Faculty of Informatics
Masaryk University

IV064
Information Society

Modern Agile Software Engineering

Adrián Tóth (491322) November 17, 2019

Contents

1 Software Engineering

From time to time, software engineering become the most important part of software development. There are many different definitions of software engineering which are sometimes misleading and incorrect. The *IEEE*[1] defines the software engineering, which clarifies its true meaning in the software development.

Definition 1.1. Software engineering

> The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [1].

Up to the present time, software engineering has evolved and adapted by the time, introducing new methodologies for software development, which lead to an increase in the number of successful projects. Software development methodologies provide a framework for planning, executing, and managing the process of developing software systems [17]. The importance of software engineering has resulted in a research of software engineering itself in a form of empirical, contemplative, case and other studies, which delivered new approaches, attitudes and practices. The speed of information technology development is massively progressive and incredibly fast and it is also part of our daily life.

Definition 1.2. Delivery

> Release of a system or component to its customer or intended user [1].

Nowadays requirements for the software are unbearably high. Developing a software present days means, to deliver the product as soon as possible and simultaneously provide the best possible product quality, which is classified by product metrics. Quality plays a vital role for the software users [9]. This article deals with the software delivery time from the view of the modern software engineering.

Developing a quality software product is an essential need for the software industry [9]. To deliver a quality product under short time, it requires development agility, cross-functional teams and collaborative effort of self-organization. To deliver a quality product as fast as possible, it is needed to meet the above mentioned requirements and use a proper framework and methodology. The most common development methodologies is agile as well as predictive methodology. In addition to agile development methodology, we will discuss and compare the newly integrated methodologies and practices.

1.1 Time to Market

Time to the market is the key to success in the field of information technology. Businesses must be prepared and shaped to adapt and evolve the modern breakthrough technologies. It is not

[1] Institute of Electrical and Electronics Engineers

so easy to be up to date, especially not in the information technology business. From time to time, as every manual tasks were transformed into a fully automated hands–off processes, the software engineered was also influenced by this automation impact. Automation is limitless because of the fact that there can be any task transfered from manual to automated. If the automated process is configured properly, it may save a huge amount of time.

The software development must be also faster, the automation of the software delivery pipeline is not the only one required for a quick software delivery. Time needed for software product creation is the most influencing part of the delivery time. Based on the development progress the time may be also prolonged or shortened.

2 Agile Development

Delivering a product in the shortest possible time requires good software development methodology, which is possible with agile development methodology. Agile methodology provides all the necessary approaches, practices and disciplines to the software development, which allow fast product delivery and high quality of the product. The main and principal approaches to software development are: flexible and adaptable, focus on people (customers, users, developers), regularly updated requirements. In a comparison to predictive methodology, agile provides less (minimal) effort to upfront planning and the product is delivered after small iterations in a form of product increments (releases). Agile methods are based on small development cycles, continuous integration of software versions, adaptive planning, team collaboration, customer involvement, and feedback [20].

Definition 2.1. Scrum

> A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [13].

SCRUM is the most well known and common agile software development framework. It is simple to understand but difficult to master. The framework describes and defines the whole development structure – events, roles and artifacts. The scrum framework poster in Figure 4 provides a graphical view of how scrum is implemented at a team level within an organization [12].

It is difficult to state precisely the advantages and disadvantages against agile alternatives. Both agile and predictive have pros and cons which can differ by the project requirements. Predictive approach is greatly used for a project where we exactly know the product requirements and specifications. Agile approach is useful while it is needed to deliver the product (or maybe its prototype) in the quickest time while it might be later enhanced by additional patches via updates. The manner of time and product specifications (requirements definition) are crucial to state the shortcomings of each methodologies.

2.1 DevOps

A new development methodology has been created to reduce the time costs for software product delivery. You likely have heard of the word DevOps, so let us clarify its true meaning. The modern software development involves an immense effort to the product deployment because of integration and connection conflicts between each product parts. DevOps efficiently integrates development, delivery, and operations, thus facilitating a lean, fluid connection of these traditionally separated silos [5]. It came to address this conflict and bridge the gap between developers and operations staff [19].

Definition 2.2. DevOps

> DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of changes. [4].

The acronym consists from two parts Dev and Ops. DevOps is an abbreviation of development and IT operations, it is not limited to those two teams; thus, there is no need for "extending" acronyms like DevSecOps or DevNetOps [4].

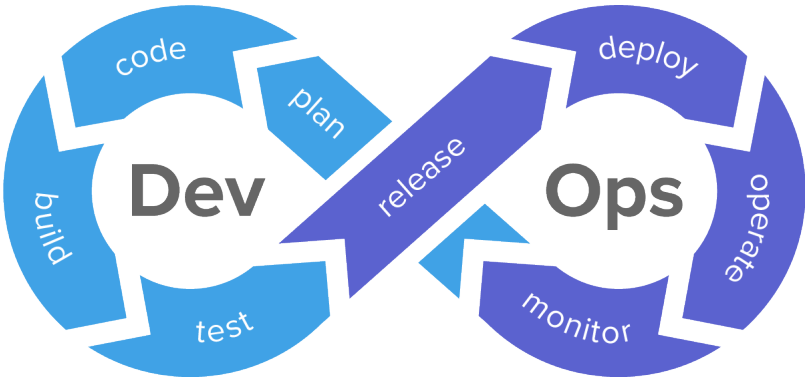

Figure 1: DevOps environment [10].

For the purpose of development acceleration, the DevOps is fundamental to agile development. The above stated facts point to that it provides an automated way how to deliver product as fast as possible, which may be also included in other types of developments. Under those circumstances, as the actions are done in a non manual way after each other, we denote this process as continuous. DevOps uses many continuous approaches, the most important one from the all of them is continuous delivery which will be described later in the following Section 3.

All things considered, DevOps has a very beneficial influence for the development itself. It results in time costs reduction for the product delivery – from source code change to a deployable usable release. Obviously, DevOps must be performed by somebody so as a disadvantage, we may consider that the development requires a special team for this purpose.

3 Continuous Delivery

Continuous delivery advocates claim that it lets organizations rapidly, efficiently, and reliably bring service improvements to market and eventually stay a step ahead of the competition [8].

Definition 3.1. Continuous Delivery

> Continuous Delivery (CD) is a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time [2].

Continuous delivery is related to the DevOps automation. Using a continuous delivery approach is very powerful and useful to the development. Implementing a continuous delivery into the development process requires a continuous delivery pipeline configuration and setup. The pipeline must implement altogether the development actions such as build, test, release, deploy in an automated process. Figure 3 as an illustrating image visualize the order of main actions from high perspective required for the implementation.

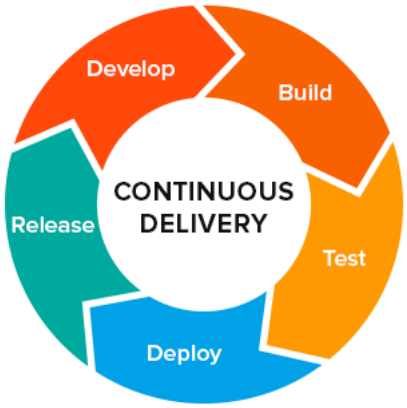

Figure 2: Continuous delivery workflow [3].

In conclusion, continuous delivery is a software development discipline where you build software in such a way that the software can be released to production at any time [7]. Continuous delivery can bring huge benefits, but implementing continuous delivery is challenging [2]. Development needs extra time for the continuous delivery pipeline implementation and later, maintenance and further support for additional changes necessary to adapt to the pipeline. Excluding the implementation and maintenance, the result of using a continuous delivery is very beneficial and advantageous. The main advantage of that it is an instant problem detection for such problems as dependency errors between each part of product, which result in a broken pipeline where the developers can respond quickly to that problems.

Continuous delivery involves continuous integration, continuous testing, continuous deployment, continuous monitoring and continuous pipeline. There are the key parts of continuous

delivery which need special attention to. We are going to discuss the basic principles and fundamentals as well as pros and cons in the following sections below about the most important parts – continuous integration and the pipeline itself.

3.1 Continuous Integration

As part of agile transformation in past few years we have seen IT organizations adopting continuous integration principles in their software delivery lifecycle, which has improved the efficiency of development teams [18]. The modern software development is using enterprise architectural design patterns which divides the system to subsystem based on its functional purpose and operation. Subsystem set (each function independent parts) of the whole system must be integrated into one system unit which is not as easy as it looks. To describe what continuous integration is, we must declare some terms for better understanding.

Definition 3.2. Build

> A Build may refer to a set of activities performed to generate, test, inspect, and deploy software [11].

Definition 3.3. Integration Build

> An Integration Build is the act of combining software components (programs and files) into a software system [11].

In brief, the integration build is a transformation process that integrate these software components together into a one unified entity. It does depend where it is done. If a developer wants to run the software on his local machine, he will probably perform a private integration build. In contrast, the pure integration build is mostly performed on some remote system after which, the system entity may pass to the following stages – system testing, deployment or release.

Definition 3.4. Continuous Integration

> Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day [15].

Martin Fowler pointed out the problem of subsystem integration in his article: "I was told that this project had been in development for a couple of years and was currently integrating, and had been integrating for several months." [6]. Some research articles distinguish integrations types into two groups – big bang integration and daily integration. The most common one is the daily integration because its alternative – big bang integration, does not reduce the risks introduced during the system unification. The daily integration, as it is stated in the name, provides periodical integrations, which lead to less conflicts and risks reduction between system parts.

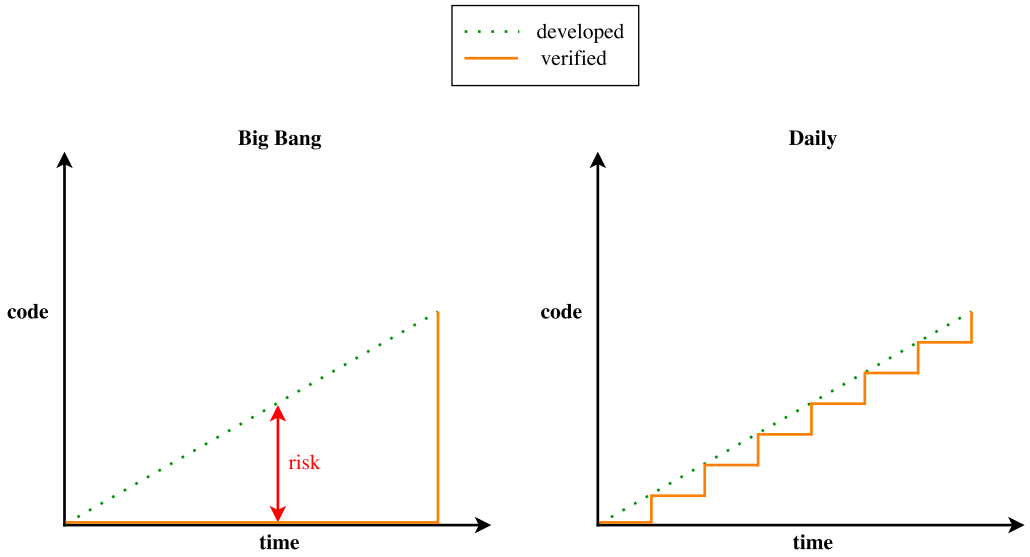

Figure 3: Comparison of integration builds [16].

> *"Not integrating continuously is expensive. If you don't follow a continuous approach, you'll have longer periods between integrations. This makes it exponentially more difficult to find and fix problems. Such integration problems can easily knock a project off-schedule, or cause it to fail altogether."*
>
> – ThoughtWorks® [15]

A continuous integration system is often considered one of the key elements involved in supporting an agile software development and testing environment [14]. There are few practices [6] that is good to follow while using continuous integration, which are listed below.

- Use and maintain a single source code repository.
- Automate the build.
- Make the build self-testing.
- Everyone commits to the mainline every day.
- Every commit should build the mainline on an integration machine.
- Fix broken builds immediately.
- Keep the build fast.
- Test in a clone of the production environment.
- Make it easy for anyone to get the latest executable.
- Everyone can see what's happening.
- Automate deployment.

The continuous integration pros and cons are most likely same as what was mentioned in paragraph 3 of Section 3. The trouble with deferred integration is that it's very hard to predict how long it will take to do, and worse it's very hard to see how far you are through the process [6]. While using a continuous approach to the integrations we completely erase the problem of later integration what is considered as a main advantage for using continuous integration.

3.2 Deployment Pipeline

4 References

[1] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE Std 610.* 1991. doi:10.1109/IEEESTD.1991.106963. [Online; Accessed: 2019-11-12].
Retrieved from: https://ieeexplore.ieee.org/document/182763

[2] Chen, L.: *Continuous Delivery at Scale: Challenges and Opportunities.* In *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE).* 2018. [Online; Accessed: 2019-11-17].
Retrieved from: https://ieeexplore.ieee.org/document/8452107

[3] CloudBees, I.: *Continuous Delivery.* 2019. [Online; Accessed: 2019-11-17].
Retrieved from: https://www.cloudbees.com/use-case/continuous-delivery

[4] Dyck, A.; Penners, R.; Lichter, H.: *Towards Definitions for Release Engineering and DevOps.* In *2015 IEEE/ACM 3rd International Workshop on Release Engineering.* 2015. doi:10.1109/RELENG.2015.10. [Online; Accessed: 2019-11-16].
Retrieved from: https://ieeexplore.ieee.org/document/7169442

[5] Ebert, C.; Gallardo, G.; Hernantes, J.; et al.: *DevOps. IEEE Software.* 2016. doi:10.1109/MS.2016.68. [Online; Accessed: 2019-11-16].
Retrieved from: https://ieeexplore.ieee.org/abstract/document/7458761

[6] Fowler, M.: *Continuous Integration.* 2006. [Online; Accessed: 2019-11-17].
Retrieved from: https://martinfowler.com/articles/continuousIntegration.html

[7] Fowler, M.: *Continuous Delivery.* 2013. [Online; Accessed: 2019-11-17].
Retrieved from: https://martinfowler.com/bliki/ContinuousDelivery.html

[8] Humble, J.; Farley, D.: *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Pearson Education. 2010. ISBN 978-0-321-60191-9.

[9] Jain, P.; Sharma, A.; Ahuja, L.: *The Impact of Agile Software Development Process on the Quality of Software Product.* In *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).* 2018. doi:10.1109/ICRITO.2018.8748529. [Online; Accessed: 2019-11-12].
Retrieved from: https://ieeexplore.ieee.org/document/8748529

[10] Ltd., N.: *DevOps and Application Security.* 2019. [Online; Accessed: 2019-11-17].
Retrieved from: https://www.netsparker.com/devops-security-tools/

[11] Paul M. Duvall; Steve Matyas; Andrew Glover: *Continuous Integration - Improving Software Quality and Reducing Risk.* Pearson Education, Inc.. 2007. ISBN 0-321-33638-0.

[12] Scrum.org: *The Scrum Framework Poster.* 2019. [Online; Accessed: 2019-11-12].
Retrieved from: https://www.scrum.org/resources/scrum-framework-poster

[13] Scrum.org: *The Scrum Guide*. 2019. [Online; Accessed: 2019-11-12].
Retrieved from: https://www.scrum.org/resources/scrum-guide

[14] Stolberg, S.: *Enabling Agile Testing through Continuous Integration*. In *2009 Agile Conference*. 2009. doi:10.1109/AGILE.2009.16. [Online; Accessed: 2019-11-17].
Retrieved from: https://ieeexplore.ieee.org/document/5261055

[15] ThoughtWorks, I.: *Continuous Integration*. 2019. [Online; Accessed: 2019-11-17].
Retrieved from: https://www.thoughtworks.com/continuous-integration

[16] Tóth, A.: *Continuous Integration and Automated Code Review in Open Source Projects*. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. 2018.
Retrieved from: https://www.fit.vut.cz/study/thesis/21007/

[17] Vijayasarathy, L. R.; Butler, C. W.: *Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? IEEE Software*. 2016. doi:10.1109/MS.2015.26. [Online; Accessed: 2019-11-12].
Retrieved from: https://ieeexplore.ieee.org/document/7006383

[18] Virmani, M.: *Understanding DevOps bridging the gap from continuous integration to continuous delivery*. In *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*. 2015. doi:10.1109/INTECH.2015.7173368. [Online; Accessed: 2019-11-17].
Retrieved from: https://ieeexplore.ieee.org/document/7173368

[19] Wahaballa, A.; Wahballa, O.; Abdellatief, M.; et al.: *Toward unified DevOps model*. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2015. doi:10.1109/ICSESS.2015.7339039. [Online; Accessed: 2019-11-16].
Retrieved from: https://ieeexplore.ieee.org/document/7339039

[20] Younas, M.; Jawawi, D. N.; Ghani, I.; et al.: *Agile development in the cloud computing environment: A systematic review*. *Information and Software Technology*. 2018: pp. 142 – 158. doi:https://doi.org/10.1016/j.infsof.2018.06.014. [Online; Accessed: 2019-11-12].
Retrieved from:
https://www.sciencedirect.com/science/article/pii/S0950584918301319

5 Appendix

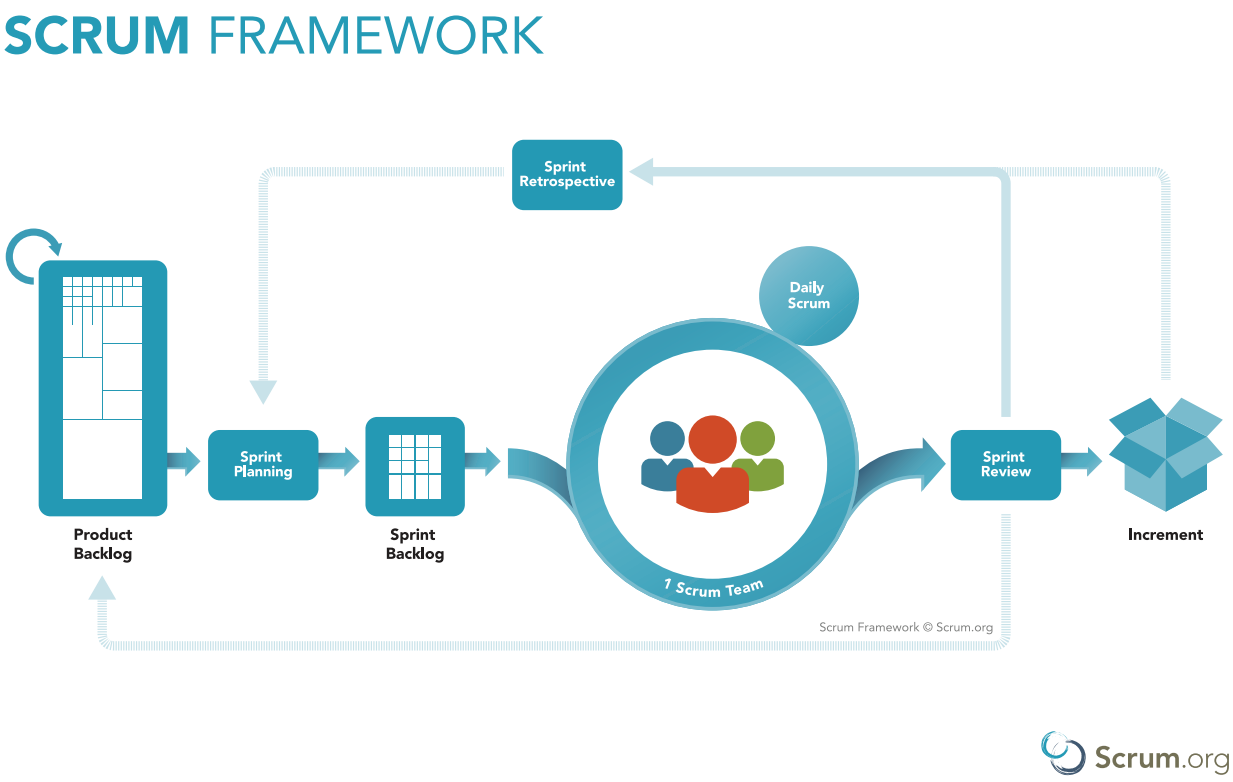

Figure 4: The Scrum Framework Poster [12].