A Recommendation for Implementing Agile Development into the Computer Science Curriculum of Kansas State University

**Prepared for:** Dr. Scott DeLoach

Department Head, Computer Science

Kansas State University

**Prepared by:** Jacob Peterson

Computer Science Senior

Kansas State University

May 4, 2020

Table of Contents

[**INTRODUCTION** 1](#_Toc39762891)

[**Summary** 1](#_Toc39762892)

[**Problem/Opportunity** 1](#_Toc39762893)

[**Options** 1](#_Toc39762894)

[**Criteria** 2](#_Toc39762895)

[**Organization of Report** 2](#_Toc39762896)

[**METHODS** 2](#_Toc39762897)

[**RESULTS AND DISCUSSIONS** 3](#_Toc39762898)

[**Description of Agile Development** 3](#_Toc39762899)

[**Proposed Plan of Implementation** 5](#_Toc39762900)

[**Benefit to Curriculum** 7](#_Toc39762901)

[*Deepen Curriculum* 7](#_Toc39762902)

[*Tie into the Industry* 7](#_Toc39762903)

[**Benefit to Students** 8](#_Toc39762904)

[*Improve Code Quality* 8](#_Toc39762905)

[*Real World Knowledge* 10](#_Toc39762906)

[*Team Skills* 11](#_Toc39762907)

[**Costs** 12](#_Toc39762908)

[*Costs to Students* 12](#_Toc39762909)

[*Costs to University* 13](#_Toc39762910)

[**CONCLUSION AND RECOMMENDATION** 13](#_Toc39762911)

List of Illustrations

[Figure 1 Agile Methodology 3](#_Toc39682550)

[Figure 2 Waterfall Development 5](#_Toc39682551)

[Figure 3 A CS Major’s Senior Year 6](#_Toc39682552)

[Figure 4 The Industry’s Shift Towards Agile 8](#_Toc39682553)

[Figure 5 Agile’s Benefits on a Code Base. 9](#_Toc39682554)

# **INTRODUCTION**

## **Summary**

The purpose of this report is to explore the opportunity of implementing agile development, a common software engineering workforce practice, into the computer science (CS) curriculum at Kansas State University. CS majors have to assimilate into a rapidly changing and evolving industry, and there is a growing disconnect between what students learn in school and what is expected of them in the field. The CS department and its department head, Dr. Scott Deloach, have the opportunity to expand the curriculum’s coursework in order to include more practical and applicable techniques that students will need to succeed in the job force, most importantly among them agile development and its large scale team projects.

## **Problem/Opportunity**

Currently, according to the computer science course catalog, K-State offers no classes relating to agile development, or even software development lifecycle or process practices at all (Kansas State University 2019). K-State courses do not promote the large-scale team projects, large code bases, sprints, and test-driven development that comes with agile, and this lack of experience will hurt students preparing to graduate into a competitive and highly skilled industry. The addition of agile methodologies into the required computer science coursework would greatly increase student preparedness for the workforce. The number of employers “embracing” agile continues to increase, so integrating agile into the curriculum will prepare students for the workforce (Campus Technology, 2018). The number one goal of the university as an establishment is to prepare students for life after college, and with the workplace constantly evolving, the computer science department needs to evolve to match employers’ needs.

These employer needs are centering increasingly around agile development in the software development industry. Teaching these practices at Kansas State University would give students a leg up when it comes time to start interviewing for jobs. A Campus Technology interview even states that employers are beginning to explicitly ask questions related to agile methodologies in their interviews for new software developers (2018). Preparing students with these practices before they hit the workforce would be a benefit to the university, and the students it serves.

## **Options**

Incorporating agile development practices into current Kansas State CS curriculum would be a great improvement over the status quo. Currently, there is no agile development practices being implemented at any level of the CS department. The best option for doing this would be to include a singular class to detail the agile methodology. This would prevent having to alter any pre-existing coursework in any classes, allowing for very minimal changes to the K-State computer science coursework. In addition, students in the computer science program already have multiple free “tech elective” hours built into their curriculum, meaning there is plenty of room for an agile class. Agile development in this class would take the form of large-scale team projects, pair programming, test driven development, user stories, and collaboration over a large and ever-evolving code base. This option would be a significant improvement to students’ learning over the status quo, in which there are no agile development practices being implemented into the curriculum.

## **Criteria**

To further discuss the incorporation of agile development into our school, I shall discuss its merit on the following criteria.

• The benefit to the curriculum. A successful plan will both improve the CS curriculum and fit into the current existing curriculum. The proposed changes to curriculum must be something that can practically and effectively be taught with the school’s resources, and must be something that can successfully be implemented in a classroom setting.

• The benefit to the students. The plan must provide value to students in terms of both educational benefits and benefits to the employability of the student.

• Cost associated with these changes. An effective implementation of agile development will have to be time and money efficient for both the students and the university.

## **Organization of Report**

This report will include six sections.

First, I shall provide a background on agile development, as it is often a foreign term even to CS majors. I will describe agile, explain its importance within the workforce, and discuss its benefit on software development. This will provide a sufficient background on the topic for the rest of the report.

Secondly, I will discuss the practical details involved with teaching agile development in a university setting. This section will detail how agile can be taught in a classroom, and lay out my proposed plan of adding an agile development class to the curriculum in place of one of the computer science tech electives.

Thirdly, I will examine the benefits implementing agile development has on the university. This section will also tie in the importance of agile development to the industry, and why it needs to be taught in schools. This will provide incentive for the school to take my recommendation into consideration.

In the fourth section, I will examine the benefits of agile development practices on students, and its ability to increase their preparedness for the job field. This will be explained in terms of how my proposed plan will increase job readiness in students, and the general benefits these practices will have on the students code and programming skills.

In the fifth section, I will detail the costs of the proposed plan. This section will include details on the costs of adding an additional class to both the university in terms of logistics, and the costs to students in terms of fitting this class into their schedules.

In the final section, I will summarize the benefits an agile development class would provide over the status quo of the Kansas State CS curriculum. I will compare its pros and cons one last time, and summarize my findings and recommendation.

# **METHODS**

For this report, a large amount of secondary research was done to compile research on agile implementation at the university level. The Kansas State University Library was used to gather this data. Scholarly articles and conference presentations were the primary type of source in which this information is presented. There were many sources detailing how a school has implemented agile into its coursework, and with what levels of success. I also pulled information from several interviews with experts in the field or recruiters to gather information on how agile is affecting the industry, and how agile is playing a role in job recruiting.

In addition, a large amount of data was obtained from informational websites, such as agilealliance.org, in order to gather sufficient background information on agile development for the audience. This information was primarily used in the background portion of Results and Discussion section of the report. Finally, Kansas State University’s website was instrumental in obtaining data regarding computer science curriculum structure and flowchart information. Class information was important to have when forming my recommendation for implementation.

# **RESULTS AND DISCUSSIONS**

## **Description of Agile Development**

Agile development is a software development approach revolving around collaboration and iteration. Some of the most common features of agile development are pair program, test driven development, stand-ups, user stories, sprints, a self-organizing team, collaboration, and a feedback loop (Agile Alliance, 2020). Agile differs from tradition development, sometimes called waterfall development, because of its circular and continuous iterations. This contrasts with waterfall development’s linear, stepwise phases of development. A visual representation of the agile development process is outlined in Figure 1.

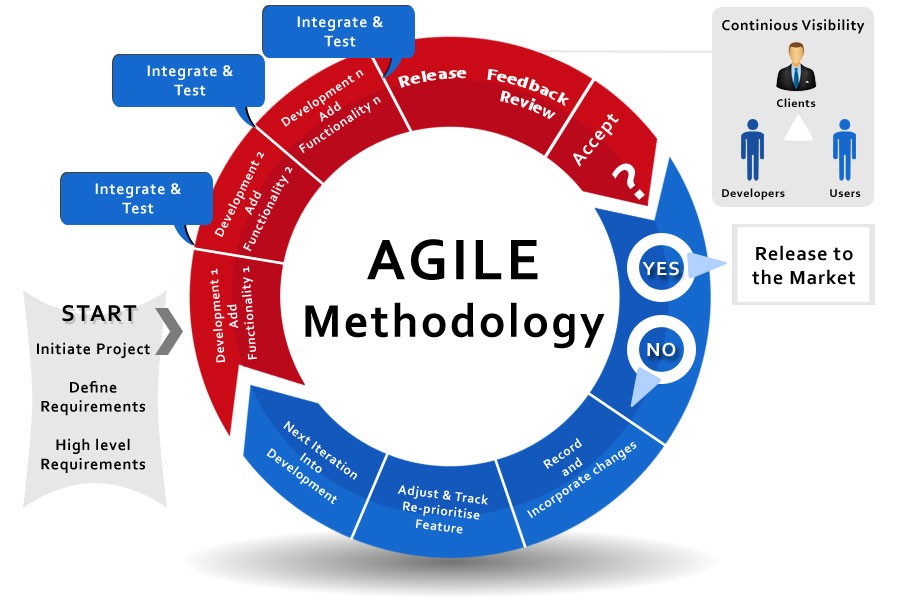


Figure 1. Agile Methodology (Scrum.as Academy, 2019)

If we were to walk through the process, Agile development begins with the defining of high-level requirements and desired functionalities of the software, as outlined in the “start” section of Figure 1. These requirements are often stored in an organizational software such as Azure DevOps, and act as a guiding outline for the software, like construction blueprints for a building. The development stage, colored red in Figure 1, describes the phase in which code is being written and the requirements of the project are being realized. These development phases are split by integration and testing phases, so that a minimum viable product is always being delivered. A Minimum viable product is a business term which describes the smallest functional product that can provide value to the customer. This also allows for maximum feedback from the client at all times, which is another major goal of the agile model.

During the development phase, the integration phases are often marked by a “standup”, or a meeting between team members where each member shares what they have been working on and any problems they have encountered. After development is complete, the product is then showed to clients and users so that feedback can be collected. A very distinct and important part of agile development is a triangle of continuous visibility between developers, users, and clients. This way, throughout every stage of development, feedback can be collected from clients and users and integrated into the final product. This feedback loop is the defining characteristic of agile development, and the circular, iterative approach of agile is what gives it its merit.

If the final product is accepted by clients and users, the product is released into the market. However, often the product is not accepted during the first cycle of development. In this case, the development process enters the blue section in Figure 1. The feedback is used to record and incorporate changes into development, adjust features and re-prioritize requirements. Once the scope of the product has been adjusted to meet feedback, a new iteration of the product is pushed into development. This cycle continues as long as necessary to produce a product that satisfies the customer base. This iterative approach to software development used by agile ensures maximum visibility of the product to customers during development, and maximum collaboration between teams. This approach also allows for constant feedback on a product, as well as increasing flexibility and seeing benefit from a product sooner.

If we compare agile to traditional development, we see several differences and benefits. The traditional (waterfall) model of software development can be seen in figure 2 below. Traditional development is very linear in nature. Requirements are defined, the code is written, the code is tested, and the product is then deployed, all within a linear fashion. There is a very distinctive start and end to the process. This allows for very little adaptability and flexibility for a project, and thus a very high-risk factor. In addition, feedback is a much less vital part of the development process, so the client has much less say in the outcome of the final product. In agile development, an iterative approach rather than a linear approach allows for more consistent feedback, and much lower risk due to the ability to alter the specifications of the product in the refactoring phase of development. These direct codebase benefits will be described in more detail in the “Benefits to Students” section of the report.



Figure 2. Waterfall Development (Search Software Quality, 2019)

## **Proposed Plan of Implementation**

Along with the benefits agile brings to the industry, it contains many useful skills and practices that would benefit students at the university. In addition, agile is becoming a growing part of the industry and will soon be all but required knowledge for students graduating with the hopes of becoming a software developer. This section will describe my proposed plan of implementation specifically at Kansas State University.

The Kansas State University computer science curriculum currently contains 120 hours of requirement coursework, and not a single class covering agile development (Kansas State University, 2019). Few classes even touch on real-world applications of software development at all, and the coursework prefers to stick to basic programming concepts, languages, architecture, and practices. In addition, the curriculum contains a very large number of electives built into it. Aside from 4 required mathematics courses (Calc I, Calc II, Discrete Math, and Statistics), students are required to take 5 additional math or natural science elective courses, a communication electives, 3 unrestricted electives, and 2 humanities electives (2019). While slightly excessive, these electives provide students with exposure to coursework outside of their major and are often required for national accreditation purposes. However, aside from these electives and coursework related to their major, CS students are required to take 3 “tech electives” during their senior year (2019). These electives are upper level classes of the students choosing within the computer science field and are designed to give students a chance to branch out into different career fields within computer science. For reference, the senior portion of the computer science flowchart can be seen in Figure 3 below.

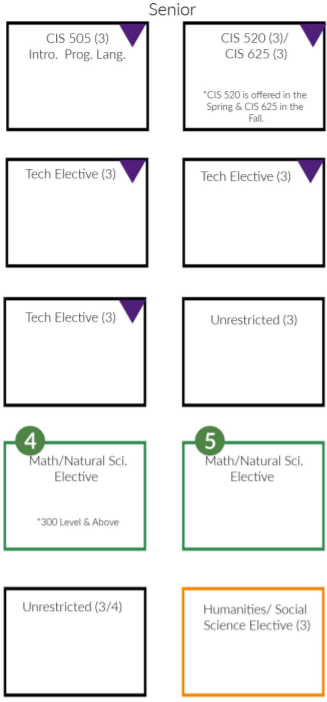


Figure 3. A CS Major’s Senior Year (Kansas State University, 2019)

My proposition is to replace a tech elective with a single agile class covering a variety of agile concepts. An agile class would still provide students with the high level, application-based course that the tech electives offer. However, this agile course would be much more impactful to the futures of the students. In addition, by replacing the tech elective, the addition of an agile class would come at no cost to the number of credit hours required or the amount of tuition money being paid by students.

The structure of this course would look like a large-scale group team project in which a variety of agile processes and frameworks taught throughout the course could be implemented. Students would use the agile process to develop a large project throughout the course of the semester, and also be taught a variety of concepts and practices that go along with agile. A very important example of this could be pair programming, which is a technique sometimes used in agile development where two programmers will work together at one computer. One programmer, often called the ‘driver’, will use the keyboard and be the one writing the code. The other programmer, called the ‘navigator’, will review the code and provide input. Pair programming is most effective when the programmers switch roles often. Both of the programmers are actively engaged in the coding process, with the intent being to increase the quality of the code through collaboration and peer review. Pair programming is just one of many powerful tools that could be taught to students and implemented hands on during this course.

Not only would this class make students more employable by providing them with a myriad of useful industry skills, but the semester length and structure of classes is a perfect fit to implement an agile process. In fact, scrum sprints (an agile process framework) use a fixed time frame which is perfect for a semester course, and when using scrum, “the average team project is superior in all respects” when compared to previous semesters’ projects using a different development process (Campus Technology, 2018). Agile methodologies in the classroom mimic a professional environment and work schedule, “without overtime or crunch time” (Schroede, Klarl, Mayer, & Kroiss 2012). Another plus of classes mimicking a professional environment is the cost employers save in not having to train new hires in becoming familiar with these processes. Mimicking an agile workforce in the classroom provides benefits for both students and employers and is a huge opportunity for Kansas State to further the employability of its students.

## **Benefit to Curriculum**

### *Deepen Curriculum*

An agile development course would have great benefits on a university curriculum. Deepening curriculum has always been the goal of the computer science department. The field of computer science is a vast and ever-changing field, with a very large amount of sub-disciplines and specialties. For this reason, offering a wide variety of coursework to CS majors is very important in order to allow them to branch out and pursue their specific interests within the field. For this reason, the university has already implemented such classes as game design, intro to artificial intelligence, cryptography, cyber security courses, and others (Kansas State University, 2019). These classes are often chosen as tech electives and provide students freedom in choosing classes to benefit their future. Agile development is becoming such a standard in the industry that it would be a disservice for the university to not offer a course on it. Agile development is already being incorporated into other university’s curriculum, including for two years at the Rochester Institute of Technology (Bergin, Kussmaul, Reichlmayr, Caristi & Pollice, 2015). Implementing a similar course at Kansas State University would have massive benefits to the dept of cirriculum offered to students.

### *Tie into the Industry*

In addition to benefitting the depth of the curriculum itself, it is important for the school to implement this course in order to improve the relevancy of said curriculum. K-State can better market its computer science department by tapping into what the industry is shifting towards, and improve student job placement by gearing courses to meet industry and employer needs. Industry standards should always set the bar for the structure of a departments coursework, as one of the mission statements of the university is that it is “pledged to prepare students for successful employment” (Kansas State University, 2008). To do this, the Kansas State computer science department must align its curriculum with the industry. Figure 4 below represents the industry’s current preference in software development strategy.

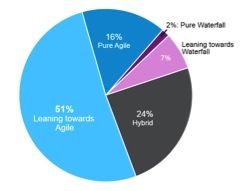


Figure 4. The Industry’s Shift Towards Agile (TechBeacon, 2019)

As can be seen, the industry has shifted strongly towards agile development in recent years. This figure represents a HP survey of 601software development and IT professionals. In this figure, ‘leaning towards agile’, meaning that the professional’s company has primarily adopted agile methodologies, along with ‘pure agile’, meaning that the company only uses agile methodologies, has dominated 67% of the survey. In addition, it is important to note that the ‘hybrid’ category, meaning that a combination of waterfall and agile approaches are used, is over twice as large as both ‘pure waterfall’ and ‘leaning towards waterfall’ combined. This represents such a huge shift in agile that universities cannot ignore. When such large portions of the industry are using agile, this is certainly a methodology and skillset that students will need to be taught before graduation. It is important for universities to recognize the industry’s shift towards agile and align their curriculum to increase student employability. This will improve the curriculum of the department, the marketability of the university, and also provide many useful benefits to students.

## **Benefit to Students**

### *Improve Code Quality*

The first and most obvious benefit to be discussed is the direct benefit that agile methodologies and practices will have on the code students write. Benefits to a student’s projects and code bases is always appealing to both students and the university, and an agile class would provide numerous. Figure 5 below shows some of the most important benefits to a code bases lifespan that students will take out of an agile class, no matter where their career takes them. These benefits include business value, visibility, risk, and adaptability, and would all be biproducts of a successful agile class implementation.

Business value is the immediate production value that can be obtained from a product at a given time in the products development lifecycle. Agile development provides substantially increased business value over this development timeframe due to the concept of minimum viable product that was discussed in the “Description of Agile Development” section. By constantly producing a minimum viable product for the consumer through multiple iterations, the agile approach will constantly provide business value throughout the development lifecycle, while the waterfall approach will only provide business value once a product has finished development, as shown in Figure 5.

Visibility is how easily a product can be seen, inspected, and be given feedback during a development lifecycle. This is incredibly important in the industry when a company wants to produce the best possible product for its client, but it can also be important in a classroom setting when a student is trying to obtain feedback from a professor or TA. Agile development allows for near constant visibility due to the continuous triangle of visibility between developers and clients. As shown in Figure 5, waterfall development only provides visibility of the product at the outset and the conclusion of the development process.

Risk, although not as severe in the classroom, is of serious consideration while on the job. Agile development practices will drastically mitigate risk in a student’s code, and teach them these important practices before they enter the workforce. Agile development has a much lower risk factor than waterfall development due to its iterative nature. Errors are much easier to catch, feedback is received more often, and refactoring and re-evaluating requirements and specifications allows for minimum risk. In a waterfall environment, the testing phase is just one step within a linear process, and the risk grows very high, especially towards the end of a project.

Finally, adaptability of code is a key improvement that agile can bring to the table. Adaptability is a very important skill for all students to learn, and it is something that is incredibly important in an industry like software development. Flexibility is a very important asset when working on a large software project where things are constantly changing, and is one of the fundamental principles of agile. The feedback loop and iterative approach allow for maximum adaptability at all points in the software development life cycle. As shown in Figure 5, waterfall development only has adaptability at the beginning of the development process, because once the team gets half of the way through the linear process, there is no going back. The entire process must be repeated for any sort of changes to the product to be implemented. Adaptability is such a useful skill for both the classroom and professional software developments. As a *Forbes* article summarizing the benefits of agile states, “The key benefit of the agile methodology is that it acknowledges the reality of a software project — namely, that things will change. Adhering to agile-friendly project management frameworks like Scrum keeps changes from disrupting a project and turns them into a positive, both for the project's outcome and the team” (Forbes Technical Council, 2016). An agile development course would allow students to practice this skill before they even enter the industry.

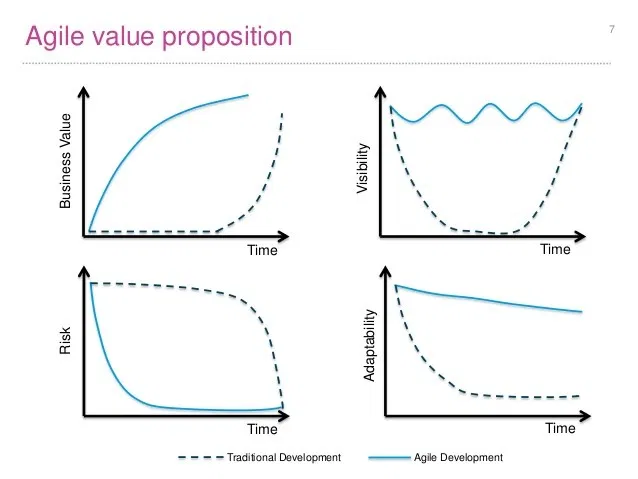


Figure 5. Agile’s Benefits on a Code Base (Flowless, 2015).

Improving upon a student’s coding ability and providing them with a useful kit of skills for any job is one of the most beneficial aspects of an agile class. An agile course would teach them these skills and allow the students to apply them in realistic development team settings. In addition to providing practical improvements to an aspiring software developers arsenal of skills, an agile course would also provide students with a wealth of industry and real-world knowledge that will enable them to succeed in a competitive field.

### *Real World Knowledge*

Perhaps even more important than the coding benefits are the real-world application experiences students would gain from an agile course. The industry is competitive and highly skilled, and a leg up goes a long way for students. Getting as much real-world experience as possible that will make them employable is more important than ever.

One example of these important skills to learn is the ability to work with people. Companies are always looking for “soft skills” in their new CS hires, and as an interview by Campus Technology with faculty and students says, “learning agile instills some of these soft skills in students — abilities in areas such as communication, project planning, and collaboration” (2018). These are skills that will stick with a student forever, and will help them in both their academic and their professional career. Another useful skill that can help students in the profession is the ability to test and review code. Code review and testing is another aspect of computer science that is barely touched on in curriculum, and one that students are often ill prepared for when entering their first job. Projects within coursework often “short change the testing process” and leave students thinking that testing “is an end of the waterfall activity that is often consumed by the need for additional implementation (coding) effort” (Bergin, Kussmaul, Reichlmayr, Caristi & Pollice, 2015). With agile’s test driven design, testing is embedded in the software development process and never feels like it is weighing down the process. These are skills that students will need in the industry, and the current Kansas State University curriculum is not exposing students to these and many other important real-world skills.

Aside from the skills agile development instills in young software developers, knowing agile itself is also becoming increasingly important when it comes to landing a job in the field. With a majority of companies now using the agile process, knowing agile before you graduate is more important than ever. In fact, some companies are even beginning to ask agile-related questions in their interviews, meaning knowing agile is a simple, entry-level requirement for even getting the job! For instance, a student interviewed by Campus Technology stated that during an interview they were asked “to walk through the agile methodology and how it relates to system implementation. My employer focuses on a revised agile methodology they term "breakaway", which engages the client throughout the development process, so I know that agile development will be a part of my career for the near future” (2018). Employers asking agile related questions is a surefire sign that agile is here to stay in the industry. Knowing agile itself is yet another benefit of including an agile course in the curriculum for computer science majors.

Agile provides a variety of useful skills to students graduating into the industry. Soft skills, code reviewing techniques, and becoming familiar with prevalent industry agile techniques are a few of the useful real-world skills that students would gain from an agile course. In addition to this, agile can also instill team-based skills that are incredibly important on the job.

### *Team Skills*

While agile is great for improving one’s coding skills or preparing students for a job, it is equally valuable in teaching essential team skills, which can be used across many disciplines and areas of life. Teamwork is especially valuable in software development, which is a highly team-based field. Software development teams always require the ability to work with people, and an agile development course would be the perfect medium for teaching these skills to students.

Pair programming is one very important skill for a software developer to have, and was mentioned in the Plan of Implementation section. Pair programming is used often at companies with large IT departments utilizing agile development, and it has many benefits and few downsides if used correctly. Pair programming increases the quality of code through the use of ‘programming out loud’: bouncing ideas off of a teammate as you program. In addition, there are much fewer code errors when utilizing pair programming, and this technique also allows for better transfer of information between co-workers. A third benefit to this process is that each programmer can bring unique experiences and intellect to the table when programming. A potential downside to pair programming is that it is sometimes criticized for its increase in required manpower on a project, but an effective pair programming implementation will provide code that’s quality outweighs its cost. In addition, pair programming decreases the amount of coordinating effort required for a team by half, because there are half as many individual workers. Paired programming is a powerful feature that can be implemented in agile development and one of many important team skills students would gain from the proposed course.

Turning CS students who attend class and primarily work alone into team players on a software development team is not easy. While a few meaningful projects exist throughout the CS curriculum, the coursework is not geared towards one of the most important aspect of software development: teamwork. More and more universities have been picking up agile as a means of introducing students to important team strategies and give them real-world experience on a development team. For example, a class at RIT has successfully been using an agile process for the past two years in its semester long project and states “as this course represents the first opportunity for students to participate in a team project, the collaborative nature of agile processes has facilitated the challenges of students transitioning from individually focused developers to contributing team players” (Bergin 2005). Agile development is a great tool for instilling team skills in students, and an agile course would better prepare students for the industry by equipping them with these skills.

## **Costs**

### *Costs to Students*

Although agile development provides numerous benefits to students and can drastically increase their employability, there are several potential costs that must be taken into consideration. The first and most obvious costs that come to mind when considering adding any new class is the costs in terms of credit hours and tuition money for students. Luckily, because of the structure of the proposed plan, these are non-issues. By taking the place of a pre-existing tech elective in the curriculum during the student’s senior year, there will be no additional credit hours added to the curriculum and thus no increase in workload. In addition, there will be no need for any extra charge on the part of the student, because they will be paying for the same amount of CS credit hours. This large issue of money or time that is often brought up when talking about adding a class is not an issue with this proposed plan.

One problem that rises with the addition of this class over a tech elective is the slight reduction in freedom to a students coursework. Tech electives offer a student the choice of any upper level computer science class to count as part of their required coursework, allowing them freedom to choose a career-specialized class that they have interest in. By replacing one of these electives with a mandatory agile class, there is a small reduction in freedom when building their desired coursework. However, an agile class will certainly help all students when they graduate into the field, and the increase in employability from the agile course will likely exceed that of a normal tech elective.

One final risk to be considered when implementing an agile course is the way in which it is implemented. A poorly designed agile course that does not interest or challenge students will potentially end up doing more harm than good. As Schroede, Klarl, Mayer, & Kroiss state, “In particular, it is crucial for students to understand the importance of following a real software development process – an ill-executed lab with a development process gone wrong will only strengthen the all-too-common view that “processes are just a waste of time” (2012). Care must be given when planning this course so that students take well to the subjects introduced and agile practices can be encouraged rather than discouraged. In addition, there is a risk that independent students used to working alone might resist some agile practices due to how those particular students have been taught previously. In one example, “a number of students resist using some of the agile practices that we try to teach them. In particular, strong students occasionally have difficulty accepting test first and pair programming” (Bergin, Kussmaul, Reichlmayr, Caristi & Pollice, 2015). Showing students the benefits of agile development and creating an engaging agile environment are of the upmost importance to avoid these risks to students.

In summary, there are several small potential costs to students. However, these costs are very dependent on how successfully the course is implemented and can be minimized with proper planning and application. Overall, these risks are well worth the reward of implementing a successful class. Using the proposed plan, costs to students is very low, and there are also very minimal costs or risks to the university.

### *Costs to University*

Similarly to potential student costs, the financial costs to the university of implementing an agile course are relatively minimal. If the university converts a potential upper level tech elective course into an agile course, no extra classroom space is necessary. The only potential costs that exists with the proposed plan is the hiring of a potential teacher. However, Kansas State often utilizes adjunct professors who also work in their select industry for teaching select specialized courses in the computer science curriculum. Using an adjunct professor would cost very little compared to a full time professor, and the adjunct professor could provide unique insight in an agile class from their industry experience. Using an adjunct professor who is also a member of a software development team could be perfect for an agile course and also be very affordable for the university. However, there is a very high likelihood that a professor already on the staff in the computer science department would be able to teach an agile course. It is very prevalent industry knowledge and there is a high probability that there are many qualified agile teachers already among the faculty. Other than the potential cost for an additional professor, the proposed plan has very little impact on the university costs-wise, and can provide serious benefits to the marketability and success of the computer science department and curriculum.

# **CONCLUSION AND RECOMMENDATION**

In summary, implementing agile development acts as a win-win-win for students, the university, and the industry of computer science. Agile has massive benefit both to a student’s toolset of skills and to their industry experience and would have much more impact on their career than a tech elective. Students would gain industry skills, team skills, and countless benefits to their coding and software development skills. Some of these coding skills include increased visibility, adaptability, and business value of code. Team skills students gain include working in a team development environment, pair programing, code testing, and the opportunity to work on large code bases. Finally, the students would receive great practiced working in an environment that mimics a professional software development environment. These benefits cannot be understated and this course would have a single greater impact on the success of a graduating student than any course in the current flowchart. The university gains increased marketability of its program, and a furthering of its mission to maximize the employability of its students. The industry benefits from a new wave of agile-educated students that need little training and can contribute more readily to the ever-changing industry of software development. Best of all, this implementation comes at very little cost to either the university or students. Technology moves fast, the industry is changing, and it is time for universities to begin aligning their curriculum to serve their students. This agile course would be an adaptive solution and a powerful educational tool for the Kansas State University computer science department.

In conclusion, agile would be easy to implement in the current Kansas State curriculum with little change needed, and would provide massive benefits to students graduating into a highly competitive field that is shifting increasingly towards agile. For this reason, I recommend the implementation of a single agile course into the computer science curriculum of Kansas State University in order to facilitate the benefits agile would have on students and the university. This course could be implemented seamlessly in place of a tech elective, and would produce a change that will have lasting impacts on Kansas State University and its students for years to come.

References

AgileAlliance. (2020). *Agile 101.* Retrieved February 18, 2020, from <https://www.agilealliance.org/agile101/>.

Bergin, J., Kussmaul, C., Reichlmayr, T., Caristi, J., & Pollice, G. (2005). Agile development in computer science education: Practices and prognosis. *ACM SIGCSE Bulletin,* *37*(1), 130-131.

CampusTechnology. (2018). *An agile team for teaching agile development.* Retrieved February 18, 2020, from <https://campustechnology.com/Articles/2018/05/14/An-Agile-Team-for-Teaching-Agile-Development.aspx?Page=2>.

Flowless. (2015). Agile Value Proposition. Retrieved March 23, 2020, from

http://www.flowless.eu/agile-value-proposition/.

Forbes Technical Council (2016). The benefits of using agile software development. *Forbes.*

Kansas State University (2019). *Academics: Computer Science.* Retrieved February 27th, 2020, from <https://courses.k-state.edu/spring2020/CIS/>.

Kansas State University (2008). *Kansas State University mission.* Retrieved May 6th, 2020, from <https://www.k-state.edu/about/mission/>.

Schroeder, A., Klarl, Mayer, & Kroiss. (2012). Teaching agile software development through lab courses. *Proceedings of the 2012 IEEE Global Engineering Education Conference (EDUCON),* 1-10.

Search Software Quality. (2019). Waterfall model. Retrieved April 24, 2020, from <https://searchsoftwarequality.techtarget.com/definition/waterfall-model>.

Scrum.as Academy. (2019). Aspects of agile approaches. Retrieved March 23, 2020, from <https://www.scrum.as/academy.php?show=2&chapter=3>.

TechBeacon. (2019). Survey. Is agile the new norm? Retrieved April 27, 2020, from <https://techbeacon.com/app-dev-testing/survey-agile-new-norm>.