

Use Scrum + Continuous Delivery to build the right thing

PETER GFADER

Introduction

How often do you release your product to your end users? How often do your end users see and use your product? Do you release in sync with your Sprint length, after the Sprint Review? Is the Sprint Review meeting the only valid release point? Do you plan your Releases in Release Planning meetings? Note: [Release Planning is gone from the Scrum Guide 2011](#).

"Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback."

From the Scrum Guide October 2011

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

From the Agile Manifesto: 1st principle behind the Agile Manifesto

As a framework for developing and sustaining complex products, Scrum combines the concepts of empiricism and the principles of the Agile Manifesto to engineer opportunities for feedback into the delivery life cycle. This is illustrated in the Scrum Guide by the following statement:

"Development Teams deliver an Increment of product functionality every Sprint. This Increment is usable, so a Product Owner may choose to immediately release it."

From the Scrum Guide, October 2011

This statement explicitly defines the cadence for delivery to be the period for the Sprint. It is also explicit about the state of delivery, ensuring its completeness so that feedback may be attained immediately.

This policy limits the ability to attain feedback to the Sprint Review. This paper removes this restriction, and describes a set of policies that enable a decoupling of the delivery of change and associated feedback from the Sprint container.



Overview

One of the key concepts of Agile is the importance of feedback loops (see Figure 1). In recent times this has evolved into the idea of Continuous Delivery. Continuous Delivery is the aim of keeping the system “Production Ready” during development to enable the release of a product to the end user on demand.

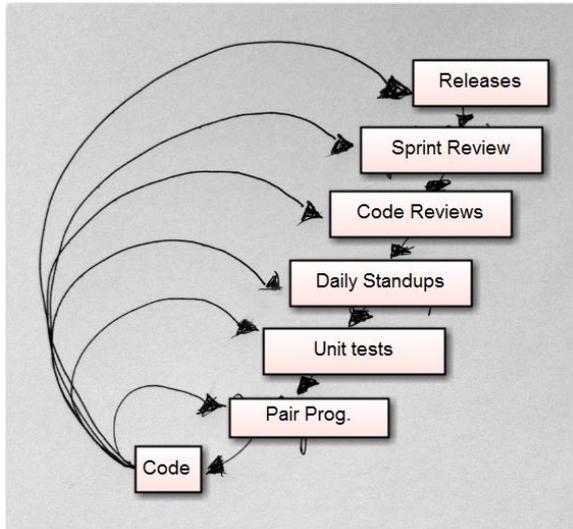


Figure 1: Feedback loops

We live in a fast paced time where companies who dominate their selected market are ones that delight their customers. Companies that produce features quickly, perform market testing with real users and use the gained feedback to drive future business decisions are proof that there is no substitute for a frequent release cadence.

If a company does not respond quickly to changes and delight their customers, a competitor could come along and grab their customers.

Validated learning facilitates further innovation, and those that learn fastest, capture the market. In order to withstand this competitive pressure in the market, a company needs to adjust and react quickly to the customer needs and wants (desires).

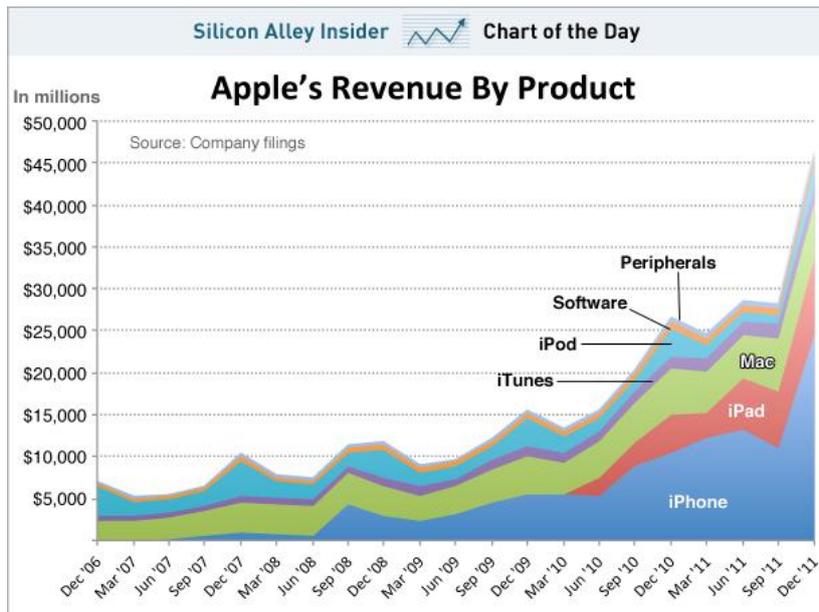


Figure 2: Apple's Revenue by Product (Dec 2011)

Apple is a good sample for an innovative company that makes ~ 75% of their revenue with products that didn't exist 5 years ago.

Problems with releasing products

"Development Teams deliver an Increment of product functionality every Sprint. This Increment is usable, so a Product Owner may choose to immediately release it."

From the Scrum Guide October 2011

Releasing a product to the market has been proven to be quite hard. The complexity of the product, external dependencies and complicated manual steps in the deployment are factors that contribute to the pain of releasing a product.

Additionally, the lack of deployment documentation, the low frequency of deployments and a deployment late in the development cycle cause more troubles and difficulties.

Releasing very often on the other hand, allows the Development Team to refine the release process and automate the deployment as much as possible.

By releasing a product often, the Scrum Team gets feedback quicker from the market. This frequent user feedback allows the Scrum Team to get closer to their users and shareholders, collaborate with them, respond quicker to changes and finally deliver a better product.

Definition of terms

“Released”: A business term that defines functionality being available to an end user.

“Shipping” is a synonym for “Released” and not used in this paper nor the Scrum Guide.

“Deployed”: A technical concern that applies in the domain of the team and means the product is introduced in a chosen environment. Different environments have a different user community and might be: Production, Testing and Integration.

“Deployed” doesn’t necessarily mean “Released”.

“Production Ready”: A product Increment that is “Done” and potentially releasable to the end user.

“Ready for Release” is a synonym to Production Ready.

“Continuous Integration”: A technical practice where every product change should trigger an event where you automatically rebuild, integrate and test your whole product

“Continuous Deployment”: Additionally to “Continuous Integration, every product change gets automatically deployed to Production.

“Continuous Delivery”: Keeping a system “Production Ready” for release at all times during development. This does not involve stopping and making a special effort to create a releasable product.

What are the differences between Continuous Delivery and Continuous Deployment?

Continuous Deployment is deploying every change into production. But that change might not be visible yet. Once the change is **released** it is available to the end user.

Continuous Delivery is about keeping your application in a state where it is always able to release.

Starting Conditions

The Development Team will require a high degree of automation to be in place both on the programming and the testing side to ensure that existing functionality continues to function.

Additionally

- Continual focus on releasable product
- Reduction of deployment transaction cost to enable smaller economic batch size

Solution

Continuous Delivery is the aim of keeping the system “Production Ready” during development and to release the product to the customer on demand.

What would change if you add “Deployed to Production” to your Definition of “Done”?

There needs to be a high level of trust between the Product Owner and the Development team in order to let the Development Team deploy every Product Backlog Item. After the deployment to Production, the Product Owner and the Development Team might decide to immediately release the latest change.

The Scrum Team adds to the Definition of “Done”:

- “Deployed to Production”
- “Ready to Release”

The Scrum Team should also consider adding “Released” to their Definition of “Done”.

Scrum Elements Being affected

The heart of Scrum is a Sprint, a time-box of one month or less during which a “Done”, useable, and potentially releasable product Increment is created. Sprints have consistent durations throughout a development effort.

From the Scrum Guide October 2011

The Sprint container policy, limits planning to a foreseeable time period, providing value by ensuring that only what is well known is considered and effort is not spent attempting to plan for, or deliver work that is not well understood.

This paper intends to explicitly decouple the constructs managing change delivery and planning. Decoupling these provides the opportunity for more rapid feedback and enables teams to maximise the value of the items delivered in an increment.

To decouple these events it is suggested that teams focus on separating the concepts of deployment from release. To this end, we define the term release to describe the action of making a product, or a part thereof available to users. We define the term deployment to mean the delivery of the technical requirements for a release to a target environment.

During the Sprint

During the Sprint the Development Team still creates a potentially releasable product Increment, although the single Product Backlog Items are in production.

Sprint Review

The inspection of the product Increment in the Sprint Review meeting should go smoother and give more time for adaptation of the Product Backlog. Additionally the Scrum Team gets more time in the Sprint Review meeting to measure the overall progress and plan major releases.

The “Release” of the product Increment is not related to the Sprint Review meeting.

In the Sprint Planning meeting the Scrum Team needs to consider the Definition of “Done” which includes “Deployed to Production”. The Development Team brings one slice of functionality from the idea till production as fast as possible. This “flow of delivery” should be observed by the Development Team and improved over time. By optimizing this flow the Development Team will realize a good size of Product Backlog Items.

The Scrum Master deals with any impediments identified in the Build Pipeline by the Development Team that introduce risk of delayed deliveries.

Clarification

Continuous Delivery is not shorter Sprints

The heart of Scrum is a Sprint, a time-box of one month or less during which a “Done”, useable, and potentially releasable product Increment is created.

From the Scrum Guide October 2011

The cadence of the Sprint is not aligned to the Release of a Product Backlog Item. Continuous Delivery is helping to move away from the activity of preparing and making software Production Ready. Instead the Development Team makes sure that the software is always “Production Ready”.

Being “Production Ready” is not “Releasing to the user”

The goal of the Development Team is to give the Product Owner the ability to release new Product Backlog Items whenever the Product Owner decides to. This means that the Product Owner might release every Product Backlog Item immediately, or he delays it until he gets feedback from stakeholders or he aligns the release to external events (Road shows, Christmas).

Releasing is possible with Product Backlog Items that are work in progress

Because the Product Owner can release the latest Product Backlog Item all the time throughout the Sprint, there could be certain Product Backlog Items that are still work in progress at the moment of the release.

Releasing Product Backlog Items that are not “Done” might be risky for the stability of the product and the Scrum Team. With certain practices it is possible to have a releasable current Product throughout the development without the risk of releasing a not “Done” work.

Releasing is possible, with features not available to the end user

Although the Product Backlog Item is deployed in Production, it doesn't necessarily mean it is available to the end user. The availability could be dependent from a date, an external decision maker, a user configuration or the need for a usage license.

Rollback is easy: Re-Release previous version

With enough automation set up, it is fairly easy to go back to a previous version. The Development Team just releases a previous version.

“Our situation is different” mindset: The Deployment in the Enterprise

Every project is different and Continuous Delivery is not easy.

There are companies which didn't have “Continuous Integration” in place, and nowadays they can't live without it. If your deployment is too complex, you might need to make it simpler. The key to get Continuous Delivery is Automation and collaboration between everyone involved in the delivery process.

“Continuous Integration” is a practice where you integrate the whole product on certain events, typically on a “Change”.

No Batch Deploy of Product Backlog Items

The Development Team should move away from accumulating Product Backlog Items, and deploying multiple Backlog Items in 1 step. There is no dedicated activity of the Development Team that involves Deployment.

What the Development Team should strive for is bringing 1 slice of functionality from the idea till production as fast as possible.

Case studies

IMVU

IMVU coined the term "Continuous Deployment" and use Scrum.

<http://engineering.imvu.com/2010/04/21/imvu-agile-engineering-process/>

Facebook

Facebook releases daily.

<http://techcrunch.com/2011/05/30/facebook-source-code/>

Netflix

Netflix is testing ideas on the market with fast deliveries.

http://www.uie.com/articles/fast_iterations/

Flickr

Flickr uses a code repository with no branches; everything is checked into head, and head is pushed to production several times a day. Flickr uses Feature Flags to enable features on a per environment basis.

<http://code.flickr.com/blog/2009/12/02/flipping-out/>

AuctionsPlus: Automated Deployment for nightly performance tests

An Australian service provider for electronic online auctions established in the mid-1980s needed to re-develop their existing online Auction Platform in a newer technology, to improve the throughput and workflow of the Auction system and to enhance the user experience. The existing technology had a number of limitations, was hard to maintain and presented an old and tired interface to the end-users.

In the early stage of the project the Development Team has set up a system that allowed them to deploy the latest executable version of the auction software to the production environment with a single button click.

With this ease of deployment to production the team was able to iterate over user interface changes very quickly and delight the customer with frequent new user experience features. Additionally the team was able to recognize some hardware problems in the production environment which were discovered on the 1st deployment to those servers. With a late in the project deployment that hardware problem would have caused a lot of stress and a huge delay on the release deadline.

Additionally there was Continuous Deployment to a stress test server in place that pushed the latest executable version to a stress test server after each developer check-in.

The latter setup allowed the team to run stress tests every night, and make sure that their development efforts during the day didn't hurt the performance of the auction system, which was an important value for the business.

Good practices

Embrace “DevOps”

Since there is no dedicated activity during the Sprint that involves releasing the product, we don't have dedicated engineers that are releasing and maintaining the product. The Development Team is responsible for releasing the product successfully. Maybe the Development Team needs an Operations engineer to achieve this.

“DevOps” is an emerging set of principles, methods and practices for communication, collaboration and integration between software development (application/software engineering) and IT operations (systems administration/infrastructure) professionals

From Wikipedia

Build a “Deployment pipeline” with Quality gates

Every change from the Development Team needs to be integrated and thoroughly tested before it is Production Ready. A Deployment pipeline helps you to establish an automated way from a Check-In till Deployment and to shorten the feedback cycles. Quality Gates build up the Deployment Pipeline, which means that quicker verifications are run before slower verifications. After a passing Quality Gate, the Build Pipeline progresses to the next Gate.

Potential Quality Gates to consider for the Development Team:

1. Check-in to version control
2. Build
3. Run automated tests
4. Run automated acceptance tests
These tests capture the acceptance criteria of the Product Backlog Items we develop, and prove that the code meets those acceptance criteria
5. Run nonfunctional tests like performance or load tests
6. Manual User acceptance tests

Set up Continuous Deployment to a production like environment

As an exercise for Continuous Delivery the Development Team should set up automated deployment to a production like environment by using the same process and tools as for Continuous Delivery.

Automation is key

Automation of the Deployment pipeline is important to get the benefits of Continuous Delivery.

Embrace Evolutionary Design

In order to transition one Product Backlog Item quickly from the Idea until Production, the Development Team needs to consider an Evolutionary Design Approach. In this approach the Product grows over time with a minimized phase of upfront planning.

The setup of Continuous Delivery is not easy and needs the collaboration of developers and the operations team. Everybody in the organization needs to focus on delivering business value and not focus on their traditional business role (developer, tester, QA). This can cause problems in an organization where change management is not treated as a first class citizen.

The business might worry about frequent releases, because with traditional methods change is a risky thing and might lead to problems. But with Continuous Delivery we can optimize the release of products by doing it more often which gives us better reliability, stability and the trust of the customer that we can deliver software successfully.

Intended Outcome

A successful setup of Continuous Delivery leads to frequent deliveries of the product to the end user and an increased collaboration between the Scrum Team and the stakeholders.

With frequent deliveries in place the Scrum Team is able to test new ideas in the market. Getting feedback from the market is very valuable because it reflects the needs of the end user and finally we want to delight them with frequent new features and support.

With the feedback from the market the Scrum Team might change priorities and order the Product Backlog correspondingly and even cancel the current Sprint just in order to follow the needs of the market.

Considerations

Continuous Delivery being a technical practice, gives benefits that go way beyond a single project or the technical teams. The benefit of delivering business value quicker from an idea to the actual usage of the feature could have a real strategic impact on the competitive advantage of an organization.

If Continuous Delivery extends beyond the internal organization and is released to market, real metrics and insight can be gained from real usage, meaning that more useful features are developed further and less useful features are not: Build the right thing.

Authors

Submitted by: Name: Peter Gfader Email: peter.gfader@zuehlke.com Date: 12.07.2012	Sponsored by: Name: Adam Cogan Email: adamcogan@ssw.com.au Name: Steve Godbold Email: steve@stevegodbold.com Name: Daniel Tobler Email: daniel.tobler@zuehlke.com Name: Martin Hinshelwood Email: martin@hinshelwood.com Date: 12.07.2012
---	--

About The Author

Peter loves quality software and tries his best to improve the profession of software development. For this reason he joined scrum.org, and this is what keeps him getting out of bed every day... and the smell of coffee.

One day, Peter woke up and realized that software development is not only about code, but also about people: From his team mates till the end user. Some people you just give donuts and some you need to give a little bit more. Peter is on a journey to make everyone happy.

If he is not sitting on a mountain bike or playing the trumpet, you might find him at a local user group to hang out with other geeks!

About Scrum.org

Scrum.org is the home of Scrum, and is leading the evolution and maturity of Scrum to improve the profession of software development.

Scrum.org strives to provide all of the tools and resources needed by Scrum practitioners to deliver value using Scrum. We host the Scrum Guide in 30 languages, provide Scrum assessments to allow people to evaluate themselves and improve, host community forums and webcasts to foster discussion and knowledge transfer, and define industry-leading Scrum training for practitioners at all levels.

Scrum.org was founded in 2009 by [Ken Schwaber](#), one of the creators of Scrum, along with [Alex Armstrong](#), out of extreme dissatisfaction with the state of the art.

About Zühlke

Zühlke Engineering works together with companies to create innovative products and bespoke software solutions. Our range of services includes consulting, development and integration. We have a wealth of experience, with more than 7000 successful client projects completed. Zühlke Engineering is part of the Zühlke Group. Founded in 1968, the Zühlke Group now has local teams in Austria, Germany, Switzerland and the United Kingdom, and in 2011 generated Euro 68 million in revenue, employing more than 500 staff.

References

Jez Humble, David Farley (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series (Fowler))

<http://continuousdelivery.com/>

Agile Manifesto

<http://agilemanifesto.org/principles.html>

Scrum Guide

<http://www.scrum.org/scrumguides>

7 Reasons why Continuous Delivery needs to be a BUSINESS initiative

<http://www.allaboutagile.com/7-reasons-why-continuous-delivery-needs-to-be-a-business-initiative/>

The Full Monty: the Continuous Delivery Business Case at Scale

<http://pagilista.blogspot.com/2011/10/full-monty-continuous-delivery-business.html>

Continuous delivery, from Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Continuous_Delivery

Continuous Delivery with TFS, msbuild and msdeploy

<http://blog.gfader.com/2011/07/continuous-delivery-with-tfs-msbuild.html>

Is Design Dead?

<http://martinfowler.com/articles/designDead.html>

Continuous Delivery VS Continuous Deployment

<http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/>

Apple's Revenue By Product (Dec 2011)

<http://www.businessinsider.com/chart-of-the-day-apple-the-iphone-company-2012-1>