Proceedings of the Informing Science + Information Technology Education Conference

An Official Publication of the Informing Science Institute
InformingScience.org

InformingScience.org/Publications

Online July 5 – 6, 2023

INTEGRATING AGILE SOFTWARE DEVELOPMENT PRACTICE IN A CLASSROOM SETTING

Anthony Kampa	Digikey Electronics, Fargo, ND, United States	kampa051@crk.umn.edu
Christine Bakke	Grand Canyon University' Phoeniz, AZ, United States	Christine.Bakke@gcu.edu

ABSTRACT

Aim/Purpose	This paper explores how best to implement Agile style courses into university curriculum. It is a starting point for teachers who are unsure how to structure their classes in an Agile way.
Background	This paper explores the researcher's experiences with Agile in the classroom, outside the classroom and in a professional setting. Recommendations are made on how to best introduce students to Agile concepts and prepare them for their careers.
Methodology	This paper is an exploratory case study in determining whether or not students are properly equipped for their careers. Information was gathered through a qualitative interview administered virtually. The sample is taken from students who are recent graduates from public universities in the Midwest.
Contribution	This study provides tangible and practical suggestions to best utilize Agile methodologies in an academic setting.
Findings	A project-based learning class taught using the agile methodology would provide a beneficial and flexible class for students no matter where their careers take them. This style of class can be offered to any level of undergraduate student but more advanced students will likely get more out of it. Advanced students should be encouraged to work across disciplines to foster communication skills and provide valuable experience working with non-developers.
Recommendations for Practitioners	Agile is not a silver bullet. Not all classes will be a good fit for this style of teaching. Practitioners should consider a blend of classes with different teaching styles.
Recommendations for Researchers	Researchers are encouraged to explore different methodologies for including Agile development into a classroom environment.

Accepted by Editor Michael Jones | Received: March 13, 2023 | Revised: May 31, June 7, 2023 | Accepted: June 9, 2023.

Cite as: Kampa, A. (2023). Integrating agile software development practice in a classroom setting. In M. Jones (Ed.), *Proceedings of InSITE 2023: Informing Science and Information Technology Education Conference*, Article 27. Informing Science Institute. https://doi.org/10.28945/5159

Impact on Society	This study will help to better prepare the next generation of software developers for eventual careers.
Future Research	Practitioners and researchers can expand on this study by exploring a multi-year study following students who had Agile curriculum and students who received traditional class methods as they graduate and join the workforce. Researchers can explore other methods of implementing Agile in the classroom to further refine suggestions provided in this paper.
Keywords	Agile development, Scrum, software development, information technology, software engineering, classroom teaching

INTRODUCTION

The world of software development is constantly changing. In the last few years there have been booms in cryptocurrency, cloud computing, IoT and AI to name but a few. As these new technologies become more and more relevant in the workplace, universities run the risk of failing to equip their students for their future career (Garousi et al. 2020; Oguz & Oguz, 2019). To address these rapid changes, development methodologies such as Agile and Scrum are replacing less flexible development methodologies such as waterfall. Sahin and Celikkan (2020) found that there is a substantial knowledge gap when it comes to the Agile Development Methodology. Implementing a project based learning class utilizing the Agile development methodology can address these shortcomings and help students tailor their knowledge to the changing environment.

This study does not seek to establish the well-known knowledge gap. Instead, it attempts to answer the question of how to emulate an Agile work environment in an academic setting. Research was primarily done through interviews with upperclassmen and recent college graduates. The proposals made in this paper are meant to be built upon to suit the changing needs of the workforce and each university.

BACKGROUND

At its core, Agile is an iterative design process that aims to put people ahead of processes (Beck et al, 2001). Work is broken down into 'sprints.' These sprints run on average over one or two weeks. Every team member works off a collective backlog. This is the place where all work is stored. Work is broken up into Product Backlog Items (PBIs) and organized in matter of importance. Items at the top of the board are the most important and next items to be pulled in. Items further down the board are of lower importance and are less likely to be addressed. Work on a PBI is only started once it has been pulled into a sprint. Each PBI is assigned a number of effort points. These represent the teams estimation as to how long the item will take to complete. Traditionally estimation is done using the fibonacci sequence. Each team decides what constitutes one effort point.

Aside from the basic workflow of Agile, there are a series of Agile ceremonies held. Sprint Planning is where the team meets and discusses what they will attempt to accomplish in the upcoming sprint. This is held at the beginning of a new sprint. During planning any unfinished PBIs are brought into the next week to continue being worked on. New PBIs are pulled into the sprint board to be worked on. After planning the Sprint has begun and work begins on any PBIs currently in the Sprint. Another major Agile ceremony is the Sprint Retro. This is where an Agile team analyzes the results of the last sprint and looks at how they can improve their process. These improvements could be at an interpersonal level, or at a process level. The final major sprint event is refinement. This is where new user stories are given effort points and initially prioritized on the backlog. Later the product owner can reprioritize as they see fit.

Agile incorporates a few additional positions within the team. The primary additions are a Scrum Master and the Product Owner. The Scrum Master is responsible for all things Agile. They are in

charge of scheduling and running the various ceremonies. Their other major responsibility is to remove any obstacles that the team encounters outside of the technical realm. This could be communicating with another team or resolving interpersonal issues within the team. These individuals often receive training to work effectively in this role. The Product Owner is the overarching owner of the whole project. They have the final say as to what gets pulled in and are responsible for making sure the backlog is prioritized.

LITERATURE REVIEW

Alongside the rigor of the academic degree, companies are looking for more than technical skills. Companies value soft skills, having a portfolio and experience with Agile (Galster et al, 2022). Additionally, projects are one of the 4 most important factors when evaluating a potential candidate (Stepanova et al, 2021). These competencies and more can be addressed through the proper implementation of a project-based course taught using Agile.

The most apparent benefit is Agile experience. Outside of strengthening a resume and applicant, previous knowledge of agile reduces the cognitive load of a new job. There is already enough to learn between a new codebase, new tools, and new deployment processes. Removing Agile from this load allows acceleration of the onboarding process.

Regardless of culture, the most desired soft skill for a developer is communication (Ahmed et al., 2012). Teaching in an Agile methodology promotes communication and better soft skills. Students should take turns being the Scrum Master on a week-to-week basis. In doing so, they will have to coordinate with other group members. This benefit will be further discussed in the implementation portion. Diverse groups help introduce students to other ways of thinking and foster better communication skills.

Outside of building a portfolio, teaching a project-based class using Agile has benefits to learning outcomes. The first is student ownership of a project has a positive effect on learning outcomes (Umar & Ko, 2022). Allowing students to select what their project is helps invest them in their work. Additionally, students tend to value autonomy to choose what they learn (Linden, 2018). As students work on a project they want to build, once they complete the course they are setting themselves up for success.

METHODOLOGY

The initial suggestion was developed through a combination of the author's experiences earning an undergraduate and a research project undertaken with a professor. The goal of the project was to develop a website for people experiencing temporary mutism. Work on this project was done utilizing the Agile design methodology. The professor acted as the product owner and prioritized work on a weekly basis. The sprint duration was one week. At the beginning of each week a meeting was held to discuss progress made during the last Sprint. A working prototype was provided early in the process. The prototype was improved each Sprint to best comply with the vision of the product owner. Later in the process, major revisions were made at the behest of the product owner. These included the ability to add buttons and multiple redesigns of the page navigation style. This experience helped form the basic class format. The idea was then improved through an interview process with recent college graduates to determine if it would have been a valuable class to have and how to improve it.

Interview participants were asked a series of open-ended questions about Agile methodology from a professional and educational standpoint. Professionally, respondents described their work environment, general size of company and how well their coursework prepared them for their current position. Academic responses were collected concerning exposure to Agile in their curriculum, how they would improve the major, any additional projects they worked on in college and provided feedback on a suggested Agile classroom implementation. Interviews were conducted over the phone and responses documented electronically.

Participants ranged in age from 23 to 28. Companies of employment were as small as 20 employees all the way up to 4000+. Many different positions were interviewed such as Software Engineer, Senior Developer and backend developer. All had graduated from a small subset of public universities in the Midwest. Graduates were graduated several years apart from each other. This helps determine whether the curriculum at these institutions has been updated in the ensuing time.

RESULTS AND RECOMMENDATIONS

The primary complaint across multiple degrees and universities was that the focus of the major was too broad. There was a strong emphasis placed on theory, but without real world examples to apply the knowledge to it didn't add a lot of value. Another common theme was the in class projects were unrealistic. They weren't software programs that everyday people would use. Another recurring theme was the value of working on a project with a professor. These projects taught valuable skills that weren't located inside of any course such as time management and working with feedback from stakeholders. Every participant agreed that a class taught using Agile would have been beneficial to have taken, even for those who are not currently working in an Agile environment.

The final interview question focused on garnering feedback about this proposed project-based learning class structure. At the start of the semester, students select a project to work on. This can be from a curated list, broad category, such as retro video games, or have freedom to choose any project they desire. The students will be broken up into groups of 4-5 students. Each group will compose one Agile team. The Sprint duration will be one week.

In lieu of traditional homework, students will be expected to spend between 3-5 hours outside of class working on their selected project. Any less than 3 hours compromises a student's ability to make progress. Requiring anything more than 5 is too heavy of a workload. Outside of an hourly requirement, groups will have to meet once a week to conduct summarized Agile ceremonies.

One person out of the group will be selected as Scrum Master. This person will change on a weekly basis. During their meeting, they will start with standup. What has each student worked on in the last week and where did they get stuck. As part of planning, they will determine what they will be working on the upcoming week and display this on their backlog. How the backlog is implemented is up to the discretion of the professor.

To simulate Sprint Retro, this meeting will provide an opportunity for students to discuss how they can overcome whatever blocked them. If they are stuck they can ask the rest of the team for advice on how to move past the blocker.

During the week, class can continue to be lecture based. One day out of the week will be presentation day. Here one group from the class will present to the entire class. If any group member has a working prototype, it can either be demoed during the presentation or within the presentation.

This idea was met with general acceptance, but there were concerns expressed due to the simplicity of this design. Many of these concerns can be left to the discretion of the teacher to be addressed. One concern was the risk monotony of having one week sprints where nothing changes. Another recurring concern was what level of undergraduate this class would suit the best. The final theme in feedback was making sure the projects are relevant.

One of the best ways to address the risk of monotony is to require each group to implement one additional Agile term from the Agile glossary each week. This way each group's familiarity with all things Agile will continue to grow, but still leave them the autonomy to tailor what they learn. This additional requirement can be documented in their presentation.

This approach can be tailored to any undergraduate level, whether the students are freshmen or seniors. This style of course should generally be offered at a sophomore or higher level to avoid over-

whelming freshmen. It also gives incoming students a chance to learn about programming before being given a more advanced class like this. If offered at a higher level or as a capstone project, the possibilities open up. Companies interested in developing a new piece of software or even students from another major would be excellent product owners. This would further help emulate a realistic working environment balancing the time the students have with the desires of the business. It also helps foster cross discipline communication, grows their portfolio and depending on the product owner potentially opens the door to internships and beyond.

Adding in product owners from outside of the class also addresses concerns about the realism of the program. This factor is strongly influenced by the professor. While students should have the ability to focus on a project of their choosing, the professor still has the final say into whether or not that project gets the green light. This will help keep software more firmly grounded in what a developer might see every day instead of something very niche like an aircraft navigation program.

One apparent question that needs answering is why not make it a group project instead of an individual project? If the class is offered to upperclassmen or as a capstone project, group work may be the way to go. This framework is meant to be adapted to fit the needs of the class and professor. Working as a group helps simulate a real work environment better. It also offers opportunities to integrate additional technologies such as source control, whether using Git bash or a Git interface. Group work may backfire at an undergraduate level where not everyone is invested into the success or failure of a project. At a lower level individual projects allow students to pursue their passions and invest in their own education. There is a place for group work, and a place for individual projects. This decision should be made on a class by class basis.

Example Progression

For underclassmen, allow students to choose a project they want to work on. Use that project as an introduction to Agile. Encourage students to start with a small project and take it all the way from conception to publication. This could be from a curated list of simple apps such as a calculator or other similar programs. Once they reach a more advanced programming level, allow them to choose any project they want. Even if a student selects too large of a project, they should be able to submit some portion of the project as working by the end of the course.

An excellent class to replace would be a project management course, or one that explores different methodologies of designing a project. The weekly lectures can still focus on different methodologies such as Waterfall, Spiral and Kanban. Small quizzes to confirm reading comprehension would be an appropriate workload alongside completing a project developed using Agile.

This approach could also be utilized as a portion of a class. Instead of running the entire class in an Agile way, a portion could be dedicated to a smaller project. A useful and succinct project could be accessing an API. The market value of APIs is projected to increase over 800% during the next decade (Precedence Research, 2022). There are plenty of APIs that are widely available. A good introduction to all of these technologies and methodologies would be a short project. Students would create a repository on Git. Have them commit their work at the end of each session and have them document with screenshots to ensure they are committing regularly. For their project, they need to reach out to an API. Provide a list of widely available ones as resources but encourage students to find their own if nothing on the list appeals to them. Their project should reach out to any endpoint on the API and display the result. A few free backlog services such as Jira or Trello would help groups emulate an Agile task board and assist with workflow.

Ideally their grade should not be tied to project completion, but on the progress they make in an Agile framework. Even if the project is not completed by the end of the class, some working portion of it should be. Have this be what is submitted at the end of the semester alongside a paper detailing how the development went. This allows students to analyze if scope creep got the best of them, or if they selected an appropriate sized project and estimated correctly. This style of class could be a good

candidate for a semester 2 class. That way students with motivation would have the summer to finish their project on their own time.

If the students are more advanced and have a grasp of Agile development, they would be good candidates for including product owners from elsewhere. At this point collaboration between disciplines would be best to emulate a real work environment. Collaborate with another class of students from a different major. Have the business students function as the 'product owner.' Business students pitch their ideas to the class of developers. Each group chooses what business idea they wish to pursue. This benefits the other discipline as well by allowing them to practice presenting an idea to a group that might not understand what they're talking about. These product owners won't be expected to perform all the Agile functions a full product owner would. Their primary job is to provide feedback on the systems produced and provide guidance to align that system more closely to their idea.

This suggestion would make an excellent capstone project. One of the major risks here is coordination across disciplines. Keeping communication channels within one class reduces the chance of a communication breakdown. I would suggest as part of the capstone include how working with a real product owner went. This will allow groups to document what their struggles were and how they overcame interpersonal challenges. Students should be allowed to fail. If the capstone falls apart, diagnose what happened with a postmortem. Did they commit to too large of a project? Were there issues with who was selected as the product owner? Product owners could also be brought in from interested businesses.

The perfect class for any given university may require several iterations. Encourage the students to provide feedback for the course. What worked well, what didn't work well, what improvement would be made and just like Agile, iterate on the class. Tweak it slightly from semester to semester to make sure students are getting benefit from the inclusion of it to the curriculum.

Limitations of Recommendation

Not every class will be a good fit for Agile. This approach lends itself to smaller class sizes which are not always possible. The goal is not for the entire curriculum to be taught in an Agile way, but to more strongly emphasize what Agile is and experience a typical Agile workflow.

There are many decisions that can be implemented on a campus by campus, or even class by class basis. It is the author's belief that a personal project is a good way to get buy-in to this form of development. If students work on a passion project, it will be easier to ask them for hours outside of class. If the entire group doesn't buy into whatever project is selected, it may produce negative results when working together.. This approach may not be considered professional enough. This idea is meant to be modified to fit any university. If resources are available to host two environments (development and 'production'), use kubernetes containers and continuous deployment that will prepare students in an even better manner. This is not possible for all campuses to achieve this level of real world simulation.

Limitations and Future Research

This research has several limitations that could be expanded upon in the future. The first major limitation is the sample size is limited to a variety of students in the Midwest. The second limitation is the qualitative nature of this research.

This topic can be expanded in the future with quantitative research following college students in Agile college environments and traditional college environments. This would help determine the impact of Agile development methods on GPA. Furthermore, as these students graduate and join the workforce follow up surveys of this cohort could offer guidance on how to improve the curriculum. A study of this magnitude would help quantify if Agile students adapt better to their new position

compared to their more traditionally educated compatriots. Furthermore, the relation could be explored between cohorts and how they perform in an Agile environment as compared to a traditional (waterfall) work environment.

CONCLUSIONS

This study aims to provide a basic framework for a project-based learning class taught using the Agile methodology. Ideally the entire semester would be taught in this manner, but it could be used to teach one module within a class. The framework is intentionally left general and high level to allow each teacher to customize it to suit their needs. They are also kept general due to the variability of available resources to each university. By adding a project-based learning class students will be empowered to take control of their education and shape it more to their interests. The idea of the class is to iterate with feedback from the students to make it more effective each semester.

REFERENCES

Ahmed, F., Capretz, L. F., & Campbell, P. (2012). Evaluating the demand for soft skills in software development. *IT Professional, 14*(1), 44-49. https://doi.org/10.1109/MITP.2012.7

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *The agile manifesto.* Agile Alliance. http://agilemanifesto.org/

Galster, M., Mitrovic, A., Malinen, S., & Holland, J. (2022). What soft skills does the software industry *really* want? An exploratory study of software positions in New Zealand. *In Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '22).* Association for Computing Machinery, New York, NY, USA, 272–282. https://doi.org/10.1145/3544902.3546247

Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2020). Closing the gap between software engineering education and industrial needs. *IEEE Software, 37*(2), 68-77. https://doi.org/10.1109/MS.2018.2880823

Linden, T. (2018). Scrum-based learning environment: Fostering self-regulated learning. *Journal of Information Systems Education, 29(1), 65-74.*

Oguz, D., & Oguz, K. (2019). Perspectives on the gap between the software industry and the software engineering education. *IEEE Access, 7,* 117527-117543. https://doi.org/10.1109/ACCESS.2019.2936660

Precedence Research. (2022). *API Management Market (by Component: Solutions, Services; by Deployment: On Premises, Cloud; by Organization Size: Large Enterprises, Small and Medium Enterprises; by End User: Banking and Financial Institutes, Retail, IT and Telecommunications, Consumer Goods, Others) - Global Industry Analysis, Size, Share, Growth, Trends, Regional Outlook, and Forecast 2022-2030.* https://www.precedenceresearch.com/api-management-market

Sahin, Y. G., & Celikkan, U. (2020). Information technology asymmetry and gaps between higher education institutions and industry. *Journal of Information Technology Education: Research, 19,* 339-365. https://doi.org/10.28945/4553

Stepanova, A., Weaver, A., Lahey, J., Alexander, G., &. Hammond, T. (2021). Hiring CS graduates: What we learned from employers. *ACM Transactions on Computing Education (TOCE), 22*(1), 1–20. https://doi.org/10.1145/3474623

Umar, M., & Ko, I. (2022) E-Learning: Direct Effect of Student Learning Effectiveness and Engagement through Project-Based Learning, Team Cohesion, and Flipped Learning during the COVID-19 Pandemic. *Sustainability. 2022, 14*(3),1724. https://doi.org/10.3390/su14031724

AUTHOR

Anthony Kampa is a Software Engineer at Digikey Electronics. He earned his Bachelors in Software Engineering from the University of Minnesota Crookston. A large portion of the curriculum there was taught using the Agile Methodology. This shaped his research interests to focus on how to bring realistic work experiences into the classroom and investigate the best way for any professor to teach a class using Agile.

Dr. Christine Bakke is an associate professor at Grand Canyon University where she is an instructor in Cybersecurity, Information Technology, and Computer Science programming. After earning an IT PhD focused on active learning and educational robotics, she taught Computer Science, IT, and Software Engineering for Universities in Northern Minnesota. Her work with students often incorporates prior experiences from her professional career working with networking, cybersecurity, databases, servers, and programming. Current research interests include complex programming projects, active learning, robotics, development of disability-assistive software, and IoT.