2017)). One of the only mixed-methods approaches confirmed student motivations are centered around learning and networking, but also began to elicit from a small population why some students do not attend hackathons (Warner & Guo, 2017). Current research understands

why students go to hackathons, but not how the hackathon experience affects participants, nor what hackathons provide for students or educators outside of the main motivators (Briscoe & Mulligan, 2013).

Adaptive Expertise

In order to develop engineers that are "experts who can adapt to novel situations and learn" (Schwartz et al., 2005) we must "situate the research in a setting that allows—in fact requires and rewards—learners to use knowledge in novel ways, i.e., to be innovative," (McKenna, 2007). However, Schwartz, Bransford, and Sears mention that having students apply their skills in capstones after minimal exposure to thinking courses is not enough to promote adaptive expertise (2005). They refer to Hatano and Inagaki's 1986 work in which they believe long-term processes are critical for adaptive expertise development (Hatano & Inagaki, 1986). In short, adaptive expertise must be practiced repeatedly over time which traditional lecture-based curriculum does not always allow for. As a result, students first exposure to adaptive expertise is in their capstones or culminating experiences at the end of the degree. However, some academic programs allow for the opportunity to conduct adaptive expertise-based research.

Adaptive expertise research is frequently situated in design challenges, education reform, and knowledge transfer. In Peng et al's work, two groups of undergraduate students across all academic years were asked to create a CAD design from a real-life object and a drawing (2014). The study focused on evaluating contextual exercises to measure and help the development of adaptive expertise characteristics in the classroom. In another study, Vanasupa et al establish that developing motivations to learn and making value visible is critical for adaptive expertise development over time (2010). Meanwhile, McKenna sought to understand how design knowledge transferred between design experiences (2007). Design of problem solutions and curricula are extremely popular due to the need to create versatile engineers that can solve new problems, but these approaches are not without their limitations.

In practice, there have been other drawbacks to studying adaptive expertise in academics. The caveat of adaptive expertise when coupled with knowledge transfer are "transfer studies focus[ing] too narrowly on measuring 'replicative' or procedural knowledge," (McKenna, 2007). Furthermore, McKenna argues that "traditional transfer approaches do not focus on capturing types of knowledge individuals are capable of transferring into new situations, or on the types of resources that better prepare students for subsequent learning" (2008). In order to effectively use this framework, this study must overcome the limitations of past works.

Addressing the limitations allows for the adaptive expertise to bridge the gap between knowledge transfer and software development process. Hackathons are a prime environment for adaptive learning in that learners design their own experience and use of knowledge. Motivated by prizes and other motivations, teams must design projects that satisfy prize criteria, and provide them with a competitive edge against other teams competing in the category. No project will be the same as another within a hackathon even within the same prize category. The frequent occurrence of hackathons coupled with unique project experiences creates an ever-changing space to study adaptive expertise in action.

By comparing capstone and hackathon experiences, how knowledge is transferred between the two can be identified, and what preferred resources for problem solving can be elicited. Software process has been found in civic hackathons, where governments use public transparency to crowdsource software, but in modified forms (Gama, 2017b). However, this study was conducted outside of a collegiate environment and does not clarify the development expertise of the participants interviewed which could have ranged from amateur to professional. Previous work on knowledge transfer within collegiate hackathons highlighted some resources participants used, but was limited in understanding the extent of what knowledge was brought into and out of a hackathon and did not address the process in which students worked on projects (La Place et al., 2017).

Methods

To extend the previous knowledge transfer work and software development work, we offer the following research questions:

1. *What technical knowledge do students use in capstones and hackathons?*
2. *Where do students learn the knowledge used in capstones and hackathons?*
3. *How does the software development process used by students differ between capstone and hackathon projects?*

This is a qualitative pilot study meant to fuel future research on knowledge transfer between hackathons and academic experiences. The nature of hackathons often results in participants designing and developing a project that results in learning new skills. Despite this being a short-term event, ASU software engineering students have a very similar experience over a longer period by developing projects that apply the required skills and concepts within a classroom setting.

Following IRB approval and receiving participant consent, qualitative interviews from five graduated software engineering students were collected using an artifact elicitation methodology (Douglas et al., 2015) leveraging an adaptive expertise framework (Schwartz et al., 2005). The interviews underwent preliminary analysis using Glaser and Strauss's grounded theory (1967) and thematic analysis (Saldaña, 2012) to develop an overall understanding and familiarity of the data for this work in progress.

Context

ASU's undergraduate software engineering students are an ideal population to look at. Students who have taken the full curriculum have been subject to semesterly project-based courses that have them designing a project to help them develop key software skills over time. Each year builds upon the previous year's skills, thus creating a long-term process development approach. The first year and a half of the program begins with introducing students to programming concepts and other general sciences (ASU Software Engineering Major Map). In the second half of the second year, a project spine is introduced in the form of a software development project class. Every semester, students extend their software development knowledge. Throughout junior to senior year, students are also expected to take classes in specific focus areas of their choosing,

which may also employ project-based course structures (ASU Software Engineering Major Map).

More specifically, the software engineering curricular design has sophomores learning individual professional skills and data structures and algorithms through a semester project, juniors designing and implementing focus area concepts (Web, mobile, or game development) in a year-long project, and seniors synthesizing advanced concepts in their industry capstone project over their final year (Gary, 2015). Though hackathons are short-form experiences, no two projects are ever alike, much like the projects found in this curriculum (Gary, 2015). Students within this program have been taught to synthesize and apply concepts across different problem contexts.

The participants in this pilot study have attended hackathons across the United States. The hackathons described have all been collegiate hackathons supported by the Major League Hacking (MLH) organization and as a result were all similar in structure. An MLH-supported hackathon begins with an opening in which the organizers describe the themes and prizes available to the participants. Throughout the weekend, participants have access to free food, can contact mentors for help, and workshops for learning experiences while they work on their hackathon projects. At the end of the hackathon, participants are invited to demonstrate their project to other participants, visitors, and judges.

Participants

As a result of strict selection criteria for the study, it was most appropriate to use snowball sampling. Five participants were contacted via email and interviewed either in-person or remotely. The participant needed to have been through the sophomore, junior, and senior project courses of the ASU software engineering program. They must not have been under a non-disclosure agreement (NDA) from when they completed their capstone in order to protect both the participant and the researchers from accidental NDA breaches. Finally, they must have been to at least one hackathon within two years of their capstone. These criteria set the stage for a comparison of skillsets developed as a result of the program and hackathon experience. Though it was not a requirement that they have graduated, the results of the sampling led to only graduated students being available for interviews. The use of graduated students provided the opportunity to have the participants reflect on their past experiences having completed their capstones and hackathons. As a result of the selection criteria, some of the participants shared either a capstone experience or a hackathon experience with at most one other participant. Demographic data for these participants were not collected.

Data Collection

Participants were asked to bring two artifacts, their capstone project and a recent hackathon project, and then participate in artifact elicitation interviews (Douglas et al., 2015). Artifact elicitation seeks to understand information about the artifact, and information surrounding when and how the artifact was constructed. By using the artifact as a focus point to understand how the builder was involved with the creation of the artifact, we can also understand what knowledge was present, obtained, and used in its creation. Artifact elicitation has previously been used to understand “the knowledge skills and attitudes,” (Douglas et al., 2015), “the process of designing

technology to support familial relationships (Paay et al., 2009), and understanding youth design (Eyerman et al., 2018) in engineering contexts. For this study, artifacts were used to help participants reflect on their project and the process in which it was built in order to elicit rich description. Artifacts were not collected but were used a memory device throughout the interviews. Participants were free to share the artifact with the researchers during the interview but were not required to do so.

The semi-structured interview protocol was developed based on the adaptive expertise framework (Schwartz et al., 2005) and situated in capstone and hackathon experiences based on the first author's experiences. Interviews were each an hour but were split into two parts. Each part was 30 minutes and covered each project. If there was time at the end of the interview, participants were asked to reflect and describe perceived comparisons between their project experiences. Some questions from the overall interviews are as follows in Tables 1 and 2:

Questions	**Research Question**
Tell me about an instance where you had a roadblock in your project	1, 2
How did you go about navigating that roadblock?	1, 2
Tell me about your personal development process in this project	3
Tell me about the process your class required for this project	1, 3

Table 1. Capstone Project Interview Questions

Questions	**Research Question**
How did you come up with the project?	2,3
Tell me about an instance where you had a roadblock in your project	1, 2
How did you go about navigating that roadblock?	1, 2
Tell me about your process for this project	3
If you were to add a new feature to your project, what would it be and how would you do so?	1, 3

Table 2. Hackathon Project Interview Questions

Information about some decisions that occurred during the project's construction may be missing within the interview due to lack of perceived importance from the engineer being interviewed. As a result, there may be gaps in the processes that are derived from these interviews.

Validity

Interviews were transcribed by the researchers to develop familiarity with the data and ensure accurate transcriptions. Member checking was also employed to build trust with the participants and ensure they had full control over the data they provided for this study. Participants were asked to review the transcribed interviews and make corrections, clarifications, and other modifications as needed. Participants were also asked to select a pseudonym or confirm a suggested pseudonym if they were unable to provide one for anonymity in this study.

Analysis

Preliminary analysis was conducted on the transcribed interviews in a qualitative analysis program known as Dedoose using a grounded theory approach to create a high-level thematic understanding of the data. The approach used Glaser and Strauss's constant comparative method (1967) which involves identifying "incidents" that are then coded into categories and compared to other coded incidents. The categories for this study were created using an In Vivo coding method (Saldaña, 2012), using the participants words to create the code, and compared to other participant's incidents to determine applicability. Next, the codes were reduced by categorizing the preliminary codes into presenting themes. Finally, a model of the relationships between the themes was developed.

Preliminary Results

Theme	**Definition**	**Example**
Challenge	Challenge refers to any time a participant encounters a perceived blockage in their work. This can be technical, physical, and emotional. It can often be as a result of lack of knowledge or lack of available resources.	"Plus, we spent a good chunk of time, I want to say several weeks, debugging, debugging, debugging, until finally we had a good work around, and I do remember this memory distinctively of the capstone because I always think of when employers ask difficult challenges you had to face and how you worked around it and this was by far one of the most difficult technical challenges I had throughout my undergraduate." – Frankie
"Break it Down"	"Break it Down" refers to when participants assess a problem and describe discrete actions to attempt to solve it. This can be separating a team into groups or individual tasks or laying out a personal course of action for a specific or general problem.	"Using that information, we were able to then say okay what would be 5 meters in front of me assuming the camera was pointing straight forward. Place a point 5m in front of me. And now draw a line between where the camera is currently located on the ground to the point." - Porter
Minimum Viable Product (MVP)	MVP refers to when participants describe what their project or component's success criteria is. This can come in the form of formal and informal requirements,	"I think that pushing it that extra 10% was the biggest push to reach the MVP that we had because it's given it was a capstone it was more of a tech demo showing can we

	diagrams, and user design. It is closely linked to the following theme.	do this and the answer is, 'yeah, we can just a little more and we can get there.'" - Porter
Priority Evaluation	Priority evaluation refers to when participants identify a need to re-evaluate the state of the project. This can come in the form of identified constraints such as time, stakeholder requirements, and project needs. It can also appear as adjusting the MVP due to the constraints.	"Eventually, since time was a constraint and we just needed to get it working and we were constrained on how long we had to work on the project, we decided there was a different implementation of SocketIO for Unity, but it was part of a Unity plugin or package that was paid. We ended up just buying that license to use that and it worked, 100% fine. They had a trial that we had used beforehand to validate that it actually worked." - Mark

Table 3. Preliminary Themes

Challenge

The theme of challenge was apparent from the start. When participants were prompted to describe impediments, issues, and problems encountered in their projects, they often expressed feeling challenged, much like Frankie's code example in Table 3. Their perceived challenges were knowledge gaps when they did not know what they needed to know to complete their project. Porter describes the challenge of drawing a line in their capstone project, stating that "it took [his team and him] quite some bit to figure out what [they] had, how to basically draw a line between where the camera was located and some arbitrary distance 5m in front of [them]." Porter mentions that the technology and features that their team needed did not exist yet due to the library's "pre-release/pre-alpha" state. Frankie, similarly, consistently describes the challenges of learning Virtual Reality and Augmented Reality frameworks and techniques in a time where it was still a new field, and few mentors were available.

Challenges also appeared as features that would not come together as easily as planned, such as Mark's experience with a familiar library in both his capstone and hackathon project. In his capstone project, he found himself having problems integrating the library. To work on the problem, he "opened GitHub issues" for the library before "attempt[ing] to fork the library and fix the issue [him]self." He and his team did not want to pay for the alternative library for their platform, but unfortunately, ended up having to as a result of time constraints. The library was used again for his hackathon project, but this time was free and open source for the platform he was using. However, it was a matter of detangling logic that challenged him. Notably, every participant's challenge was considered resolved in some way if it allowed them to continue on with the project, even if was not solved as originally planned.

"Break it Down"

Frequently, participants broke down the steps they took to solve problems, broke up teams to tackle different issues and tasks, and described their typical approach to resolving knowledge gaps. Stepwise descriptions litter the interviews when participants were asked to describe their various processes as exhibited in the code example in Table 3. When planning a project in capstone and hackathon environments, individuals or groups were selected to accomplish specific categories of tasks such as frontend sub-teams and backend sub-teams. In some of these sub-teams, further identification and divvying of tasks occurred. To acquire new knowledge and face new technology, Alex described breaking the task into smaller accomplishable programming tasks that built up to the original task's main goal. Frankie mentioned that breaking the task into smaller parts allowed for rapid prototyping of new components that once successful and refined, could be integrated into the main project.

Regardless the project, all participants use this technique to handle the challenges they face. By taking the task or goal and making smaller accomplishable tasks, Alex conveyed that they "gained confidence" through "little wins" that allowed them to tackle bigger and more difficult challenges in unfamiliar knowledge spaces. As a result of diagramming and requirements experience, Frankie frequently identified key inputs, outputs, and functionalities of projects he is about to begin in order to start compartmentalizing how the project can be built from the bottom-up.

Minimum Viable Product (MVP)

Throughout both capstone and hackathon projects, participants always returned to their MVP to ensure they were on the right track. Both hackathons and capstones had demonstrative elements that required the participants to develop their project in a way that was functional and could be used by stakeholders and possible users in a live demonstration indicated by Porter in Table 3. As a result, it was important to the participants to understand and distinguish from the start of each project what is the MVP.

> *"I think that especially for a hackathon type setting it's important to focus on what is your MVP and getting to that point. A lot of the ideas that are tossed about are […] going to be added to the important things we need to do and that's going to be added to the MVP or that's cool, goodbye we have 48 hours left" - Porter*

In both environments, formal and informal uses of requirements elicitation and diagramming of the project occurred. Formal uses typically involved following classroom requirements and extensive documentation. Informal uses did not involve any documentation aside from the occasional free form diagram or discrete requirements between sub-teams to ensure component integration success. As seen from Porter's excerpt, the MVP took form conversationally as key ideas were kept and requirements were designed. It is important to note that MVP's in hackathons are not always met as show in the following quote from Alex:

> *"I was working on one very specific thing and other members were working on very specific things. It ended up not... It didn't end up fully coming together. There was a lot of brand new stuff that we were doing." - Alex*

Requirements in hackathons are dependent on the team and how responsibilities are broken down. An excerpt from Mark's hackathon project shows a slightly contrasting way that he planned his work with his teammate.

> *"Especially since we had divided the work between the frontend and the backend. We had to come up with the requirements between them so that we weren't working on two totally independent things so that they would be able to connect." – Mark*

The capstone course, as Frankie mentions, involved following a "strict and orderly" project lifecycle. He describes how "[his team] spent a good few months on design and then [they] started working on requirements and then coding and then testing and then finally deployment." Though this makes for a stark contrast between the environments, we still see evidence of software development process in the project developments.

Priority Evaluation

When participants were not on the right track or had constraints affecting project progress, priorities were always re-evaluated to determine the next steps that were necessary to develop the MVP. Time was a constant constraint in both capstone and hackathon environments, and often determined whether specific libraries, methods, or frameworks would be used. At times, constraints also affected the timeline of feature development and coding habits when paired with specific lifecycle and course requirements.

In capstone, priorities were time and course requirements. In Mark's example in Table 3, he mentions that time constraints led to buying a license for an alternative to the library they were trying to use. In regard to course requirements, Frankie describes a "lingering pressure to get some code done in the sprint just to satisfy our stakeholders." As a result, he says he and his team "started kind of aimlessly coding pointless things to add to this just to say [they] got the code done." The time constraint can also lead to "blue sky features" being planned, but never actually executed as Seth explains.

Hackathons present some different constraints besides time. In the case of Alex's experience, it was only his first hackathon and did not expect the environment he was, "it was kind of like that hierarchy of needs." Though he did reach his goal, he focused more on the "other things in [his] brain that [he] was trying to take care of," to start. As a contrasting point, Seth found out his original project goal was "easier than [they] thought it would be so [they] kind of just kept adding things to make […] it feel like an actual experience that you would remember." For Seth, the priority was to make his hackathon project a memorable experience.

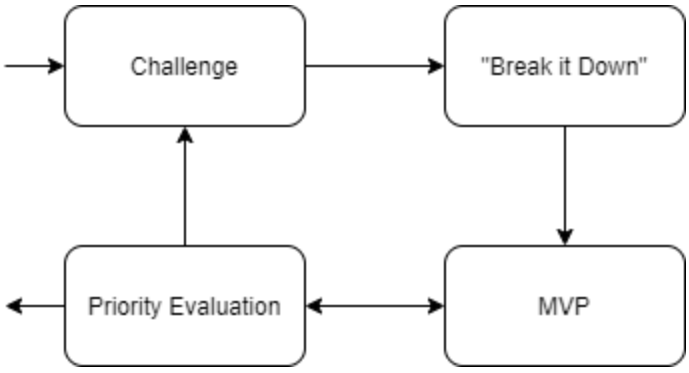

Figure 1. Preliminary Process Model

Preliminary Model

The preliminary model in Figure 1 shows the problem-solving process that the participants employed in both projects. First, they identified a challenge to tackle. Next, they broke down what it would take to resolve the challenge. Third and fourth, they cycled between checking what needed to be accomplished to achieve their MVP and what the current constraints of the product were. At times, they may run into new challenges, thus starting the overall cycle again, or have succeeded, and move on to the next challenge in the project. There is also the possibility that they have addressed all challenges or have completed the project, therefore ending the cycle.

Conclusions and Next Steps

In this work in progress pilot study, we have leveraged adaptive expertise in two unique project development environments. We have identified some preliminary themes about how students approach structured projects such as capstones, and messy projects such as hackathons. These students employ formal and informal uses of software process such as requirements development and various diagramming methods. In both environments, students broke down the challenges they faced into manageable parts and referred to their MVP to guide feature development and decision making. The constraints of each environment require participants to constantly evaluate the state of their project and make adaptations as problems arise.

The next steps for this project involve conducting analysis using a more rigorous set of coding methods: process coding, and versus coding (Saldaña, 2012). Process coding will highlight specific techniques and methods participants use to problem solve, debug, plan, design, and work on their development projects. As a result of using an artifact elicitation interview, we will be able to understand the subconscious decisions and processes behind the artifact's creation. To better draw comparisons between the two environments, versus coding between hackathons and capstones will be used. Similarly, pattern coding will address similarities between the two environments in greater detail.

This pilot study will then capture the experiences of undergraduate students at ASU still in their capstones and obtain more real-time reflection on projects and work toward achieving theoretical saturation. When considering Walther's concept of theoretically validating the process of *making* the data (Walther et al., 2013), a purposive sampling approach will be limited by the lack of

voice from students who have an incomplete experience or are not from the software engineering program. Therefore, software engineering undergraduate students will be selected across the in-person and online ASU software engineering students who are currently in or have already completed their capstone course. Having both in-person and online students account for differences that could be present in the software engineering bachelor's program.

Finally, innovations continue to change the face of technology, and engineers must rise to meet that challenge. This work opens a conversation on how to support student development outside the classroom. Not all classrooms have the opportunity or freedom to teach adaptive expertise to students. Faculty are already working to develop students into adaptive experts, but non-classroom experiences may provide a supplemental benefit toward this goal.

References

ASU Software Engineering Major Map. (n.d.). Arizona State University. https://webapp4.asu.edu/programs/t5/roadmaps/ASU00/TSSERBS/null/ALL/2020

Birbeck, N., Lawson, S., Morrissey, K., Rapley, T., & Olivier, P. (2017). Self Harmony: Rethinking Hackathons to Design and Critique Digital Technologies for Those Affected by Self-Harm. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, 146–157. https://doi.org/10.1145/3025453.3025931

Briscoe, G., & Mulligan, C. (2013). *Digital Innovation: The Hackathon Phenomenon*. http://www.creativeworkslondon.org.uk/wp-content/uploads/2013/11/Digital-Innovation-The-Hackathon-Phenomenon1.pdf

Calco, M., & Veeck, A. (2015). The Markathon: Adapting the Hackathon Model for an Introductory Marketing Class Project. *Marketing Education Review*, *25*(1), 33–38. https://doi.org/10.1080/10528008.2015.999600

Douglas, E., Jordan, S., Lande, M., & Bumbaco, A. (2015). Artifact elicitation as a method of qualitative inquiry in engineering education. *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference and Exposition*, 26.235.1-26.235.10. https://doi.org/10.18260/p.23574

Eyerman, S., Hug, S., McLeod, E., & Tauer, T. (2018). *Uncovering K-12 Youth Engineering Design Thinking through Artifact Elicitation Interviews*. 11.

Flores, M., Golob, M., Maklin, D., Herrera, M., Tucci, C., Al-Ashaab, A., Williams, L., Encinas, A., Martinez, V., Zaki, M., Sosa, L., & Pineda, K. F. (2018). How Can Hackathons Accelerate Corporate Innovation? In I. Moon, G. M. Lee, J. Park, D. Kiritsis, & G. von Cieminski (Eds.), *Advances in Production Management Systems. Production Management for Data-Driven, Intelligent, Collaborative, and Sustainable Manufacturing* (Vol. 535, pp. 167–175). Springer International Publishing. https://doi.org/10.1007/978-3-319-99704-9_21

Gama, K. (2017a). Crowdsourced Software Development in Civic Apps—Motivations of Civic Hackathons Participants: *Proceedings of the 19th International Conference on Enterprise Information Systems*, 550–555. https://doi.org/10.5220/0006377005500555

Gama, K. (2017b). Preliminary Findings on Software Engineering Practices in Civic Hackathons. *2017 IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*, 14–20. https://doi.org/10.1109/CSI-SE.2017.5

Gama, K., Alencar Gonçalves, B., & Alessio, P. (2018). Hackathons in the formal learning process. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*, 248–253. https://doi.org/10.1145/3197091.3197138

Gary, K. (2015). Project-Based Learning. *Computer*, *48*(9), 98–100. https://doi.org/10.1109/MC.2015.268

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Transaction Publishers.

Hatano, G., & Inagaki, K. (1986). Two Courses of Expertise. In *Child development and education in Japan* (pp. 262–272). H. Stevenson, H. Azuma, & K. Hakuta (Eds.), New York: Freeman.

Komssi, M., Pichlis, D., Raatikainen, M., Kindstrom, K., & Jarvinen, J. (2015). What are Hackathons for? *IEEE Software*, *32*(5), 60–67. https://doi.org/10.1109/MS.2014.78

La Place, C., Jordan, S., Lande, M., & Weiner, S. (2017). Engineering Students Rapidly Learning at Hackathon Events. *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference and Exposition*.

Linnell, N., Figueira, S., Chintala, N., Falzarano, L., & Ciancio, V. (2014). Hack for the homeless: A humanitarian technology hackathon. *IEEE Global Humanitarian Technology Conference (GHTC 2014)*, 577–584. https://doi.org/10.1109/GHTC.2014.6970341

McKenna, A. F. (2007). An investigation of adaptive expertise and transfer of design process knowledge. *Journal of Mechanical Design*, *129*(7), 730–734.

McKenna, A., Linsenmeier, R., & Glucksberg, M. (2008). *Characterizing Computational Adaptive Expertise*. 13.288.1-13.288.11. https://peer.asee.org/4415

Mtsweni, J., & Abdullah, H. (2015). Stimulating and maintaining students' interest in Computer Science using the hackathon model. *The Independent Journal of Teaching and Learning*, *10*(1), 85–97.

Paay, J., Sterling, L., Vetere, F., Howard, S., & Boettcher, A. (2009). Engineering the social: The role of shared artifacts. *International Journal of Human-Computer Studies*, *67*(5), 437–454. https://doi.org/10.1016/j.ijhcs.2008.12.002

Pedra, E. (2019, September 3). *Hackathon Demographics—Who's Going to MLH Hackathons?* Major League Hacking News. https://news.mlh.io/mlh-hackathon-demographics-09-03-2019

Peng, X., McGary, P., Ozturk, E., Yalvac, B., Johnson, M., & Valverde, L. M. (2014). Analyzing Adaptive Expertise and Contextual Exercise in Computer-Aided Design. *Computer-Aided Design and Applications*, *11*(5), 597–607. https://doi.org/10.1080/16864360.2014.902693

Porras, J., Khakurel, J., Ikonen, J., Happonen, A., Knutas, A., Herala, A., & Drögehorn, O. (2018). Hackathons in software engineering education: Lessons learned from a decade of events. *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials - SEEM '18*, 40–47. https://doi.org/10.1145/3194779.3194783

Richard, G. T., Kafai, Y. B., Adleberg, B., & Telhan, O. (2015). StitchFest: Diversifying a College Hackathon to Broaden Participation and Perceptions in Computing. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15*, 114–119. https://doi.org/10.1145/2676723.2677310

Roberts, Chell, Morrell, D., Henderson, M., Danielson, S., Hinks, R., Grondin, R., Sugar, T., & Kuo, C.-Y. (2007, June). *An Update On The Implementation Of A New Multidisciplinary Engineering Program*. 2007 Annual Conference & Exposition, Honolulu, Hawaii. https://peer.asee.org/2964

Saldaña, J. (2012). *The coding manual for qualitative researchers*. Sage.

Schwartz, D. L., Bransford, J. D., Sears, D., & others. (2005). Efficiency and innovation in transfer. *Transfer of Learning from a Modern Multidisciplinary Perspective*, 1–51.

Swift, M. (2019, September 30). *Major League Hacking (MLH) is coming to APAC!* Major League Hacking News. https://news.mlh.io/mlh-is-coming-to-apac-09-29-2019

Vanasupa, L., Stolk, J., & Harding, T. (2010). *Application of Self-Determination and Self-Regulation Theories to Course Design: Planting the Seeds for Adaptive Expertise*. *26*(4), 914–929.

Walther, J., Sochacka, N. W., & Kellam, N. N. (2013). Quality in Interpretive Engineering Education Research: Reflections on an Example Study: Quality in Interpretive

Engineering Education Research. *Journal of Engineering Education*, *102*(4), 626–659. https://doi.org/10.1002/jee.20029

Warner, J., & Guo, P. J. (2017). Hack.edu: Examining How College Hackathons Are Perceived By Student Attendees and Non-Attendees. *Proceedings of the 2017 ACM Conference on International Computing Education Research - ICER '17*, 254–262. https://doi.org/10.1145/3105726.3106174