Application of Scrum Methods to Hardware Development

An overview on how to run a hardware development project using the Scrum framework within the Agile software development methodology

Backblaze Inc.
July 2015

■ INTRODUCTION ■

A road-legal car that gets more than 100 miles per gallon is sure to turn a few heads. Add to its list of specs a top speed of 149 mph, zero to sixty in five seconds, and a sleek chassis, and you've raised some eyebrows. But there's one stat on this car that drops jaws: the prototype took only three months to develop. Using Agile software development methodology and applying it to their hardware development process, the Wikispeed team was able to cut years from the automobile development timeline without sacrificing quality.

Although iterative, incremental development principles were first used as early as the 1950s, they became more common in the 90s. In 2001, a group of seventeen software developers met to discuss these lightweight development methods. The result of that meeting was the Manifesto for Agile Software Development, a collection of values and principles by which Agile teams to this day organize, collaborate, and develop products.

Since then, a myriad of software development methodologies and process frameworks have been created that follow these principles and are therefore considered Agile methodologies. Agile methodology, in all its forms, was conceived for the purpose of developing good software; however, developers have begun thinking outside the box and applying Agile methodology in hardware development processes.

The most common process framework that applies Agile methodologies is known as Scrum. It is widely praised for its usefulness in greatly increasing the productivity, efficiency, and quality of the work of software development teams. Using Scrum for hardware development is not as widely appreciated; like other forms of Agile, Scrum has many critics who claim it has a neutral or even detrimental effect on the hardware development process.

Still, many Scrum teams have found success in using Agile methodologies for the purposes of hardware development. This document serves the purpose of describing the basic building blocks of the Scrum framework and exploring the challenges and benefits of applying it to hardware development.

■ DEFINITIONS AND PURPOSE ■

Agile is a product development methodology guided by a set of four values and twelve principles that emphasize frequent production of working products, continuous communication with customers and among team members, and flexibility under changing requirements.

Agile methodology is used to develop products in increments through many iterations of the production process. It is most effective when projects require a lot of creativity, can be broken into many small tasks, and must be designed and developed to fit a dynamic set of expectations. When applied to such projects, the goal of Agile methodologies is to optimize the development process by significantly increasing flexibility, communication, and production.

Scrum has proven to be one of the more popular process frameworks that lends practicality to these abstract Agile principles.

■ THE SCRUM TEAM ■

Usually a Scrum team (often referred to simply as a Scrum) is composed of five to nine members. When a team has twelve or more members, it is split into smaller Scrums that work independently, but communicate frequently throughout the duration of the project. There are three roles in a Scrum team: one product owner, one Scrum master, and several members of the development team.

On the business and communications end of the Scrum team, the product owner is the voice of the customer. He or she is responsible for maintaining the requirements of the project and communicating with customers to ensure the team is developing the right product.

The Scrum master is not a traditional project manager like the word "master" might suggest: this position has no authority over other team members. Instead, the Scrum master serves to facilitate the success of the team by removing impediments to communication and technical progress, and even addressing problems caused by the outer organization. The Scrum master’s function, in short, is to pave the way for the Scrum process to progress as intended.

The remaining Scrum members form the development team, between three and nine individuals who are responsible for completing the tasks at hand. Teams are cross-functional and self-organizing, meaning each member is capable of a number of tasks, without being delegated each task by a manager.

■ THE SCRUM PROCESS ■

A Scrum project begins when the product owner, the development team, or the stakeholders create a product backlog, a to-do list for the project based on the customer’s needs.

After the backlog is created, a meeting for product backlog refinement occurs. During this meeting, Scrum team members contribute to making the backlog the most effective it can be by adding and removing items, breaking down complex tasks into smaller components, organizing existing items by order of importance, estimating items’ business value, clarifying the requirements of each item, and determining which items are ready.

The items at the top of the backlog are clarified and prepared first, while other items may remain less distinct until the items above them are completed and removed from the backlog. Product backlog refinement is a continuous process, taking place throughout all stages of Scrum; therefore, items that are indistinct at the beginning of the Scrum process have become more clearly defined by the time they need to be addressed.

Next, Scrum projects are divided into sprints, iterations of work repeated multiple times over the course of a project. A typical sprint lasts between 1 and 4 weeks. The number of sprints in a project is decided based on the amount of work to be done and the team’s rate of task completion, if it has been measured.

The sprint is structured around four kinds of meetings called ceremonies, and a fifth that occurs when large projects require two or more Scrum teams. The first is a sprint planning meeting. This ceremony occurs at the beginning of each sprint, and is limited to two hours of planning for each week of the sprint. During this meeting the top items are taken from the product backlog to form a sprint backlog, a to-do list for the sprint. The product owner and development team then collaborate to break down, clarify, and estimate the business value of each item on the sprint backlog until it is sprint-ready.

Every day during a sprint begins with the daily Scrum meeting. This fifteen-minute ceremony is sometimes called a stand-up because teams are encouraged to remain standing to keep it brief. During the daily Scrum, each team member answers the following three questions:

1. What did I do yesterday that moved the team closer to reaching the sprint goal?
2. What will I do today that will move the team closer to reaching the sprint goal?
3. What impediments do I see that might prevent the team from reaching the sprint goal?

At the end of each sprint, a sprint review is held. The sprint team invites the customers and stakeholders to this two-hour ceremony, and together, they review work that was completed during the sprint, as well as any work that was planned but not completed. Stakeholders and customers are invited to offer feedback, which the Scrum team considers during planning for future sprints.

The final ceremony of a sprint is known as the sprint retrospective. This one-and-a-half hour meeting gives the team a chance to look back on the sprint without customers or stakeholders present. The development team and Scrum master discuss the sprint process, and agree upon ways to improve it. They answer the following questions:

1. What went well during the sprint?
2. What could have gone better?
3. What can we do to improve the process?

When Agile methods are used for large projects that require multiple teams, a fifth meeting is held after each team's daily Scrum that is known as the Scrum of Scrums. This ceremony occurs sometimes daily, sometimes as little as once or twice per week, depending on the team's needs. Each Scrum team sends an ambassador to this meeting to ask parallel questions to those asked during the daily Scrum:

1. What did my Scrum complete since the last Scrum of Scrums?
2. What will my Scrum do before the next Scrum of Scrums?
3. What impediments are preventing my Scrum from making progress?
4. Is my Scrum creating impediments that might prevent other Scrums from progressing?

■ BENEFITS OF SCRUM FOR HARDWARE DEVELOPMENT ■

Many of the benefits of using Scrum, whether for software or hardware development, are clear upon gaining even a basic understanding of the process. Customers and developers alike profit from the Agile focus on flexibility, communication, and production.

Some of the most universal positive results of applying Scrum come from its focus on active collaboration among all parties involved. Even disciplines that cannot apply the iterative, incremental structure of Scrum can benefit from applying Agile communication principles. Not only does Scrum give customers a much clearer view of the progress being made on their product, but it facilitates much-needed positive team-building in a discipline that is often operated from solitary cubicles.

The obvious benefit of increased flexibility that comes from the sprint structure provides another, potentially less obvious perk. When space is made to adapt to changing expectations and requirements, the development process results in better hardware. Traditional methodologies rely on a full set of requirements laid out at the beginning of the development process. If those requirements change, the result will be a less-than-satisfactory product. Teams also have the option of drawing up a new plan to fit the new requirements, but if they choose this course of action, the time and money they spent working on the first plan is wasted. Agile methodology makes it possible to improve upon an original hardware design without significantly increasing spending and time required.

Not only will hardware designs improve, but the hardware development process itself will improve as Scrum progresses. As teams meet at regular intervals, they are encouraged to discuss pros and cons of the way things are done, and they are enabled to change elements of the process to facilitate the most efficient, effective development possible. Over the course of an entire project, the sprint process can be refined enough to improve drastically on the quality and efficiency of the work done.

The list of Scrum's benefits for hardware development goes on. Every member of the cross-functional development team is enabled to contribute to any part of the project. The sprint structure imposes time constraints that encourage more efficient hardware development. Agile's positive, team-focused values create freedom to make mistakes, fostering more healthy risk-taking, less blame-shifting, and more reward in the long run.

■ Challenges of Scrum for Hardware Development ■

Despite the many benefits to using Scrum for hardware development, it is important to remember that Scrum was originally conceived for the purpose of software development and therefore comes with a few challenges to developing hardware. It is vital that teams address these challenges before applying Agile methodologies to the hardware development process. Facing challenges early affords teams the opportunity to minimize the limitations they might come across and ultimately maximize the benefits of Scrum.

At the top of the list of challenges often faced by hardware Scrum teams is a demand for time. Hardware development usually requires a good deal more time than software development, and the one- to four-week sprint structure often proves too short for the smallest increments of progress possible in a hardware development project.

It’s also difficult for many hardware projects to frequently release working products, though it’s a high priority in the principles of Agile. For one thing, hardware development is expensive. Delivering a functional piece of hardware to the customer after each iteration and then adapting the design to the customer’s changing specifications increases the cost of a project dramatically. For projects working with especially expensive materials, traditional methodologies often involve much less financial risk.

A lack of modularity also makes it difficult to use Scrum. Software is usually simple to divide into many small tasks to be added to the product backlog, but hardware often doesn’t have the modularity that makes dividing projects into small increments possible.

■ Adapting Scrum to Hardware Development ■

The challenges of applying Scrum to hardware are considered insurmountable by some, but many teams have found ways to adapt Scrum to hardware development. When considering using Scrum, it’s crucial to acknowledge that, for some teams and for some projects, Agile methodologies are simply not possible. However, in most cases, Scrum can be adapted to fit hardware development projects.

One of the most useful solutions that Scrum teams have found in working with hardware is adjusting the frequent release principle. Instead of delivering a functional, physical product to the customer at the end of each sprint, teams have opted to deliver virtual simulations of the result of the sprint. This provides a reasonable solution for many of the challenges of using Scrum for hardware.

It saves money, since a virtual representation of a product is typically less expensive than the physical product itself, and changing the design or representation of a piece of hardware is also vastly less costly. It saves time, since the time it takes to manufacture custom hardware components is completely eliminated from each sprint. And it solves the problem of modularity, since delivering the product at that point would simply mean showing the customer the progress that the team has made during the sprint. Delivering virtual increments may not provide the same high level of customer satisfaction that functional product does when Agile is applied to software development, but the problems it solves may be well worth the trade-off.

Another solution often used is to focus each sprint on a particular component of the hardware being designed. Once the components are defined, additional sprints can address the integration of these components.

There are many other solutions to make Scrum accessible to hardware development teams. Making early prototypes with more efficient methods and inexpensive materials, or sacrificing physical prototypes in favor of simulated ones cuts down on time and money. The sprint development framework combined with the Scrum master's problem-solving role and the emphasis on communication results in a more efficient hardware development model, and saves a lot of time before any adjustments are made to the Scrum process. And it's worth considering simply working to come up with more modular hardware designs; if it's not possible to fit Agile to the project, maybe it's worth considering fitting the project to Agile.

In the end, the solutions that hardware development teams need in order to make Agile methodologies work for them are often specific to their particular project. Every team is different, and therefore requires different adaptations to Scrum.

■ CONCLUSION ■

When hardware development teams consider implementing Agile methodology, it's common for them to think about all the values and all the principles as though the changes will be useless if one principle is left out. Since there are Agile principles that tend to disagree with effective hardware development, then, teams often throw the baby out with the bath water and stick with traditional linear methods.

Instead, teams could benefit from viewing each value and each principle of Agile methodology separately, evaluating their usefulness in the context of hardware development, and choosing to implement a few principles that have the greatest chance of benefiting the team's effectiveness. The principles of Agile methodology (and, by extension, the specific methods used in Scrum) are not a strict set of rules meant simply to be followed; but a flexible, dynamic set of principles meant to be evaluated and shaped to fit a team's needs.

Can a hardware development team effectively use Scrum? Yes, if the team has a clear understanding of their specific needs and how to make Agile work for them.

■ SOURCES ■

1. Reynisdóttir, Þórdís. "Scrum in Mechanical Product Development." Chalmers Bibliotek: 2013. http://publications.lib.chalmers.se/records/fulltext/191951/191951.pdf
2. "The Agile Hardware Design Mindset." Innovel, LLC. 26 February 2014. http://www.innovel.net/?p=427
3. Sigel, Sara. "Pros and Cons for Agile Hardware Product Development." GrabCAD. 14 May 2014. http://blog.grabcad.com/blog/2014/05/14/pros-cons-agile-hardware-product-development/
4. Johnson, Neil. "Agile Hardware Development: Nonsense or Necessity?" EETimes. 10 October 2011. http://www.eetimes.com/document.asp?doc_id=1279137
5. "A Heretic Speaks (Why Hardware Doesn't Fit the Agile Model)." AgileSoC. 30 September 2012http://www.agilesoc.com/2012/09/30/guest-blog-a-heretic-speaks-why-hardware-doesnt-fit-the-agile-model/
6. Maccherone, Larry. "Top 10 Questions When Using Agile on Hardware Projects." 23 February 2010. http://maccherone.com/larry/2010/02/23/top-10-questions-when-using-agile-on-hardware-projects/
7. Cerisier, François, and Mike Bartley. "Are We Too Hard for Agile?" Design & Reuse. http://www.design-reuse.com/articles/32442/are-we-too-hard-for-agile.html
8. "Manifesto for Agile Software Development." 2001. http://www.agilemanifesto.org/
9. Johnson, Neil. "Applying Agile to Hardware Development." Agile Alliance. 2011. http://www.agilealliance.org/files/session_pdfs/applying%20agile%20to%20ic%20development.pdf
10. "Scrum (Software Development)." Wikipedia. 2015. https://en.wikipedia.org/wiki/Scrum_(software_development)
11. "The Basics of Scrum: An Introduction to the Framework." ScrumInc. 2014. http://www.scruminc.com/wp-content/uploads/2014/06/The-Basics-of-Scrum.pdf
12. Montoya, Monte. "Challenges of Adopting Agile in Combined Hardware and Software Environments." CPrime. 1 August 2012. https://www.cprime.com/2012/08/challenges-of-adopting-agile-in-combined-hardware-and-software-environments/

13. "The Agile Times, Volume IV, Issue 1." Renaissance Software. 2004. http://www.renaissancesoftware.net/files/articles/AgileAllianceNewsletterVol4.pdf
14. Denning, Steve. "Wikispeed: How A 100 mpg Car Was Developed in 3 Months." Forbes. 10 May 2012. http://www.forbes.com/sites/stevedenning/2012/05/10/wikispeed-how-a-100-mpg-car-was-developed-in-3-months/
15. Levison, Mark. "Agile for Hardware and Embedded Systems." Agile Pain Relief Consulting. 12 December 2008. https://agilepainrelief.com/notesfromatooluser/2008/12/agile-for-hardware-and-embedded-systems.html#.Vamg2flViko
16. Graves, Eric. "Applying Agile to Hardware New Product Development: Part 3." Playbook HQ. 31 December 2014. http://playbookhq.co/blog/applying-agile-hardware-new-product-development-part-3/
17. Johnson, Neil. "Hardware Development and the Agile Manifesto." LinkedIn. 1 December 2014. https://www.linkedin.com/pulse/20141201172054-5203931-hardware-development-and-the-agile-manifesto
18. "Using Scrum Methodology for Hardware Development." LX Group. 6 October 2014. https://lx-group.com.au/using-scrum-methodology-hardware-development/
19. Ovesen, Nis. "The Challenges of Becoming Agile." The Inventive Step. 2012. http://www.theinventivestep.net/Ovesen-2012_The-Challenges-of-Becoming-Agile_WEB.pdf
20. "Core Scrum." Scrum Alliance. 15 August 2014. https://www.scrumalliance.org/why-scrum/core-scrum-values-roles
21. James, Michael. "The Backlog Refinement Meeting (or Backlog Grooming)." CollabNet. 15 April 2011. http://blogs.collab.net/agile/the-backlog-refinement-meeting-or-backlog-grooming#.VaqJPflViko
22. "The Scrum Master Role." Scrum Methodology. http://scrummethodology.com/the-scrummaster-role/
23. "Scrum of Scrums." Agile Alliance. 2013. http://guide.agilealliance.org/guide/scrumofscrums.html