ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #37984

Adapting Hackathon-Honed Skills Toward Software Engineering Capstone

Cecilia La Place (Arizona State University, Polytechnic Campus)

Cecilia La Place is a third-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 organizes, attends, and studies hackathons as informal learning environments that hold the potential to empower students of any and all backgrounds.

Shawn S. Jordan (Associate Professor)

Shawn Jordan 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. Jordan is PI on several NSF-funded projects related to design, including an National Science Foundation (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. Jordan co-developed the STEAM Labs™ program to engage middle and high school students in learning science, technology, engineering, arts, and math concepts through designing and building chain reaction machines. He founded and led teams to two collegiate Rube Goldberg Machine Contest national championships, and has appeared on many TV shows (including Modern Marvels on The History Channel and Jimmy Kimmel Live on ABC) and a movie with his chain reaction machines. He serves on the Board of the i.d.e.a. Museum in Mesa, AZ, and worked as a behind-the scenes engineer for season 3 of the PBS engineering design reality TV show "Design Squad." He also held the Guinness World Record for the largest number of steps – 125 – in a working Rube Goldberg machine.

Adapting Hackathon-Honed Skills Toward Software Engineering Capstone Courses

Abstract

Coding marathons known as *hackathons* are informal teaching and learning environments where students build skills they can use in class, and design innovative projects they can use to pursue industry employment. Though many elements of undergraduate engineering programs facilitate similar outcomes, students still seek out hackathons as a means to obtain more real-world experience. Prior work has shown that students are enticed to attend hackathons to take advantage of independent learning and networking opportunities. Prior work also suggests that hackathons engage students by incorporating a culture of teaching and learning through self-regulated learning, and through sharing and learning with other participants. This prompts the applicability of hackathons in the classroom and inquiry into students' transferable skills development. Exploring this query benefits students who struggle in class by examining alternative engagements and technical development opportunities.

Extending existing work to examine if and how students develop and apply skills developed during hackathons in the classroom, the following research questions guided this study:

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 paper builds upon a previously published work in progress, finding that students who attended hackathons and a project-based learning Software Engineering degree, built transferable skills between hackathons and capstones. Participants described the employment of software design methodologies in both hackathons and capstone projects, various problem solving and researching approaches used, and a distinct level of attention to the context of development, informal or formal, to determine their courses of actions. Leveraging adaptive expertise as a lens to understand the development of learning mechanisms and problem-solving processes, we position hackathons as a valuable supplement to Computer Science and Software Engineering coursework as it engages students in project development, problem-solving, and learning through dynamic constraints.

Introduction

Engineering is evolving alongside the changing landscape of technology and demands of society. It follows that engineering education must also adapt to better prepare students, future engineers, for the challenges inevitably ahead. In 2004, the Engineer of 2020 became the guiding manual for educators and administrators to shape the academic expectations present in today’s engineering classrooms (National Academy of Engineering, 2004). Adaptable engineers capable of solving complex problems never seen before are the ideal outcome of engineering programs.

However, classrooms do not support the influx of students' diverse ways of learning and knowing. Students are seeking additional experiences outside the classroom finding that classroom instruction is insufficient. Despite this endeavor being an expectation of engineering students, the motivation is cause for concern. As classrooms have begun to move away from the traditional pedagogy of learning by rote, we ask if current and traditional classrooms engage students to apply their externally gained knowledge within the classroom. Capstone is one of the few if only courses positioned to do so. Framed as a culminating experience, capstone is often the only exposure to "real world" problems. Capstone may likely be the only opportunity to apply knowledge gained in and outside of the classroom. Meanwhile, hackathons–coding marathons–support students seeking additional learning experiences. They are free financially and of academic responsibility, inspiring students to design solutions to problems they identify–including their own learning gaps.

The intersection of the outcomes of hackathons and capstones highlight the need for adaptability to be part of the development of engineers. Students currently develop this indirectly through their coursework and experiences, but classes do not directly facilitate this skill. Furthermore, classes do not often draw upon students' external experiences, making it difficult for engineering students to combine their sources of knowledge and deem it valuable. This in turn affects the transferability of skills from external to formal, and vice versa.

This work follows up on a pilot study that examines the project-based learning trained students of a software engineering program who additionally attended a hackathon. Capstones are effective tools in training soon-to-be engineers to be adaptable. Capstones alone should not be the only answer or exposure to solving real world problems for student engineers, particularly as unseen and unknown challenges continue to appear with the growth of our society. Hackathons exhibit many of the same features as capstones under a similar guise of real world experience, such as team work, project scoping and development, problem solving and critical thinking, working with constraints. Hackathons are poised to be a supportive element in the development of engineers as long as transferable skills are effectively utilized within classroom settings.

Literature Review

Hackathons

Despite the word hackathon invoking a negative reaction associated with hacking – the act of breaking into technology – hackathons themselves do not promote invasive activities. Hackathons are time constrained prototyping marathons, where students motivated to learn new technologies and network with industry professionals gather for a lively weekend of technological development (Briscoe & Mulligan, 2013). Here, they design and implement prototypical solutions (often software, though sometimes hardware) for social problems, to compete for prizes, or to engage in a new technology (Warner & Guo, 2017). Hackathons can be leveraged in various contexts, such as for course projects, for social good, or even to promote learning programming and other technical skills (Birbeck et al., 2017; Gama et al., 2018; Linnell et al., 2014; Richard et al., 2015).

For the purposes of this research, hackathons in this context refers to collegiate weekend events that predominantly engage computer science and software engineering students–though there is

often a smaller population of other disciplines in attendance. These events are typically supported by a larger non-profit organization known as Major League Hacking (MLH). Events under this organization have a consistent event structure. Events begin with an opening ceremony introducing the event, expectations, potential prize categories (or development pathways), sponsors and partners, and ways to access resources (MLH, n.d.). These hackathons can be general, where they are open to everyone of any skill level, or specialized where only specific genders, specific technical topics, or skill levels are allowed. Hackathons supported by MLH are generally 24 - 36 hours long, provide mentors, speakers, and activities throughout the event, and also provide on site food and lodging for students in attendance (MLH, n.d.). Participants spend the weekend engaging with hosted events and resources, culminating in a final demonstration at the end of the event of the projects created, prize winner announcements, and a formal closing of the event.

Past research on collegiate hackathons has labeled them as external learning environments for students seeking hands-on learning. Students are driven to find opportunities outside the classroom to supplement their degree. Though limited to beginner hackathon contexts, small-scale qualitative studies have found that hackathon attendees engage in a culture of sharing and learning not emphasized in the classroom (La Place et al., 2017). Other research has likened hackathons to project based learning environments and exemplifying desired engineering outcomes (Horton et al., 2018; La Place & Jordan, 2020). As hackathons engage students outside the classroom, questions remain as to what is transferable back to the classroom and when.

Adaptive Expertise

An adaptive expert is one who can navigate their existing knowledge to resolve their lack of knowledge, particularly when solving a problem. This work draws upon adaptive expertise as framed by Schwartz, Bransford, and Sears (2005) which builds upon Hatano and Inagaki's types of expertise development: routine expertise and adaptive expertise (Hatano & Inagaki, 1986). Routine expertise develops over time as individuals become more proficient at a task (Hatano & Inagaki, 1986). However, if an individual solely becomes a routine expert, they may find themselves unable to apply the skills of their routine task elsewhere (Hatano & Inagaki, 1986). Adaptive expertise is developed slower, but with an open mindset toward finding new methods of resolving tasks (Hatano & Inagaki, 1986). Schwartz et al. (2005) build on these ideas in broader educational settings, noting that procedural exercises in classrooms do not build adaptive expertise, but routine expertise.

Adaptive expertise is a suitable framework to compare capstones and hackathons. Both contexts involve participants selecting unique projects from their peers, thus removing the risk of studying procedural tasks. Furthermore, the projects are often open ended, resulting in participants adapting to the needs of their projects. The dynamic nature of these projects present a unique opportunity to observe knowledge transfer throughout the development of students' adaptive expertise. The following research questions guided this study:

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?

Methods

Participants

To answer these research questions on a pilot study basis, members of the software engineering degree program at the author's home university were reached out to. Undergraduate students in their capstone who had participated in at least one hackathon within a year were ideal candidates for this study. Thus, to access these students, the researcher contacted the program's capstone professor via email and requested they distribute the participant call to their students. However, no students responded to the call. As a result, the researcher then used purposive sampling by emailing students who graduated from the software program at most 2 years prior to the time of the research study being conducted and had attended at least one hackathon within 1-2 years of beginning their capstone. Of the nine (9) graduated students contacted, seven (7) responded, and five (5) were selected due to meeting the requirements and having availability. All participants were part of the same graduating class and had teamed up with at least one participant in a hackathon and/or their capstone. Due to the small sample size, demographics were not collected to preserve the anonymity of the participants.

Data Collection

The data used in this work were collected for a previously published analysis by the authors (La Place & Jordan, 2020). Semi-structured interviews were conducted on Zoom or in-person based on availability over the course of an hour. The interviews consisted of two parts, one focusing on their recent hackathon project, and the other focusing on their capstone project. However, both parts consisted of the same or similar questions to simplify the interview and analysis processes. A selection of interview questions and their corresponding research question connection can be seen in Table 1. This resulted in two remote interviews over Zoom, and three in-person interviews. Despite these mediums, only audio was recorded and retained from the interviews, which were then transcribed manually and sent back to the participants to revise their responses and correct transcription errors.

Sample Interview 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

If you were to add a new feature to your project, what would it be and how would you do so?	1, 3

Table 1. Sample interview questions used for capstone and hackathon project elicitation.

Data Analysis

This paper relied upon a deductive analysis of the data to complement the previous paper's inductive analysis. The deductive process consisted of using process coding for the first round of coding. The adaptive expertise (Schwartz et al., 2005) framework was operationalized as a deductive codebook shown in Table 2 below. This work echoes the approach taken by Larson et al. (2020) which also began with inductive analysis and concluded with a deductive analysis to address the weaknesses in each method by strengthening and extending the existing theories. Their work focused only on a junior engineering project course, whereas this work examines students' experiences who attended hackathons and a project based curriculum. Sample codes can be found in Table 2. Consolidating the process codes in a second round consisted of using versus coding as an additional method of comparing capstones and hackathons.

Code	**Definition**
Designing	Designing refers to when participants create specifications for a project.
Problem Solving	Problem Solving refers to when participants are describing specific techniques to approach and resolve a problem, issue, bug, etc.
Researching	Researching refers to when participants meet a knowledge gap and work to address it

Table 2. Deductive codes.

Results

Designing Solutions

Depending on the project type, participants described different design processes. For capstone projects, participants participated in a formal design process known as Agile, a process frequently taught in their classes. Meanwhile, hackathon projects consisted of informal design processes that used selections of formal design processes, such as requirements design, but often repurposed them to adapt to the hackathon environment. In capstone, participants noted they were expected to follow a strict agile approach as defined by their classroom experiences by the capstone professor. This included creating diagrams such as use cases, class diagrams, state diagrams, requirements elicitation and development. Participants also described situations involving potential users as a use case and user centered design idea. In hackathons these diagrams and requirements were not often used. Rather, diagrams were used loosely to help address a specific need, such as business model logic in participant Mark's project, or were not rigorous with their nomenclature as participant Frankie mentions. Requirements during

hackathons were often spoken requirements and were used to define interactions between subsystems or as goals for the project's MVP, which participants Mark and Porter respectively describe in their interviews. The results show a stark contrast between the application of design principles and methods due to the needs of the environments.

Customized Problem Solving

Participants employ a variety of techniques to approach problems. Participant Alex uses the same problem-solving approach in both capstones and hackathons. They break down the problem into stepwise approaches of researching examples or possible solutions, then they try the methods and sometimes rely on mentors or friends to help them with things far beyond their experience. They aim to "understand the best way of doing that [activity]." However, participant Porter describes a different approach. They ask themselves "what if X" and other scenario questions that lead to a trial and error approach to resolving the issue. Participant Porter also uses a break-it-down (into smaller pieces) method to make difficult tasks more accomplishable. Participant Seth's problem-solving process was also trial and error, but it was only in hackathons that he used that approach. During capstone he mentions only "reading up" on relevant information.

Research Sources

Knowledge gaps were a frequent problem for participants that was often solved through research. Research took the form of visiting Google and StackOverflow, but also consisted of looking up example code, tutorials, and videos. At times, it also meant contacting friends with greater knowledge of the topic, utilizing online communities, and interacting with mentors. Though research happened in both project environments, the availability of research resources varied from project to project. Though all participants did online research, some used mentors, some asked friends, and some relied on videos and tutorials. For example, participant Alex did not know mentors were available at the hackathon they attended and relied on tutorials and examples found online. However, for their capstone, they relied on their friend who had greater knowledge of the software Docker, which standardizes software environments for development, testing, and deployment. Participant Porter relied on a friend and mentors during their hackathon for various topics. Participant Seth often found himself Googling examples and reading about the topic.

Informal vs. Formal Development

Between the inductive analysis (La Place & Jordan, 2020) and the deductive analysis, participants indicated both directly and indirectly their formal and informal uses of software development process. In all their capstone experiences, participants noted their need to follow a strict agile format as a result of the class requirements. This led to accurate software diagrams, requirement elicitation, and clear task designations and breakdowns. However, in hackathons, participants either simplified their use of these methods, or simply did not use them at all. Rarely if ever did participants do testing during a hackathon outside of user testing. Yet, in capstone projects, testing was limited as well. Similarly, maintainability was completely ignored during the project, but given the opportunity to return to the project all said maintainability or more user-oriented features would be added. Requirements in hackathons were often only verbal, or to assist with future integrations between sub-teams. Diagramming was rarely used, except in the case of participant Frankie, who now uses simplified diagramming to create a plan of action for

he and his team if they have an idea for a hackathon project upon arrival. It appeared that the participants had a toolset of software processes to leverage at any given time but chose how and to what extent to use those tools depending on the project and the expectations of the project that they were working on.

Discussion and Implications

Adaptive Expertise

According to Schwartz et al. (2005), an adaptive expert must balance innovation and efficiency. Through this study, we have interviewed students who have participated in a program that prioritizes providing experiences that teach them how to "arrive at a solution" (Gary, 2015). A caveat to studying adaptive expertise in engineering is that students are not often prepared to enter an adaptive experience (their capstone) due to their course structure. We believe that the project-based learning structure in the ASU Software Engineering program exposes students to adaptive thinking. By also having various amounts of short-form adaptive experiences through hackathons, these students have been forced to optimize their problem solving and learning abilities that have been developed through their schooling. Coming to a solution is not often a linear or repeatable task, but the process of doing so is–based on these students' methods. These students were able to describe their preferred way to problem solve in capstone and hackathons respectively. For some, they used the same break-it-down methodologies, while others relied on research and social connections to supplement their learning. The combination of these approaches are similar to the ideas presented in Schwartz et al. (2005) that adaptive experts are able to break down problems into routine problems and problems that they need to gain more knowledge to solve.

Participants adapted their use of software techniques to the situations. In capstone, participants described spending a lot of time doing rigorous planning and coding. In hackathon settings, participants either lightly planned or only planned for very specific needs to ensure they had the most amount of time to develop their MVP for their demonstrations. In hackathons, the need for innovative solutions often came about due to the time constraint, but when they had knowledge of software, libraries, or even planning approaches, participants were able to pick and choose what would best suit the needs of the project or problem they were working on. In capstones, participants were innovative to meet time constraints again, but also at times personal goals. Mark, for example, wanted to use open source libraries in his capstone project and tried numerous methods to meet this goal before time became a constraining factor once more. Adaptive expertise as a framework on a more generalized population at hackathons (such as STEM) may result in different domain-specific techniques being applied during hackathons. This idea is suggested by McKenna et al. (2007) as they highlight in their future work: application of adaptive expertise studies in different disciplines through praxis of "design process knowledge, disciplinary knowledge, and metacognition."

What technical knowledge do students use in capstones and hackathons?

In Gama's (2017) research on software skills at civic hackathons, they found that it depended on the skills of the group and the quality varied. Gama found that participants were their own stakeholders, that software architecture was not a main topic, and that "project tasks were defined and assigned as the project evolved." This research affirms many of the conclusions

Gama drew. Participants at hackathons did not often discuss the architecture and were their own stakeholders or had to rely on each other when stakeholders were unavailable. Project tasks were taken on by the requirements decided by the group. Based on the discussions with the participants, some projects have a clear MVP, like participant Mark and his teammate's rock paper scissors application. However, some projects develop over time, like participants Porter and Alex's hackathon projects. Similarly, testing was minimal and only to ensure the MVP worked, but provided the motivation and opportunity to return to the project. Participants often expressed a desire to focus on maintainability. In the case of participant Frankie's hackathon experience, he often uses diagramming to break down the needs of his MVP, but he does not use diagramming to the structured extent that was expected in his capstone. As time is a constraint and there are no formal evaluators, the need to be strict and orderly is severely reduced.

Predominantly, the participants were leveraging software development lifecycle (SDLC) skills. SDLC skills commonly consist of creating an idea, developing requirements, designing the architecture and interactions, coding, testing, and deployment and maintenance (Ruparelia, 2010). However, the participants focus primarily on idea generation in hackathons, not in capstones. Otherwise, design and requirements can vary between the two environments and by each participant's team and approach. Testing and maintenance, however, are limited in their use. Testing in hackathons is often to ensure the code works within minimal bounds whereas testing in capstones is to ensure the MVP would be functional for the demonstration. Participants often situated themselves in agile (Birbeck et al., 2017) in capstones and an iterative development (incremental) (Ruparelia, 2010) cycle in hackathons. Though this work extends Gama's, it would be beneficial to dig more deeply into how participants in hackathons know when to cut corners in software life cycle skills and when not to based on their overall software development skills.

In terms of specific technologies, the participants found themselves working with simulations, either in virtual web spaces or with virtual and augmented reality. As indicated by the participants in their interviews, they found themselves learning C# in Unity, Javascript, third party libraries, and more in both hackathons and capstones. From these findings especially, leveraging capstone and hackathon experiences might lend to understanding motivations behind decisions to enter graduate school or the workforce.

Where do students learn the knowledge used in capstones and hackathons?

According to many of the participants, they often broke tasks down and researched what they needed. Research often came in the form of relying on friends with more expertise, mentorship from sponsors or stakeholders, or reviewing tutorials and online forums for specific fixes to their code where applicable. At times, their experience with a library or concept made an appearance in the other environment or drove an interest in exploring a concept more. This picks up from where La Place et al. (2017) left off in their knowledge transfer work. Though some participants ended up on similar teams at different instances, knowledge was still being shared and acquired from sources within and outside an environmental context. However, most of the time participants used the resources available (mentors and the Internet) if they did not have access to other resources. This also aligns with self-regulated learning, as studied previously in La Place et al. (2017), affirming that hackathons enable students to digest new material. Combined, these events and environments contribute to the goals of engineering education to engage students in learning to learn strategies that enable life long learning (Cornford, 2002).

As a result of their exposure to various technologies, some participants went on to pursue higher education or jobs in similar fields, while others were able to apply knowledge from one experience to another. In other instances, participants had developed their own problem-solving methods that they then carried into both experiences as a method to handle challenges they were faced with. Many, if not all, of their software processes came from their schooling, but discrete coding techniques and methods were developed as a result of their own project experiences in capstone where they were required to have good code quality. More research should be done on the experiences and events in which participants found a problem-solving approach that worked for them. These instances may not be situated in hackathons and capstones, but possibly even in work environments, school projects, or other external learning environments.

Conclusions and Next Steps

Exposing students to more opportunities to become adaptive engineers is necessary. Through the combination of capstones and hackathons, the participants interviewed in this study were able to adapt and transfer skills based on the constraints of their projects. From the deductive analysis, we learned that students were employing SDLC skills in various levels of formality that depended on the requirements of the project and environment, and that participants carried the same problem-solving skills into both capstones and hackathons. However, testing and maintainability were the two least used skills in both projects, but given the opportunity to continue with either projects, many participants expressed a desire to address this. Though their software skills came from their schooling, their coding, learning, and problem-solving abilities were developed as a result of their experiences in both environments.

Future studies must explore this phenomenon on a larger scale across more universities and pedagogical methods such as problem based learning and traditional learning. This work did not consider the aspect that students who have other responsibilities are unable to participate in hackathons or pursue other external learning experiences. Comparing the experiences of those who are able and unable to pursue external learning experiences as they reach their culminating experiences is critical to adapting engineering curriculums to suit more students. Developing adaptable engineers will require more insight delving into the adaptability and transferability within the classroom.

References

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

Cornford, I. R. (2002). Learning-to-learn strategies as a basis for effective lifelong learning. *International Journal of Lifelong Education*, *21*(4), 357–368. https://doi.org/10.1080/02601370210141020

Gama, K. (2017). 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

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.

Horton, P., Jordan, S., Lande, M., & Weiner, S. (2018). Project-Based Learning Among Engineering Students During Short-Form Hackathon Events. *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference & Exposition*.

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*.

La Place, C., & Jordan, S. S. (2020, June 22). *WIP: Building a Bridge Between Hackathons and Software Engineering Capstones Through Adaptive Expertise*. 2020 ASEE Virtual Annual Conference. https://peer.asee.org/wip-building-a-bridge-between-hackathons-and-software-engineering-capstones-through-adaptive-expertise

Larson, J., Jordan, S., Lande, M., & Weiner, S. (2020). Supporting Self-Directed Learning in a Project-Based Embedded Systems Design Course. IEEE Transactions on Education, 63(2), 88-97.

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.

MLH. (n.d.). Introduction to organizer guide. Introduction - Hackathon Organizer Guide. Retrieved February 16, 2022, from https://guide.mlh.io/

National Academy of Engineering (Ed.). (2004). *The engineer of 2020: Visions of engineering in the new century*. National Academies Press.

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

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13. https://doi.org/10.1145/1764810.1764814

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.

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