Innovative Practice: Agile Training for Year-Long Capstone Project

William Eberle
Department of Computer Science
Tennessee Tech University
Cookeville, TN, USA
weberle@tntech.edu

Gerald C. Gannod
Department of Computer Science
Tennessee Tech University
Cookeville, TN, USA
jgannod@tntech.edu

Eric Brown
Department of Computer Science
Tennessee Tech University
Cookeville, TN, USA
elbrown@tntech.edu

***Abstract*— This innovative practice full paper describes a two-semester capstone experience that trains students in agile software engineering principles and incorporates the material into building an actual product for an industry partner. Recently, teaching agile software engineering has garnered considerable attention, and research has focused on effective pedagogical approaches, challenges, and outcomes. However, while computer science students are exposed to agile methodologies in their curriculum, and students even use the approach in a project, the experience tends to be brief and non-real-world. In this work, we believe our approach provides a more cohesive learning experience, better prepares students for jobs in industry, and introduces them to incorporating an overall agile mindset. We outline specific activities, timelines, and best practices for managing team projects and providing a better experience for the students. The results of our efforts are reported through retrospectives and reflections with the students over five years.**

Keywords—agile, software engineering, capstone

I. INTRODUCTION

Agile software engineering methodologies revolutionized the software development landscape in industry over 20 years ago by fostering adaptive, collaborative, and iterative approaches to product development. Initially outlined in the Agile Manifesto [1], its principles emphasize customer collaboration, flexibility to change, frequent delivery of working software, and continuous improvement. Organizations across diverse sectors, such as technology, finance, healthcare, and more, have increasingly embraced Agile due to its ability to enhance project management, streamline development processes, and deliver high-quality products efficiently. Recent studies continue to affirm Agile's effectiveness in improving project success rates, with research by Deloitte indicating that 94% of respondents experienced increased project success rates after adopting Agile methodologies [2]. Additionally, a survey by McKinsey found that 94% of respondents believed Agile transformation provided moderate to significant value in achieving business objectives [3]. Agile's relevance and impact drive innovation, adaptability, and customer satisfaction within the software development industry.

Most undergraduate programs in computer science or software engineering include a senior design or capstone course. In the last 10+ years, many programs have gone towards students forming Agile product development teams that follow the Agile development principles and practices originated in the software industry. Agile teams are organized to be autonomous (with limited external input) and role-specific, with each member lending both specific and general expertise. Designated roles on small Agile teams include Scrum Master (team leader or coach), team members, and product owners. Sometimes, the latter role will be external consultants drawn from experts in institutional and learning assessment in higher education. Agile teams are organized to integrate frequent feedback with an emphasis on working toward maximum flexibility and efficiency. Many features of Agile product development mirror best practices in teaching and learning,[4] including frequent feedback, group coordination, and creative problem-solving [2], [3], [5], [6], [7], [8], [9], [10], [11]. In short, students who work in Agile teams gain experience in methodologies they are likely to encounter in the workplace.

In this work, we present an approach implemented at our institution for not only training students in Agile software engineering principles but also incorporating the material into a mini project, followed by 1.5 semesters of building an actual product for an industry partner. While, in general, many of the techniques and tools that faculty and students use are not unique to our institution, to the best of our knowledge, how the course(s) is laid out *is* unique and could be instructive to others. The remainder of this paper is organized as follows. Section II describes background and related work. Our approach for the capstone experience is presented in Section III. Section IV describes several best practices that we have employed in our offerings, and Section V draws conclusions and suggests future investigations.

II. RELATED WORK

A. Agile Training

The field of teaching Agile software engineering has garnered considerable attention in recent research, focusing on effective pedagogical approaches, challenges, and outcomes. This attention reflects the influence of external industry advisors attesting to what has happened in the workplace for the last two decades. Processes must become leaner and more dynamic to retain and improve productivity and profitability. This leanness must be demonstrated in both hardware and software

development processes. In correlation with this change, software engineering instruction must also change to facilitate the lean, dynamic development cycle mindset [12].

B. Software Engineering Capstones

Along with the push for general Agile training and incorporation of its principles into general computer science instruction, software engineering capstones had to make substantial changes or risk releasing an ill-prepared cohort of new professionals into a market that has technologically left them behind. Capstone updates required more than an introduction to the manifesto and a statement of "this is a good idea." Agile disciplines had to become systemic to the instruction process itself, not merely a discussed topic. As discussed in the principles, we must lead and teach by example – an active agent perpetuating dynamic processes in instruction and implementation [13].

C. Project Formats and Faculty Workload

One of the larger concerns regarding capstone projects is overall project management. A software engineering capstone project requires regular management, unlike traditional courses where the faculty lectures for just the class period and involvement is limited to grading and some office hours. As far back as 20 years ago, one can find reports discussing the issue of faculty workloads when running capstone courses [14]. Supervising small projects requires a significant amount of time and attention from the instructor. However, just like the "Mythical Man-Month" issue [15], adding more instructors to the course does not solve the problem, as there becomes the issue of quality control (e.g., consistent grading). Some institutions use graduate students to supervise projects while the instructor provides lectures and instruction. In addition, with external projects, industry partners can also provide some supervision. Yet, while faculty could just provide instruction, students typically want more interaction and answers to detailed questions.

In the work of Goold [16], he prefers a small number of project supervisors whom the course instructor monitors. In his approach, the groups are "essentially self-managing," and supervisor involvement is minimal. However, he does report that when the scope of a project changes or the customer changes details, students struggle with managing the project. Thus, other than making teams self-manage – which can be problematic – lessening the faculty workload is not directly addressed.

In Holmes et al. [17], they present an approach where students are members of a distributed software project managed by a Canadian program called Undergraduate Capstone Open Source Projects (UCOSP). A steering committee of faculty members administers the program to provide real-world development experiences that do not require a large amount of administrative overhead. Institutions can sign up for (and provide funding for) an open-source project that their students can work on, where the majority of interaction takes place online. The UCOSP steering committee is responsible for all organizational aspects of the program, including paying for student travel, hotel, and meals (participants are responsible for their other expenses). The primary tasks of the home faculty are to recruit and screen applicants to the program, to ensure that once accepted, their students are putting forth the required effort, and to handle administrative details at their home institution, including submitting a final grade. However, while several aspects would be an interesting experience, this is a Canadian-only program with a cost. For most institutions, the administrative effort and cost to set up something similar would be prohibitive.

In Bastarrica et al. [18] they present an approach very similar to the one presented in this work, where the primary focus is on the students' soft skills – something they deem to be much more of a factor in the success of a project. Students work in teams of four to seven people, grad students act as tutors, and the instructor selects the projects, coordinates and monitors all teams, intervenes when major problems emerge, and evaluates them at the end of each iteration. However, in their approach, students are expected to work 16 hours a week (for one semester), and customers pay "a modest amount of money." In addition, other than the use of grad tutors, there are no other suggestions for helping with faculty workload.

Fan's study [19] focuses on agile management of team-based capstone projects. Again, similar to the approach presented in this paper, their capstone course(s) spans over two semesters, and students work on projects from industrial customers. However, there are two roles: (1) the faculty advisor who closely monitors the progress of a team on their projects, reviews the team's project work, and provides advice for improvement; and (2) the course instructor who oversees all capstone projects to ensure that standards are consistently applied. In their approach, they focus on the team's awareness of time through a capstone management system, which also provides reports, Kanban boards, etc. The primary advantage of their dashboard is one central place for many Agile technique tools: GitLab, Kanban, etc. Another advantage of the dashboard is perhaps no need for the students to produce a report (all the information is viewable by the grader). The disadvantages of their dashboard are that it is also the communication platform that students must use (and they prefer social media, e.g., Discord), and not everything most students would need is available. Another interesting aspect of their course is that teams can kick out underperforming team members. In short, the tool/dashboard provides a single repository, but it is unclear how this helps with faculty workload.

Finally, in the work of Broman et al. [20], they have an approach that gives students a more large-company-based feel (but not an actual customer) regarding organization, process, and communication. However, not every student gets the same experience – some are developers, some architects, some testers, etc. In addition, their capstone course is interdisciplinary with engineers, computer science, and business management. They also used TAs as supervisors, including up to four for one cohort. Unfortunately, one of the noted drawbacks was that some students could hide.

III. Innovative Approach

The overall objective of our proposed approach to software engineering and a senior capstone is to provide a more cohesive learning experience, better prepare our students for jobs in industry, and introduce them to incorporating an overall agile mindset into any task. The following lays out the path for this two-semester sequence.

A. *Agile Software Engineering (Semester 1 – Weeks 1-8)*

The goal of the first half of the first semester is to introduce Agile software engineering and an understanding of various aspects of software analysis and design in a manner similar to short courses taught to new developers in software development organizations. The primary Agile development process we use is based on Scrum [21] and includes steps such as concept initiation, backlog construction, iterations, daily standups (or scrums), and iteration releases.

The following content is a list of topics that we teach that are similar to what one would find in a Certified Agile Professional course (e.g., ICAgile, Scrum Alliance, and PMI). The topics include:

- Game-storming [22]
- Scrums and Sprints [23]
- Project and Social Contracts
- Individual/Interactions
- Concept Initiation and Stakeholders
- Prioritization
- Value-Driven Development
- User Stories
- Iteration and Release Planning
- Sprint Execution

Additionally, we introduce other software engineering concepts that are not unique to Agile, such as:

- Conflict Resolution
- Software Architectures
- Testing
- Secure Software Development

A mid-term exam and weekly (online) quizzes are focused on general knowledge and understanding of the aforementioned topics.

B. *Mini-Project Labs (Semester 1 – Weeks 1-8)*

Each week consists of two team-based labs to provide experience for the concepts/topics discussed. Starting in Week 2, self-selected teams of 5-6 students complete different labs during selected class days of the week, ultimately building a mockup of a product. The labs coincide with the topic delivery that week and follow these steps:

1. Project Ideation (e.g., Silly Cow)
2. Game-storming (e.g., Product Box – See Figure 1)
3. Team & Social Contract
4. Project Charter
5. Story Ideation
6. Planning (e.g., Planning Poker)
7. MVP and Release Plan
8. Mockups (e.g., wireframes)
9. Showcase

Sometimes, it is difficult for students even to develop an idea. So, to get them started, we choose a theme that all teams must follow. For instance, at our institution, we are currently in the middle of a 5-year program to reach out and help our surrounding rural community. For example, one year's theme was building an app for helping communities with unemployment issues. This last year, with the rise in popularity of generative artificial intelligence (e.g., Chat-GPT [24]), the theme was "A.I. for Societal Good."

Figure 1 shows a description of one of the lab activities: Product Box. The lab demonstrates a way for developers to gain a better understanding of a customer's perspective on a new product.

C. *Real-World Project (Semester 1 – Weeks 9-15 and Semester 2)*

After the midterm exam (Week 8), the students are put into teams based on their responses to a survey (more about this in Section IV), and a kickoff meeting is held to discuss expectations (Week 9). The team project involves using what is learned in the class towards the requirements analysis and design of a software product for an actual customer. This takes place over two iterations (Iteration 0 and Iteration 1), culminating in an end-of-semester showcase of a mockup to their customer and a team Project Pitch to their classmates.

By the end of the Semester 1, the students will have started to achieve the following learning outcomes [25]:

- An ability to analyze a problem and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An understanding of professional, ethical, legal, security, and social issues and responsibilities.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to function effectively on teams to accomplish a common goal.

Semester 2 consists of five iterations (Iteration 2 – 6), each delivering some product (and eventually a Minimal Viable Product, or MVP) to their customer. The rhythm of an iteration is a kickoff, development, weekly standups with their customer (usually virtually), deliverables, showcase, and retrospective. Also, from a formal grading perspective, there are an end-of-iteration team and individual reports, a video showcase, customer evaluations, and peer evaluations. The semester then culminates with our College's Senior Design Expo where all majors present their work in a science-fair-style type of approach, inviting their customers as well as the entire community.

By the end of Semester 2, the students should be able to demonstrate the following additional learning outcomes:

- An ability to communicate effectively in a variety of professional contexts.
- An ability to function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

IV. Best Practices

In this Section, we will discuss the specific activities and best practices that we have learned to help manage the team projects

Product Box – Object of Play

Before you begin, focus on the end. In this exercise, teams create the physical "box" that sells their idea—whether that idea will ultimately become a tangible product or not. By imagining the package for their idea, the teams make decisions about important features and other aspects of their vision that are more difficult to articulate.

This game is popular among software developers when setting out to capture the customer's view of a new application, but its use does not stop there. The game can help facilitate any vision-oriented discussion and has been used to describe topics ranging from "our future methodology" to "the ideal hire."

In all cases, the box is a focusing device: it wraps up a lot of otherwise intangible information into a nice physical object, prompting decisions along the way. When teams present or "sell" their boxes to each other, a number of things come to life, including the natural translation of features into benefits. Also, it's fun to do. The exercise results may be simple drawings or an actual box, which may live on well after as a friendly reminder of the big picture.

Figure 1 Product Box Game [29]

(Semester 1 – Weeks 9-15 and Semester 2) as well as provide an optimal experience to our students.

A. Weekly Activities

The following activities occur each week as part of their project sprints.

1) Scrum Standups

Once a week, the instructor meets with each team for a 10–15-minute standup. During this time, each team member discusses what they have done since last week's meeting, what they are planning on accomplishing before next week's meeting, and any roadblocks/obstacles that they have encountered. This works well for a couple of reasons:

*a) **Engaged Students**. The students have to speak up about what they are actually doing, and they cannot become wallflowers and not participate.*

*b) **Instructor Knowledge**. Actively managing a project will help the instructor understand how well a project is doing, as well as calibrate iteration reports (e.g., "They said they had no issues, but that is not what they are reporting in their end-of-iteration reports", or "They kept telling me they were on schedule, but in the end, they didn't deliver").*

One other approach that we have used when the class is large (e.g., in one class, one of us managed 83 students with 15 projects) is turning the meetings into what we call a "scrum of scrums." Each week, a different team member would be selected, and the instructor would meet with the team representatives. While the format is the same, the advantage is that it is easier to manage, and the disadvantage is that it is not as detailed an understanding of how a team is doing.

2) Customer Meetings

One of the key components of an Agile approach to software engineering is the customer (or user) involvement with the development team. By meeting regularly with the customer, the team can better meet the customer's needs, stay on track for the iteration, and get buy-in for any decisions. In our implementation, the students meet at least once a week with their customer, usually virtually, in the form of an Agile Standup where they talk about what they have accomplished since the last meeting, what they are planning on working on for the next week, and any issues/roadblocks they are dealing with. This works well for a couple of reasons:

*a) **Regular Accountability**. By meeting regularly, the status of an iteration should not be a surprise, and what is being promised is being delivered.*

*b) **Customer Interaction**. By requiring that the customers and students meet in person (usually virtually) weekly, students are provided with the opportunity to interact regularly with industry professionals.*

In addition, most of our customers are not on campus as they work in industry. Using virtual meeting technology (MS Teams, Zoom), we can solicit projects from anywhere in the world (we have had multiple projects with customers located outside of the U.S.), providing us with an unlimited number of options for possible projects. (More on this in Section IV.C.)

B. Iteration Activities

The following activities occur at the end of each iteration (i.e., every 3-4 weeks).

1) Customer Deliverables

Every three weeks, the students deliver an iteration to the customer. After a few iterations, hopefully, they can give their customer a Minimal Viable Product (MVP), but even before delivering the MVP, the customers still get the following products:

*a) **Showcase**. The students create a 5-6 minute professional video of the new features that are in the iteration, as well as do a live demo of the product as part of their end-of-iteration retrospective with their Customer. Each Student briefly presents what features/functionality they completed, showing a demo of how it works. This also has the bonus of students working on their presentation skills and professionalism.*

*b) **Prototype**. Something they can touch, look at, ask questions, etc. By doing this, students are getting pieces of the product in the customer's hands as soon as possible, heading off any possible issues further down the road when it may be too late to change.*

Some of our customers have asked for other products as part of their collaboration with the students. For example, some have asked to view the Team Report that students deliver to the instructor at the end of the iteration.

2) Team and Iteration Reports

In addition to customer/company deliverables, the students are required to submit a variety of products to the instructor for a grade, including a Team Iteration Report and Individual Iteration Reports. (The video Showcase discussed in Section IV.B.1 is also graded, as well as the Peer Evaluations discussed in IV.B.2.) In the written Team Iteration Report, the students include what user stories were attempted and completed, plans for the next iteration, burnup/burndown charts, retrospective, team temperature, and links to their Kanban board. In the written Individual Iteration Report, a student includes what user stories they personally attempted and completed, as well as stories they planned to complete but were unable to. Having these two different reports accomplishes a couple of objectives:

a) ***Team Iteration Report****. Not only does it provide project insights for the instructor, but it also forces the students to reflect on their outcomes and how they are working together. Further feedback from the instructor (after reading and grading the report), can allow for further reflection as to how the Team might be able to improve (from an outside perspective).*

b) ***Individual Iteration Report****. Similar to what was described in Section IV.A.1, this forces an individual student to account for their activities. And from an Instructor's perspective, discrepancies between what is reported in the Team Iteration Report and the Peer Evaluations (see IV.B.3), can come to light.*

Students often do not see the value of either of these reports. They see it as a task needed for a grade. The instructor must take the time to explain the value in both reports, as outlined in the points above.

3) Customer and Peer Evaluations

Perhaps the most important metrics are the feedback students receive from not only their customers but from their peers as well. Customers receive a Qualtrics survey where they evaluate the Team (on a Likert scale) in the following 6 areas:

- Professionalism
- Understanding of appropriate technologies
- Analyzing requirements
- Effectively communicating
- Showcase
- Overall expectations

In addition, customers are invited to include a free-form response regarding any additional feedback regarding their experiences. In total transparency, all of the customer feedback is shared with the students so that they feel good about what went well and reflect on what needs to be improved. Students also submit peer evaluations of their teammates as well as themselves. All of this provides useful feedback to not only the instructor but, perhaps more importantly, to the students:

a) ***Constructive Criticism****. By working closely with an individual (or sometimes individuals) working in industry, the students not only get to work on their professionalism and project management, but they learn about new tools, how companies work, and what sorts of problems they deal with.*

b) ***Self Reflection****. In general, Students rarely take the time to reflect on what they are doing well, what they are not doing well, and what they do not understand. To them, it is a race to get something done and get a grade. This chance to self-reflect forces them to think about what they are doing and how they can improve. Plus, that additional feedback from their peers adds another layer of self-reflection.*

The tool we use for peer evaluations is provided by PeerAssessment.com [26], a product of Hot Rocks Consulting L.L.C., owned by Rob Anson. We have found the tool to be an excellent, easy-to-use tool for gathering student peer evaluations (and at a low price – and free for the first course). It should also be noted that we do not include the self-assessment in the grade calculation in order to prevent the students from gaming the system regarding their grade. However, it can be useful to provide insight to the student as to how they see themself versus how others see them, but according to the literature, it is not recommended to include it in the grade calculations [27], [28].

C. Other

The following are some additional aspects to our approach to capstone projects that, while not necessarily Agile, are still relevant to creating a solid software engineering experience for the students.

1) Senior Design Expo

The College of Engineering, of which our department is a part, holds a Senior Design Expo during the last week of classes each semester. This is a chance for capstone projects across all engineering and computer science majors to show off what they have built. Setup like a science fair project, with posters on tables and laptops demonstrating their product, attendees walk around looking at their products, asking questions, and interacting with all the team members. This provides the students with two important outcomes:

a) ***Professionalism****. Students dress appropriately and have put together a professional pitch and poster. The event is advertised not only to the College but also to the University (we always see the Provost and President at the event) and the general outside community (including the customers).*

b) ***Communication****. This event highlights their ability to communicate effectively in various professional contexts. Until now, the students have just been "presenting" to either their customer or the instructor. This now opens them up to explain their work to others who do not know what they are doing, have not seen what they have produced, and know nothing about the problem.*

When this event was not held every semester (and sometimes not as well organized as it is today), our department would hold a mini version. The students do the same things they do for the larger event, but only for computer science, showing the scalability of such an event.

2) Tools

Depending on the project and what the customer requires, Students get exposed to a wide variety of tools. However, no matter what the "flavor" of the tool, every team gets exposed to three primary forms of communication:

a) ***Kanban Board****. This is where the students put their user story backlog and work with their customer to groom it for*

each iteration, laying out their path to the MVP. This also forces the students to think about how long a story (feature) will take to complete and, ultimately, how much can be promised to the customer for an iteration.

b) ***Source Code and Data Repository****. This is where students share their code and data with the customer. (Sometimes also used for their Kanban board.) This allows the customer to see how the students are progressing on the build and, in the end, how the product is delivered. Locally, we use GitHub for this purpose*.

c) ***Virtual Meeting****. While students are accustomed to virtual technologies like Teams, Zoom, or even Slack, using it for a group meeting with a customer forces them to put together an agenda, have their camera on (see Professionalism), and engage with their stakeholders. Such technology also allows us to solicit customers from anywhere in the world (see Section IV.C.3), where customers and students do not have to meet in person*.

Our University is a Microsoft shop. We encourage students to use MS Teams for the Kanban board and virtual messaging and GitHub for storing coding and data in a repository. However, for some companies, these platforms are not allowed or are inaccessible (i.e., they cannot be shared between our university and their company). In those instances, we allow the students to work with whatever technologies their customer wants, as long as the instructor also has access.

3) Soliciting Projects

The biggest upfront work when it comes to running a capstone course is the soliciting of projects. We start about three months before the project ideas are needed, allowing potential customers the opportunity to think about what projects they would like a team of students to work on, as well as identify what company personnel will be their customer contact. We primarily seek out *external* customers (i.e., not faculty/departments on campus), who are interested in working with our computer science students. In particular, we seek out customers interested in engaging with students over desiring a polished end-product. While we value having students create excellent products, we value an excellent experience for our students more. As such, we build the expectation that the end product will not necessarily be production quality. By approaching the experience with this mindset, concerns over intellectual property and ownership of the end product become less critical than the success of the engagement.

With this approach, we realize three positive experiences:

a) ***Real World Experience****. We target people in industry where they can practice Agile software engineering with an actual company. While it is easier to make up toy projects or build something for a researcher on campus, that interaction is not close to what we want them to experience. Plus, they lose out on learning what tools are being used in industry.*

b) ***Jobs****. Employers work with our students, who they may ultimately employ. It works both ways, as sometimes a student learns who they do not want to work for or what they do not want to do when they graduate.*

c) ***Alumni Relations****. Many times, the industry partners are alums of our University. This allows them to be more engaged with their alma mater and potentially become engaged in other avenues, such as serving on boards, providing internships, or contributing to scholarships*.

As part of our customer onboarding process, we conduct a webinar to communicate the expectations of how customers and students are to interact. This includes setting expectations on stakeholder time, frequency of communication, and how much guidance customers are expected to provide.

By the end of the capstone engagement, most customers express an appreciation of the opportunity to interact with our students and choose to do future projects. We typically experience a 33% turnover rate (not because of bad experiences, but usually because the customer is just too busy to take on a project), but we have always been able to reach out to new companies through either our campus career fair or alumni who have even taken the course when they were here.

4) Assigning Students to Projects

Once a customer has identified a willingness to work with our students on a project for their company, we ask them to provide the following details:

- Description of Proposed Project (brief - no more than a few sentences)
- Anticipated Skill Set/Tools
- Contact Name and E-mail Address

The project description and anticipated skill sets are shared with the students. The students then submit (through a Qualtrics survey) their ranked order of preference for what projects they want to work on, along with additional information like their major concentration, software engineering experience as part of an internship or co-op, technical proficiencies, etc. We also incorporate general demographic information to achieve better diversity outcomes. The instructor then uses that information to form the project teams.

Based on student feedback, we experimented with conducting a Pitch Day for projects. Rather than the students using only a textual representation of the project to make their preference decisions, we hosted the potential clients in person and through a virtual meeting to pitch their project to the students directly. One of the clients this semester presented from Peru, with their partner presenting in person.

This presentation event allowed students to gain more than a basic understanding of the project. Students could hear the interest and passion of the presenter about their project, which could indicate how invested they will be in the teams moving forward. Students could interact with the potential client and ask detailed questions about material not well presented in the narratives.

This process also allowed potential clients to appreciate better the overall process and journey on which these students were about to embark. One potential drawback of this event is project bias, where a dynamic speaker with a weak project could "prevail" over a stronger project with a less engaging

presenter. Ultimately, it is a truer representation of market presence and market forces in the process.

The benefits of this approach are:

*a) **Engagement**. Weighting their project rankings highly results in a student getting a project they really want to work on. Better engagement should lead to a stronger desire to be successful.*

*b) **Stronger Teams**. Matching up the anticipated skill sets of the project with the skill sets of the students plays to the strengths (or desires) of the students, allowing for a better chance of success for the entire team.*

V. Conclusions

In this work, we presented an approach implemented at our institution for not only training students in Agile software engineering principles but also incorporating the material into a mini project followed by 1.5 semesters of a real-world project for an industry partner. In addition to outlining the layout of our course, we provide several best practices for implementing our approach that will make managing the course more efficient, and improve the student experience. We hope to provide the students with as close to a real-world experience as possible, better preparing them for job interviews and placements. In the future, we are looking to add additional educational modules to the curriculum that are not necessarily related to Agile approaches but would be beneficial for the students to understand better when it comes to working on real-world projects. We have already added modules around secure software development and testing. Some other potential modules include dev ops or web development.

VI. References

[1] "Manifesto for Agile Software Development." Accessed: May 09, 2024. [Online]. Available: https://agilemanifesto.org/

[2] "2022 Tech Trends | Deloitte Insights." Accessed: May 09, 2024. [Online]. Available: https://www2.deloitte.com/us/en/insights/focus/tech-trends/2022.html

[3] "McKinsey 2021 business articles: The year in review | McKinsey & Company." Accessed: May 09, 2024. [Online]. Available: https://www.mckinsey.com/featured-insights/2021-year-in-review

[4] T. C. Krehbiel *et al.*, "Agile Manifesto for Teaching and Learning," *Journal of Effective Teaching*, vol. 17, no. 2, pp. 90–111, 2017.

[5] J. Campbell, S. Kurkovsky, A. Tafliovich, and C. W. Liew, "Scrum and agile methods in software engineering courses," *SIGCSE 2016 - Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pp. 319–320, Feb. 2016, doi: 10.1145/2839509.2844664.

[6] G. C. Gannod, D. A. Troy, J. E. Luczaj, and D. T. Rover, "Agile way of educating," *Proceedings - Frontiers in Education Conference, FIE*, vol. 2015, Dec. 2015, doi: 10.1109/FIE.2015.7344019.

[7] G. C. Gannod *et al.*, "Agile University: Using Agile Processes to Increase Engagement, Assessment, and Internalization of Outcomes," in *34th Lilly International Conference on College Teaching*, 2014.

[8] M. Görner, S. Kassel, and T. Klein, "Agile software development in business informatics: using agile methods for teaching purposes at the University of Applied Sciences, Zwickau," in *Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015)*, 2016, pp. 21–27.

[9] A. Jaime, J. M. Blanco, C. Dominguez, A. Sánchez, J. Heras, and I. Usandizaga, "Spiral and project-based learning with peer assessment in a computer science project management course," *J Sci Educ Technol*, vol. 25, pp. 439–449, 2016.

[10] M. Kropp, A. Meier, and R. Biddle, "Teaching agile collaboration skills in the classroom," *Proceedings-2016 IEEE 29th Conference on Software Engineering Education and Training*, CSEEandT 2016, 118–127." 2016.

[11] K. Royle and J. Nikolic, "A modern mixture, agency, capability, technology and 'scrum': Agile work practices for learning and teaching in schools," *University of Wolver Hampton, WIRE*, 2016.

[12] A. Cockburn and J. Highsmith, "Agile software development: The people factor," *Computer (Long Beach Calif)*, vol. 34, no. 11, pp. 131–133, Nov. 2001, doi: 10.1109/2.963450.

[13] C. Scharff and R. Verma, "Scrum to support mobile application development projects in a just-in-time learning context," *Proceedings - International Conference on Software Engineering*, pp. 25–31, 2010, doi: 10.1145/1833310.1833315.

[14] T. Clear, M. Goldweber, F. H. Young, P. M. Leidig, and K. Scott, "Resources for instructors of capstone courses in computing," in *Working group reports from ITiCSE on Innovation and technology in computer science education*, 2001, pp. 93–113.

[15] F. P. Brooks, "The mythical man-month," *Datamation*, vol. 20, no. 12, pp. 44–52, 1974.

[16] A. Goold, "Providing process for projects in capstone courses," *ACM Sigcse Bulletin*, vol. 35, no. 3, pp. 26–29, 2003.

[17] R. Holmes, M. Craig, K. Reid, and E. Stroulia, "Lessons learned managing distributed software engineering courses," in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 321–324.

[18] M. C. Bastarrica, D. Perovich, and M. M. Samary, "What can students get from a software engineering capstone course?," in *2017 IEEE/ACM 39th International Conference on software engineering: software engineering Education and Training Track (ICSE-SEET)*, 2017, pp. 137–145.

[19] X. Fan, "Seven Principles of Undergraduate Capstone Project Management," in *Proceedings of the*

International Conference on Software Engineering Research and Practice (SERP), 2018, pp. 106–112.

[20] D. Broman, K. Sandahl, and M. A. Baker, "The company approach to software engineering project courses," *IEEE Transactions on Education*, vol. 55, no. 4, pp. 445–452, 2012.

[21] K. Schwaber and J. Sutherland, "The Official Scrum Guide," scrumguides.org. Accessed: May 13, 2024. [Online]. Available: https://scrumguides.org/scrum-guide.html

[22] D. Gray, S. Brown, and J. Macanufo, "Gamestorming: A playbook for innovators, rulebreakers, and changemakers," *O'Reilly Media, Inc*., 2010.

[23] K. Schwaber, "Scrum development process," *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*, 1997, pp. 117–134.

[24] "ChatGPT." Accessed: May 13, 2024. [Online]. Available: https://chatgpt.com/

[25] "Criteria for Accrediting Computing Programs, 2023 - 2024 - ABET." Accessed: May 13, 2024. [Online]. Available: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2023-2024/

[26] "Peer Assessment." Accessed: May 13, 2024. [Online]. Available: https://peerassessment.com/

[27] M. Lejk and M. Wyvill, "The effect of the inclusion of self-assessment with peer assessment of contributions to a group project: A quantitative study of secret and agreed assessments," *Assess Eval High Educ*, vol. 26, no. 6, pp. 551–561, 2001.

[28] Y. Tu and M. Lu, "Peer-and-self assessment to reveal the ranking of each individual's contribution to a group project," *Journal of Information Systems Education*, vol. 16, no. 2, 2005.

[29] "Design The Box – Gamestorming." Accessed: May 13, 2024. [Online]. Available: https://gamestorming.com/design-the-box/