What is the Difference Between DevOps and Scrum?[1, 2]

Brian Vanderjack

As a professor in the graduate program at Illinois Institute of Technology, in the Information Technology Management department, I teach Information Technology (IT) team leadership, mainly via scrum and waterfall. During class discussion last semester, one of my top students asked, "What the difference was between Scrum and DevOps?" This article will capture the answer that I provided to this student and will answer this question for others who have the same question. The outline I will use to address this question is:

- What Agile is (and what it has to do with Scrum and DevOps);
- What Scrum is;
- What DevOps is; and
- What the difference is between DevOps and Scrum.

This is an important discussion become Scrum is now recognized as a significant contributor to success in the IT space and DevOps is becoming a key component of adding business value in the IT industry. Having IT people read this article could introduce them for the first time to DevOps and Scrum, encouraging even more organizations to employ Scrum and DevOps as competitive weapons in the marketplace.

What is Agile, and what does it have to do with Scrum and DevOps?

"Agile is credited for 'dramatically' increasing the productivity of many (software) development organizations" (Kim, Humble, & Willis, 2016, p. 5). The seminal document that defines the Agile movement, quoted below in its entirety, is hosted on the website AgileManifesto.org (2001):

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

[1] Provided "use at your own risk"

[2] How to cite this article: Vanderjack, B. (2019). What is the Difference Between DevOps and Scrum? *PM World Journal*, Vol. VIII, Issue VII, August.

That is, while there is value in the items on
the right, we value the items on the left more.

(authors) Kent Beck, Mike Beedle, Arie van
Bennekum, Alistair Cockburn, Ward
Cunningham, Martin Fowler, James Grenning,
Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon
Kern, Brian Marick, Robert C. Martin, Steve
Mellor, Ken Schwaber, Jeff Sutherland, and Dave
Thomas

There are a few development methodologies used in IT to achieve the above standards, and Scrum is one of these methodologies.

On the next web page of the Agile Manifesto, there are 12 "principles." One of these principles opened the door for "DevOps," which aims to "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" (Beck et al., 2001). I have underscored the words in the relevant principle (above) that can be considered to have launched the DevOps movement.

So, what can you tell me about scrum?

The main idea of Scrum is summarized by Ken Schwaber (2004, p. 1): Scrum was devised to "wrestle usable (software) products from complex problems." Scrum's core tenants are (a) transparency (known information is shared with relevant stakeholders, clients, and decision makers); (b) inspection (clients inspect software deliverables frequently for accuracy and to ensure new software supports business needs); and (c) adaption (new concepts and ideas are rapidly identified and woven into the work product, especially when defects are observed (Schwaber, 2004, p. 3).

Scrum is a method of accomplishing software delivery by leveraging the aspirations of the Agile Manifesto. The high-level Scrum process flow is summarized as follows.

1. Business needs are deconstructed by a business analyst (in Scrum, we call this person a "Product Owner") into small requirement bits, called "User Stories."

2. User Stories are stored in a queue, called a "product backlog," and prioritized by the Product Owner based on business value, business need, time needed to build it, etc.

3. User Stories are then inspected and improved by the developers.

4. Iterations are at the heart of Scrum (Schwaber, 2004). Iterations are blocks of time (typically 2-3 weeks), each of the same length, during which User Stories are worked on, in their entirety, by the development team (the development team is composed of coders and testers). A select number of User Stories are pulled into an iteration to be worked on, in priority order, by the development team.

5. Each time an iteration is completed, the development team returns to the prioritized queue of User Stories (the "Product Backlog") and selects more work to do from the top of the queue (the top is where the highest priority User Stories are located).

6. The team has all their work approved by the Product Owner.

7. At the end of each iteration:

    - The software work product is displayed to interested/relevant stakeholders by the team during a demonstration (called a "demo").

    - The team also gathers together, in a family like setting, to determine how they can keep getting better; this is called a retrospective.

8. Then, in a non-DevOps environment, the software waits on a release for its deployment. *Please note, this is just like putting a gourmet dinner into a refrigerator for a few weeks before serving it!*

With respect to step 8, it is a bad thing to have software that is ready to deploy sitting in a refrigerator ready to deploy because one of your competitors can beat you to the market place and steal any advantage that you may have by being first to market. Also, if the software adds value, it will only add value once it is deployed. In the "refrigerator," waiting for a release, it adds no value. So, the question becomes, how do we get our software to users faster?

DevOps to The Rescue!

The problem that we now need to solve, one that Scrum does not address, is how to minimize the amount of time our working and value adding software spends in the refrigerator getting old and worthless. DevOps directly addresses this.

According to Kim, Humble, and Willis (2016), two underlying principles that lead to the DevOps advantage are (a) reducing the batch sizes of work deployed to the customer and (b) reducing the intervals between work products being deployed (p. 11). These well-respected authorities on DevOps indicate that following these steps leads to:

- increasing throughput (Kim, Humble, & Willis, 2016, p. 12)—that is, getting work products out to the market fast to begin harvesting the value of the deliverables as soon as possible, and

- boosting the performing organization's ability to "out experiment" the competition (Kim, Humble, & Willis, 2016, p. 12); that is, by seeing how customers actually react to new software, your marketing organization can gain a huge advantage in determining what new features to add with future software deployments.

The above principles lead us directly to two of the core functions in DevOps: Continuous Integration (called "CI") and Continuous Deployment (called "CD").

- CI refers to the idea that a developer can "check out" a well contained/limited amount of software from a "main branch" of code (this main branch represents an entire application's software), make a change, test it, get the client's approval of the software upgrade, "demonstrate" the new functionality to interested stakeholders, and then add it right back to the main "branch." The idea behind CI is that coders are continuously updating the main branch (the base application) with the most up-to-date code available moments after the software is approved.

- CD refers to the idea that once the software is approved and added back to the main branch of the application, it is deployed into production as quickly as possible. The idea of a "release" is gone. Rather, the release is comparable to airplanes at an airport; once an airplane is ready to take off, it goes into the sky (or, in this case, the software is launched into production).

So, having developers focus on a sliver of software to work on allows the main application to continue working without interruption as development takes place. Then, since operations will take working, value-adding software and move it to the marketplace on an ASAP basis (preferably instantly), the software does not have a chance to get stale sitting idly waiting for the next release date.

What is the difference between Scrum and DevOps?

Now for the final answer.

Scrum is an iterative software development methodology, under the umbrella of Agile. DevOps is the support that the Scrum Team needs to get well-contained software changes into the marketplace as soon as possible. The main idea is that once the software is working, it should be shipped out to the development environment quickly so that:

- *Value is realized as quickly as possible;*
- *Competition does not beat you to the marketplace; and*
- *The business can start examining how real users use the software so that they can start creating new ideas and designs to advance their position in the marketplace.*

Summary

I hope you have enjoyed reading this article and have learned a bit reading it. We covered:

- What Agile is
- What Scrum is
- What DevOps is
- What the Difference is between DevOps and Scrum.

Should you like to comment, I can be reached at BVanderjack@IIT.EDU. Have a wonderful rest of your day.

References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). Manifesto for agile software development. Retrieved from http://agilemanifesto.org

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2017). *The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations*. Portland, OR: IT Revolution Press.

Schwaber, K. (2004). *Agile project management with scrum*. Redmond, WA: Microsoft Press.

About the Author

Brian Vanderjack

Illinois, USA

Brian Vanderjack, PMP, MBA, PMI-ACP, CSM is a graduate level adjunct faculty member at Illinois Institute of Technology (IIT), a top 100 University in the USA. For IIT he is a professor of Information Technology Team Leadership (mainly using Scrum and Project Management). In 2019, he was awarded Adjunct Faculty of the year from his department. He was a contributing subject matter expert for the creation of AT&T's class on How to Pass PMI's Agile Certified Practitioner certification exam; this class has a 100% pass rate. He has also taught many classes on how-to-pass-the-PMI-PMP-certification-exam.

Brian's professional certifications are: The Project Management Institute's (PMI) Project Management Professional (PMP), MBA (earned from DePaul University "with distinction"), Certified Scrum Master (CSM), and PMI's-Agile Certified Practitioner (PMI-ACP). Brian's title at the Fortune 10 company which currently employs him during the days as his full-time-day-career is "Senior Scrum Master." His track record of success there includes 8 years as a successful Sr. Scrum Master and Agile Coach. Also, he served as a Project Manager for over 10 years at this Fortune 10 company.

As an award-winning nationally respected speaker, he has spoken at many venues including Microsoft, IBM, AT&T, Abbot Labs, Freddie Mac, IIBA Chapters, and many PMI chapters across the USA on team leadership, communication, emotional intelligence, Scrum, and Project Management. He has also spoken at the University of Chicago's Booth Graduate School of Business and Northwestern University's Kellogg Graduate School of Business.

His ability to share information has resulted in many articles published in the *PM World Journal* (distribution approx. 15,000), and a book on Agile/Scrum called "The Agile Edge" which was published by Business Expert Press. He is honored to have received over 1,000 Linkedin "endorsements", and dozens of LinkedIn "recommendations" for his contributions in leadership and other areas. At his former university, he was honored to have earned "faculty of the year" for teaching Project Management.

Also, a special thank you to Jeremy Hajek for providing feedback on this article. Brian can be reached at BVanderjac@gmail.com.