Developing Software Using Agile Scrum

College of Engineering and Technology
East Carolina University
Greenville, NC, USA

Abstract—This paper considers what the Agile Scrum software development methodology is, what benefits it has over other methodologies, and how its implementation has become a highly utilized method amongst software development teams. The paper discusses where Agile originated, from the initial meeting in Utah, to the various methodologies that used the core foundation. Also discussed are the different roles used in scrum and how they facilitate a modern efficient guide to software engineering in today's ever changing landscape of collaborative work in designing new software to meet the demands of the customer. The paper also discusses how effective teaching scrum to programmers in school with the help of coaches is, and how doing so improves the entire process of working as a team to create and deliver software that customers are looking for.

Keywords—agile methodologies; Scrum; sprint; efficient; manifesto; backlog; flow; project; team

I. INTRODUCTION

Of the many agile process methods being used in modern software development, the focus of this paper is the Scrum method. "It is based on the notion that the development process is unpredictable and complicated, and can only be defined by a loose set of activities. Within this framework, the development team is empowered to define and execute the necessary tasks to successfully develop software [3]." Developed in the early 1990's scrum has become a widely popular software development method.

II. BACKGROUND

According to Merriam-Webster, the word “agile” means "marked by ready ability to move with quick easy grace;" or "having a quick resourceful and adaptable character [1]." In February 2001 the organizational anarchists, a group of 17 of the top leaders in software development convened in Snowbird, Utah for for two days and created the Agile 'Software Development' Manifesto. This manifesto centered around four core values: Individuals and interactions over process and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan. In addition to these four core values the authors outlined a guiding set of twelve principles which reinforced the manifesto:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly [2].
The Dynamic Systems Development Method (DSDM) was released in 1994. It is used in software development as well as non information technology solutions and consists of eight principles: focusing on the business need, on time delivery, collaboration, not compromising quality, incrementally built off firm foundations, iterative development, continuous clear communication and demonstrative control [6].

Lean Software Development (LSD) was written by Mary and Tom Poppendieck and was introduced in the book *Lean Software Development* in 2003. This method incorporates the Lean manufacturing and Lean Information Technology (IT) principles and applies them to software development. It can be characterized by seven principles: waste elimination, amplified learning, deciding as late as possible, delivering as fast as possible, team empowerment, integrity built in, and seeing the whole [7].

Kanban was created by David Anderson in 2004, and was based off of an inventory control system used in Japan in the 1940’s. The word kanban means “visual sign” or “card” in Japanese and promotes continuous, small changes. It consists of five principles: workflow visualization, limiting of work in progress, managing and enhancing the flow, explicit policies, and continuous improvement [8].

Crystal Clear was created by Alistair Cockburn. It was released in his book *Crystal Clear: A Human-Powered Methodology for Small Teams* in 2004. This methodology is part of the Crystal family of methodologies. This method is intended to be used with teams of six to eight developers, with the focus on the people, not processes or artifacts. It has three requirements: users given frequent usable code, reflective improvement, and osmotic communication [9].

Lastly, there is the Scrum methodology, which is based off of the agile process. Scrum emphasizes decision making from real-world results rather than speculation. Time is divided into short work cadences, known as sprints, typically one to two weeks long. The product is kept in a potentially shippable (properly integrated and tested) state at all times. At the end of each sprint, stakeholders and team members meet to see a demonstrated potentially shippable product increment and plan its next steps [19]. A typical scrum flow can be seen in Figure 2.
The Scrum Team is comprised of six to nine team members. The team works on the project together, helping each other. This is similar to how a sports team operates, they work together playing on each members strengths to work as one. Doing so brings out the togetherness of the team. Scrum Teams do not specifically follow the traditional development team roles such as designer, tester, or programmer. The team is responsible for deciding what each sprint is trying to achieve and how much work they feel they can get done for each iteration.

A. Roles and Responsibilities

The Scrum method defines three roles: Scrum Master, Product Owner and Team, which is composed of the team members. The Scrum Master is comparable to teams head coach, who directs the team members in a way that lets the team members put forth their best possible work. Duties performed by the master include: organizing meetings, handling challenges and roadblocks, and working with the product owner to ensure the project backlog is handled in way that makes the project ready for the next sprint. It is important to know that while the Scrum Master ensures the team is following the scrum process, they do not really have any authority over team members. They do however have authority over the process. As an example, the Scrum Master cannot specify what a team member has to do, rather they can specify the sprint cadence.

The Product Owner is the person responsible for the overall vision of the project. They handle all of the business and market requirements and lay out what needs to be accomplished. This includes building the backlog, deciding what features need to be ready to go out next, and interacting with team members and stakeholders. One of their most important duties is to ensure that everyone knows what items are in the backlog. An important distinction is that the Product Owner is not a project manager. They do not focus on managing the project’s progress, rather they focus on motivating the development team to achieve the goal, providing the the project vision.

B. Scrum Process Steps

The scrum flow is comprised of six things: product backlog, sprint planning, backlog refinement, daily scrum meetings, sprint review meetings, and post sprint meetings.

1) Product Backlog: The product backlog is comprised of requirements and user stories that the Scrum Team members and Product Owner have decided on. These items are prioritized into what needs to be completed during each sprint, and include desired product features and functions, see Figure 4.

2) Sprint Planning: Before a sprint begins, the team is presented with the items of highest precedence on the backlog by the Product Owner. As a team the group determines which items they can accomplish during respective sprint. Once decided upon, these items are moved from the product backlog to the sprint backlog. These will be the items that need to be completed in each sprint.

3) Backlog Refinement: At the conclusion of each sprint, the team meets with the Product Owner to discuss and make sure their backlog is ready for the next sprint. During this meeting user stories not needed can be removed, new stories can be created, and the prioritization of these stories can be reassessed. The refinement of the backlog ensures that only items that are relevant to the projects completion are included.

4) Daily Scrum Meetings: The daily scrum is a fifteen minute meeting where all the team members talk about any issues that have come up, as well as discussing their goals. Hence the name, this is a meeting that happens every day of the sprint with the objective of keeping the team on track.
5) Sprint Review Meetings: After a sprint has concluded the team meets and presents what has been completed. During this meeting live presentations of the project are typically demonstrated.

6) Post Sprint Meetings: Like the Sprint Review, the Post Sprint meetings take place at the conclusion of each sprint. The team discusses how the scrum is going and what they should look at changing in the next sprint. This is typically where the team members can talk about things that did not go according to plan and how they can do things differently.

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Study</th>
<th>Design</th>
<th>Development</th>
<th>Test backlog</th>
<th>Integration testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Scrum Board
Source: Quoted and edited from http://atourom.com/?p=10

C. Scrum Tools

The scrum method incorporates some tools that help visualize the backlog. These tools include: scrum boards and user stories.

1) Scrum Board: scrum boards help teams see the sprint backlog with a scrum task board. This board is usually index cards or the use of a white board divided into several columns, a typical board can be seen in Figure 5. Items on these boards usually include: backlog, team assignment, study, design, development, backlog testing and integration testing. As the team accomplishes items they are moved into respective columns.

2) User Stories: user stories are descriptions of what features customers want in their software. These stories include what type of user it is, what they will want, and why they need it. With this information the programmers can write code tailored to the needs of each story. User stories are one of the strengths of agile. They can be written with largely different amounts of detail depending on what functionality the end product needs to have. Typically they are broken down in a couple of ways, splitting them up into smaller user stories or by adding satisfaction conditions. When splitting larger user stories up into smaller stories it allows for more detail to be added. Satisfaction conditions are basically high level acceptance tests.

IV. Scrum in Practice

Because Scrum acts as a wrapper around existing development processes, it can work with any existing methodologies you follow [16]. Some developers new to scrum are sometimes apprehensive about the process. The change in roles and responsibilities is often seen as uncomfortable and can have an adjustment period. Another reason developers have reservations about implementing the practice is the worry of who would provide the teams direction, and if it would hinder creativity freedom. One of the key motivations for moving to scrum is its ability to create a sustainable pace for the development team.

The pros of implementing scrum include saving time and money, keeping the team focused on having regular meetings, open communication, which makes collaboration a lot more efficient, tests being completed frequently, and lastly, a great feedback mechanism which makes possible changes a lot more beneficial to the making the product successful.

Like anything else, scrum does have cons. Some cons of using scrum include team members who are not pulling their weight. Some team members can fall behind because of illness or just not being very productive. Other team members can see this as negatively and it can have a detrimental effect on the team cohesiveness which can ultimately bring the teams efficiency way down. Another potential con of using scrum is not picking the correct team size. Smaller team sizes tend to work more effectively because there is less time trying to get everyone on the same page. Other cons of using scrum is team members not willing to be flexible, friction during meetings, and stress.

V. Case Studies

Journal articles and software development papers have shown that many large companies like Intel, Microsoft, Nokia, and Yahoo! have adopted scrum. The common and most critical issues in using scrum are integration and incomplete requirement analysis. Scrum is a well-known agile approach for rapid solution development but it contains integration complexities. The integration complexities lessen the quality of overall solution. Therefore, there is a need to enhance the process quality assurance of Scrumb model to develop a more reliable and flexible product with increased customer satisfaction [20].

Scrum use in the software industry has increased incrementally from 37% of companies using it in 2007 to 66% in 2013, resulting in a rapidly widening gap between the skills demanded by industry and those taught in the academy. Thus, universities need to effectively provide students with the skills needed to succeed in current software organizations. There is a growing need to teach students agile, scrum specifically. Issues that arise from implementing this include: students not having adequate skills, requirements documents that are too lengthy which tends to lend itself to more of the waterfall-like issues.

A comparison was conducted between course offerings with and without coaching; students' performance and their perceptions of using Scrum for the first time were evaluated.
Student performance was measured across a set of recommended software engineering practices, widely performed throughout a software development life cycle. Student perceptions were evaluated by administering a survey that assessed the impact of agile coaching on the training model. The results showed that incorporating an Agile Coach allowed students to increase their coverage of software engineering practices. The majority of students in both groups found the course useful and interesting, but to a statistically significant degree, the coached students perceived that their gain in nontechnical skills to be higher than did non-coached students. Finally, there was no significant difference between both groups of students in terms of the number of physical meetings needed. The main findings obtained from this research work are that students found the course useful and interesting and the Scrum-based process easy to follow. The survey results indicate that agile-coached students gained more nontechnical skills than did non-agile-coached students [21].

The conclusion of the survey training model which had enriched agile coaching showed encouraging results. Students with agile coaching improved both their technical and nontechnical skills in software development. The survey also concluded that courses taught with agile coaching where able to utilize scrums faster solutions to problems as well as preparing them for working in teams, improving their communication and project management. While this particular case study showed promising results, the sample size and use of a survey rather than a test may have skewed the results.

 Likewise, the survey may have been skewed because of how the agile coach role was handled. Different teachers have different methods and as such some teachers are able to educate students at a higher level than others. Students in this survey both agile coached, and non-agile coaches were able to handle several challenges like projecting workload estimates and non-standard/effective coding practices. Overall the study showed that students learning experiences were enriched, Figure 6.

In another case study involving a project that had a main application and eight web-based subsystems. This project had roughly 50,000 lines of code which is about the size of typical mid-sized systems. Using scrum the team was able to produce and use more reliable planning to meet stakeholders expectations.

VI. SUMMARY

In today’s day and age of software development, companies looking to hire software engineering teams to write software for their business needs expect efficiency and products that meet their expectations. While there are many different types of methodologies, from the traditional waterfall method, to XP, to Crystal Clear, Scrum has made its impact on all of them.

Utilizing the four core values and twelve principles the organizational anarchists decided upon in their manifesto, one can see that almost all current methodologies in software development utilize them in one aspect or another. Particularly in Scrum it can be seen that these foundations are implemented throughout the process. The roles is scrum play a vital part in making sure the team stays on task and works with the customer to develop the end product that they are expecting. Through constant sprints, meetings and refinements the teams that use scrum work together, giving the team members a feeling of working as a team, playing of each others strengths and weaknesses to achieve the goals that they have set forth, just like the sport rugby from which they got their namesake.

It is important to recognize the tools that scrum teams use in their development as these tools allow the teams to focus on different aspects of detail in a well-structured way so as to not miss any details which could negatively effect the project they are working on. The importance of having a product owner on the team is largely important to ensure the product that is being developed meets all of their requirements, having this person around cannot be understated. While the scrum master does not have direct control over individual developers creativity their role on the team is vital. Being able to see what each team member's strengths and weaknesses are and motivating them in the appropriate direction is also vital.

Lastly, looking at teaching the agile scrum process to you software developers through coaching is important. Doing so allows these developers to learn how to work in teams as they move on into the workforce where it is easy to fail because of not having quality experience working with different people who may have differing views or coding tendencies. By teaching them at the beginning of their coding career they can only add it to their tool bag to use throughout their career and as such will be better suited to handle situations that can come up when working on new projects.

Figure 6: Results of agile coached students vs. Non-agile coached students

VII. REFERENCES


