IMPORTANCE AND EFFECTS OF CONTINOUS DELIVERY ON AGILE SOFTWARE DEVELOPMENT LIFECYCLE

DORUK KUTLUAY

MARCH, 2018

IMPORTANCE AND EFFECTS OF CONTINUOUS DELIVERY ON AGILE SOFTWARE DEVELOPMENT LIFECYCLE

BY

DORUK KUTLUAY

DISSERTATION SUBMITTED IN PARTIAL FULLFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ARTS

IN

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

YEDITEPE UNIVERSITY

MARCH, 2018

Approval of the Institute of Social Sciences

[illegible]

Prof. Dr. M. Fazıl GÜLER
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master.

[illegible]

Prof. Dr. Avadis Simon HACINLIYAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis degree of Master of Arts.

[illegible]

Assist. Prof. Dr. Gökhan ŞAHİN
Supervisor

Examining Committee Members

Assist. Prof. Dr. Gökhan ŞAHİN - Yeditepe University [illegible]

Assoc. Prof. Dr. Aşkın DEMİRAĞ - Yeditepe University [illegible]

Assist. Prof. Dr. Hacı Ahmet YILDIRIM - Sakarya University [illegible]

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

08 /03 /2018

DORUK KUTLUAY

Signature: [illegible]

ABSTRACT

Technology development and digitalization affect the real world differently. Automated systems integrated every transaction in daily life. Continuous integration and continuous delivery are the heart of the automated systems. The aim of this thesis is to investigate and identify the present and future effects of continuous integration and continuous delivery methodologies on agile software development life-cycle which are improved by new generation technologies.

In the first part, the term of agile software development, continuous integration and continuous delivery methodologies are presented with their historical evolutions. These technologies highlight the world-wide competitive game changer effects, but differ from other software development methodologies. The evolution of the terms clarified with comparisons.

The thesis then identifies the automation term which is developed by continuous integration, and further outlines the structure of the need for continuous delivery. Detailed sections of the thesis provide an overview of applicable continuous delivery model which includes integration, progresses with well-defined properties. In addition, detailed sections reveal the digital enterprise solutions, benefits and importance of continuous delivery which answer the big argument of digitalization.

On the other hand, detailed sections argue possible threats for continuous delivery model with possible ways to prevent failure.

In conclusion, the thesis argues that continuous integration and continuous delivery methodologies and their technologies affect the enterprises in a positive way and increase the profit of software development projects. Detailed sections prove the importance of the continuous integration and continuous delivery methodologies on agile software development life-cycle.

Keywords: continuous delivery, continuous integration, project management, business solutions, agile methodologies, software development life-cycle.

ÖZET

Teknoloji geliştirme ve sayısallaştırma gerçek dünyayı farklı şekilde etkilemektedir. Otomatikleştirilmiş sistemler günlük hayattaki tüm işlemlere entegre olmuştur. Sürekli entegrasyon ve sürekli teslim otomatik sistemlerin kalbidir. Bu tezin amacı, yeni nesil teknolojilerle geliştirilen çevik yazılım geliştirme yaşam döngüsü üzerindeki sürekli entegrasyon ve sürekli teslim yöntemlerinin bugün ve gelecekteki etkilerini araştırmak ve tanımlamaktır.

Birinci bölümde çevik yazılım geliştirme, sürekli entegrasyon ve sürekli teslim metodolojileri tarihsel gelişmeleri ile birlikte sunulmaktadır. Bu teknolojiler, dünya çapında rekabetçi oyun değiştirici efektler olarak vurgulanır, ancak diğer yazılım geliştirme yöntemlerinden farklıdır. Koşulların evrimi karşılaştırmalarla açıklığa kavuşmuştur.

Daha sonraki bölümde sürekli entegrasyon ile geliştirilen otomasyon terimi tanımlanmıştır ve sürekli teslim ihtiyacının yapısı özetlenmiştir. Entegrasyonu içeren, uygulanabilir sürekli teslim modeline iyi tanımlanmış özelliklerle genel bir bakış sunulmuştur. Buna ek olarak, sayısallaştırma konusundaki büyük argümana cevap veren dijital kurumsal çözümleri, faydaları ve sürekli teslimin önemi ortaya koyulmaktadır.

Öte yandan, detaylandırılmış bölümler, başarısızlığı önlemenin olası yollarıyla sürekli teslim modeli için olası tehditleri tartışmaktadır.

Sonuç olarak tez, sürekli entegrasyon ve sürekli teslim metodolojileri ve teknolojilerinin işletmeleri olumlu yönde etkilediğini ve yazılım geliştirme projelerinin karını artırdığını iddia etmektedir. Detaylı bölümler, çevik yazılım geliştirme yaşam döngüsü üzerinde sürekli entegrasyon ve sürekli teslim metodolojilerinin önemini ispatlıyor.

Anahtar kelimeler: sürekli teslim, sürekli entegrasyon, proje yönetimi, iş çözümleri, çevik metodolojiler, yazılım geliştirme yaşam döngüsü

ACKNOWLEDGEMENT

I am using this opportunity to express my gratitude to everyone, especially to my family and who supported me throughout the courses of this MBA program. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the thesis work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the thesis.

I express many warm thanks to my advisor at Yeditepe University, Mr. Gökhan ŞAHİN for his special guidelines within the scope of every courses I joined to complete the MBA program.

I present dignity and thanks to Mr. Avadis Simon HACINLIYAN, Division President of the Yeditepe University and Mr. Aşkın DEMİRAĞ, for their encouraging guidelines and commentaries within the scope of the thesis.

I would also like to thank to Mr. Tuncay ŞENTÜRK, and my other colleagues in Snapbytes, for their continuous support and guidance during the research process.

Finally, I must express my very profound gratitude to my parents and to Ms. Burcu KIRIK KUTLUAY, Sr. Mobile iOS Developer, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you,
Doruk KUTLUAY.

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1. INTRODUCTION AND OVERVIEW

1.1. Introduction

Technology is a life-long process of change, improvement and competition in every sector, world-wide range. Organizations, companies, sectors have to be aware of technology to survive in today’s world. Customer demands on products or services that these organizations present became top priority in technological improvements. So that automatization and automation systems transformed into a critical component of these improvements while also being indispensable for business competition and survival.

When the subject is automation systems in business, profit seems to be the first reason that comes to mind. However, automation systems are not only related to profit; it is about more about quality, intelligence, standards and usability. Well-designed and implemented automation systems can enable lots of opportunities to the company and the people, both of workers and customers, such as fast but complex processes, safe and user friendly products, standardized production, etc.

In this thesis, the problem is how strong are automation systems and their effects on both of personal and business life. The study includes automation systems' details, implementation, effects and threats in terms of technical and business which highlights the numbers coming from the automation systems outputs and, or project results. This thesis presents world-wide analysis in terms of both local and global data which mean analysis contains different countries.

1.2. Objective and Motivation

This thesis study aims to prove that the concepts of “continuous delivery”, “continuous integration” and “continuous deployment” are globally improving and have positive effects on software and project management. It will categorize the advantages and disadvantages created in software and management perspectives and demonstrate the superiority of advantages, if these methods are preferred. Quality and convenience studies are continuously carried out in the areas of software development and management. Much time and effort are being spent to determine the best methods in these areas. With this thesis study, the quality and facility that these concepts will reveal will be explained, while explaining the methods and principles to providing better quality software and more sophisticated management cycle. In addition to these principles, the thesis aims to clarify possible risks, provide solutions, and ask the right questions to improve software and management methodologies. Thus, at the end of this study, the necessity of using agile software methods will be clearly demonstrated.

2. LITERATURE REVIEW

2.1. Early Stage of Software Development

According to Fowler[1] and Leffingwell[2], early stages software development can be expressed as "*code and fix, code-some-more, fix-some-more*". This very simple scheme can be thought of as the first generation in the history of software development methods. The basic concept of scheme, without much effort in preliminary planning and design write code and to fix bugs later if any are found. This shows that software development processes do not contain structured and disciplined software development methods in the first stages. It worked very well for such small-scale and relatively simple projects. On the other hand, as the size of the projects increases, developers notice that they spend more time writing code to fix the errors. This has led to a dramatic decline in the effectiveness and predictability of the software developer. The origins of the concept of early-structured software development methodology are engineering disciplines that focus on precise planning.

2.2. Waterfall Software Development

Capability Maturity Model framework of The Software Engineering Institute sets the philosophy of the waterfall methodology, and on the other hand, it is a frequently used standard for measuring process excellence. [3] The classic waterfall software development model and deliverables are represented in *Figure 1* and Table 1, respectively. It is characterized by gate system in physical engineering and has various stages number and names. As can be seen from *Figure 1* and Table 1, each

[1] Fowler, M. (2005, December 10). The new methodology.
[2] Leffingwell, D. (2007). Scaling software agility: best practices for large enterprises. NJ: Pearson Education.
[3] Paulk, M., Weber, C., Curtis, B., & Chrissis, M. (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. Indianapolis: Addison-Wesley.

level needs to be completed first. A lower phase may start, but it cannot move to the upper phase. Each phase cannot go back to the previous phase.

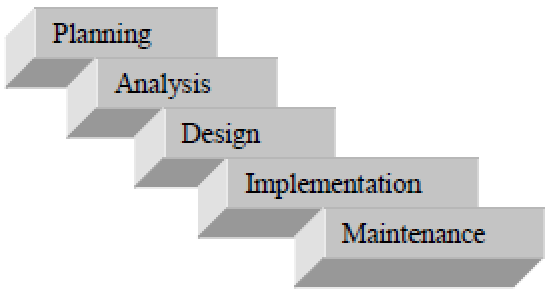


Figure 1. The “Waterfall” methodology lifecycle[4]

Table 1. *The “Waterfall” methodology deliverables*

Phases	Deliverables
Planning phase	Planning specifications
Analysis phase	Analysis specifications
Design phase	Design specifications
Implementation phase	Implementation specifications

The original version of the Waterfall model was described by Royce in 1970. Royce thought that the software development process had to be a recursive process contrary to what was believed until then. If the processes are consecutive, they argue that applications tend to be risk and unsuccessful. On the other hand, the Royce model generally emphasizes heavy feedback and iterations. However, it has been misinterpreted by many as a constant sequential process.[5] The original Royce Waterfall model is shown in *Figure 2*.

[4] Cho, J. J. (2010). An exploratory study on issues and challenges of agile software development with scrum. All Graduate theses and dissertations, p.8.
[5] Leffingwell, D. (2007). Scaling software agility: best practices for large enterprises. NJ: Pearson Education.

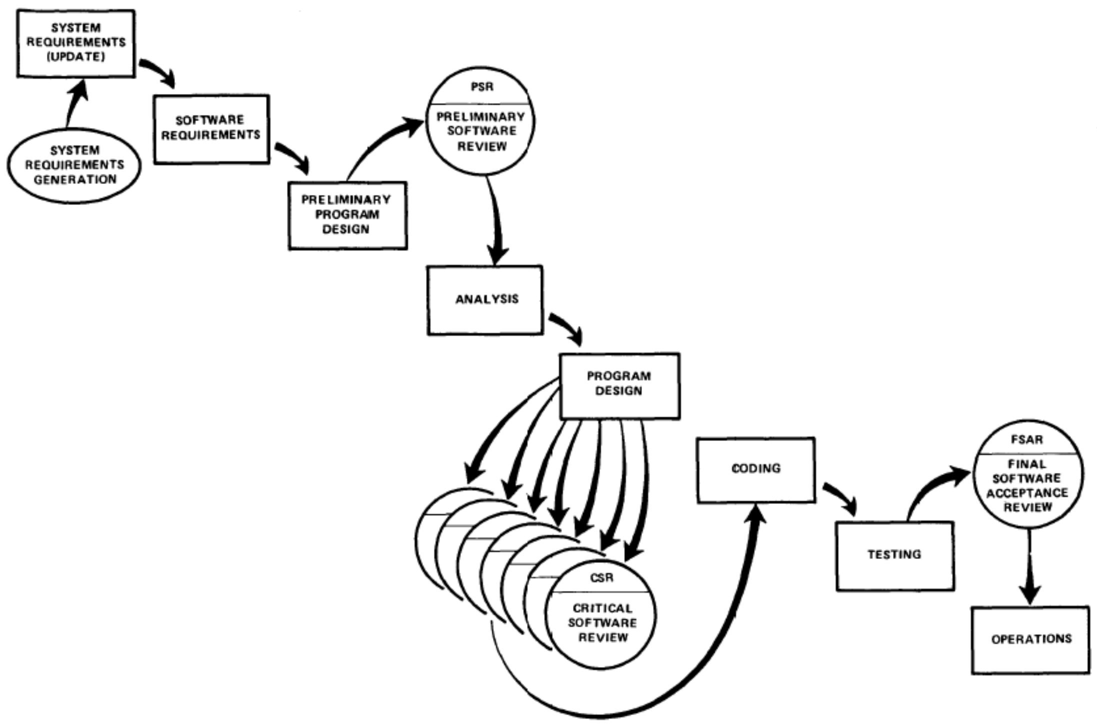

SYSTEM REQUIREMENTS (UPDATE)
SYSTEM REQUIREMENTS GENERATION
SOFTWARE REQUIREMENTS
PSR
PRELIMINARY SOFTWARE REVIEW
PRELIMINARY PROGRAM DESIGN
ANALYSIS
PROGRAM DESIGN
CODING
FSAR
FINAL SOFTWARE ACCEPTANCE REVIEW
TESTING
CSR
CRITICAL SOFTWARE REVIEW
OPERATIONS

Figure 2. Original Royce waterfall model[6]

Especially in the past years, developments using traditional waterfall methodology have been preferred by large-scale software projects and governments because they are simple, predictable, stable.[7,8] On the other hand, traditional software development models have some key and basic shortcomings. These problems can be summarized as slow adaptation to innovations, budget and timing problems. [9]

[6] Cho, J. J. (2010). An exploratory study on issues and challenges of agile software development with scrum. All Graduate theses and dissertations, p.10.
[7] Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. Computer, 36(6), 57-66.
[8] Fruhling, A., & De Vreede, G. J. (2006). Field experiences with extreme programming: Developing an emergency response system. Journal of Management Information Systems, 22(4), 39-68.
[9] Cho, J. J. (2010). An exploratory study on issues and challenges of agile software development with scrum. All Graduate theses and dissertations, 599.

2.3. Agile Software Development

Agile software development methods (ASDM) have been developed to overcome the shortcomings of traditional software development methods. Unlike waterfall methodologies, agile methodologies see the software development process as an "empirical" process. Waterfall and agile methodologies were described by Schwaber (1996) as follows: *"If a process can be fully defined, with all things known about it so that it can be designed and run repeatably [sic] with predictable results, it is known as a defined process, and it can be subjected to automation. If all things about a process aren't fully known-only what generally happens when you mix these inputs and what to measure and control to get the desired output-these are called empirical processes… A defined process is predictable; it performs the same every time. An empirical process requires close watching and control, with frequent intervention. It is chaotic and unrepeatable, requiring constant measurement and control through intelligent monitoring."*[10]

In the second half of the 1990s, different software languages, different locations, different projects started to be used in parallel by many practitioners. Country based different agile methods and their developers are shown in Table 2.

Table 2. *Agile models*

Country	Method	Developer
U.S.A.	eXtreme programming (XP)	Ken Beck & Eric Gamma
	Scrum	Ken Schwaber & Jeff Sutherland
	Crystal	Alistair Cockburn

[10] Schwaber, K. (1996). Controlled chaos: Living on the edge. American Programmer, 9, 10-16.

	Adaptive software development (ASD)	Jim Highsmith
	Lean software development	Tom & Mary Poppendieck
Europe	Dynamic systems development method (DSDM)	Dane Faulkner
Australia	Feature driven development	Peter Code & Jeff DeLuca

Priorities according to the agile approach are presented in Table 3.

Table 3. *Priorities in Agile Manifesto*

More Valuable Items	Less Valuable Items
Individuals and interactions	Processes and tools
Working software	Comprehensive documentation
Customer collaboration	Contract negotiation
Responding to change	Following a plan

According to agile values;

- (1) Individuals and interactions are more important than processes and tools.
- (2) It focuses on software instead of extensive documentation.
- (3) Customer collaboration is more significant than contract negotiation.
- (4) It focuses on responsive to change instead of following a plan.

2.3.1. Agile Manifesto and Model's Key Characteristics

There are twelve principles behind the agile in Table 3. In 2001, Beck et all presented the properties of agile. [11]

Table 4. *Principles of Agile Manifesto*

No	Principles
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 thought 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.

[11] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

2.3.2. Waterfall vs. Agile Methods

The basic distinction between waterfall and agile is specified in two statements.[12]

- Customer satisfaction with early and continuous delivery.
- The agility harness changes for the competitive advantage of the customer.

The differences between waterfall and agile models are examined in *Figure 3*. In the waterfall model, requirements are fixed and resources and date dimensions are estimated. On the contrary, in the agile model, the sources and the date dimensions are fixed and the requirements are estimated. [13]

Another important difference between the two approaches is that in the traditional approach, the customer is at the beginning of the project and in the agile approach is in the whole process.[14]

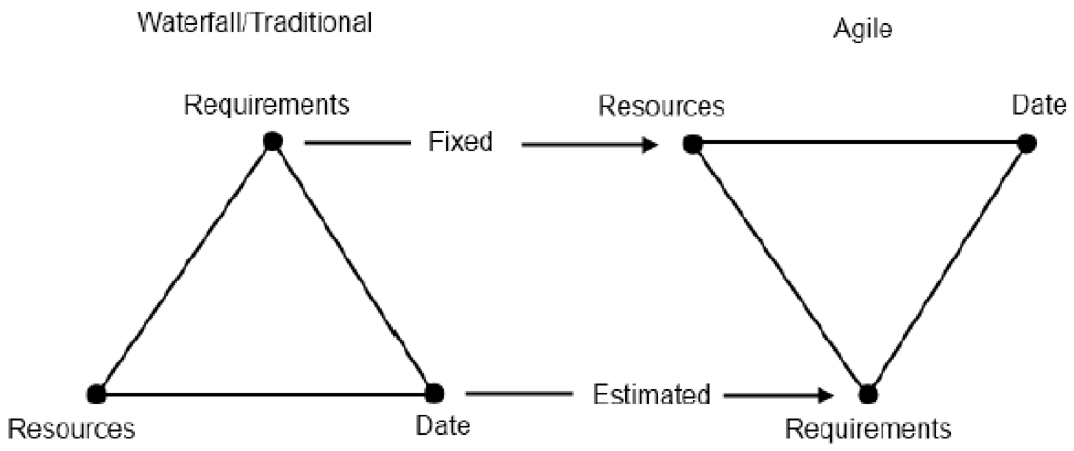


Figure 3. Waterfall vs. agile models[15]

[12] Leffingwell, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series). Boston: Pearson Education, LTD.
[13] Leffingwell, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series). Boston: Pearson Education, LTD.
[14] Paetsch, F. & Eberlein, A. & Maurer, F. (2003). Requirements engineering and agile software development. In Enabling Technologies: Infrastructure for Collaborative Enterprises, 308-313.
[15] Leffingwell, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series). Boston: Pearson Education, LTD, p.17.

Huo et all mapped waterfall and agile approaches as *Figure 4*.[16] As a result of the mapping, they found that the agile development phases included some applications in software quality protection practices and supportive roles.

In addition, the Standish Group CHAOS[17] report results for 2011 are given in *Figure 5*. According to the report, projects using agile approach are seen to be about three times more successful than waterfall. On the other hand, quite a few projects have failed with using agile approaches.

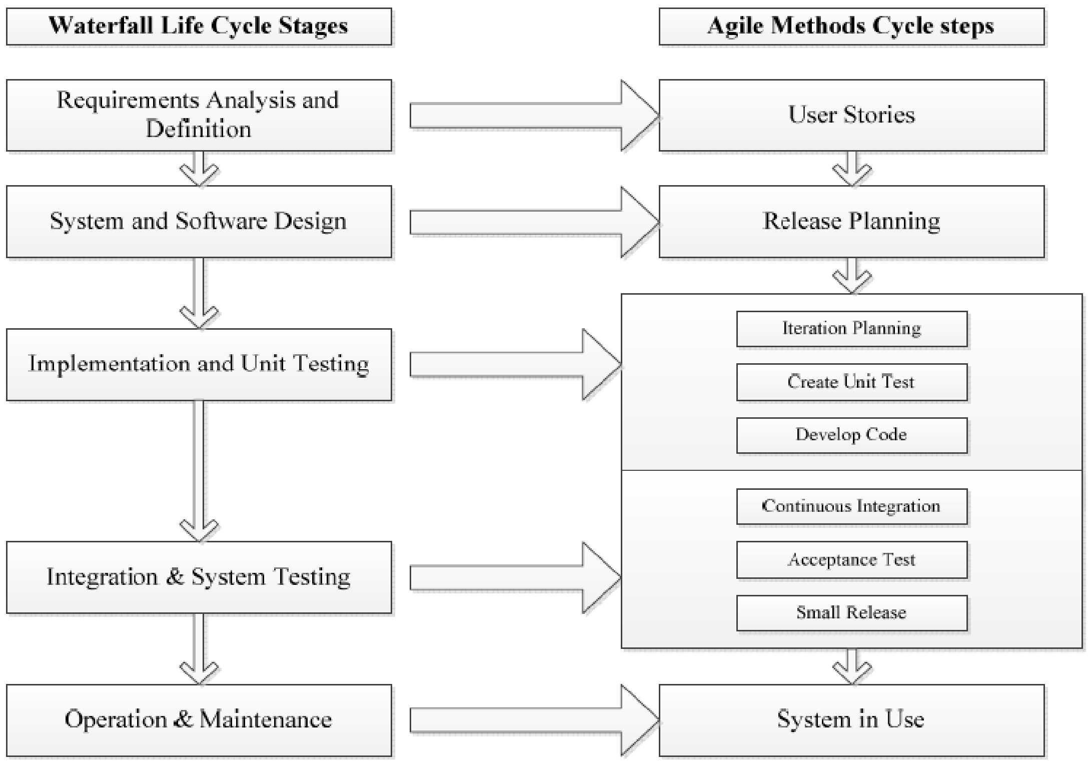

Waterfall Life Cycle Stages
Agile Methods Cycle steps
Requirements Analysis and Definition
User Stories
System and Software Design
Release Planning
Implementation and Unit Testing
Iteration Planning
Create Unit Test
Develop Code
Continuous Integration
Acceptance Test
Small Release
Integration & System Testing
Operation & Maintenance
System in Use

Figure 4. Life-cycle comparison of waterfall vs. agile models[18]

[16] Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and Agile methods. Proceedings of the annual International conference on Computer software and applications. (1), 520-525.
[17] The Standish Group (2011). CHAOS. West Yarmouth, MA: The Standish Group International.
[18] Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and Agile methods. Proceedings of the annual International conference on Computer software and applications. (1), 520-525.

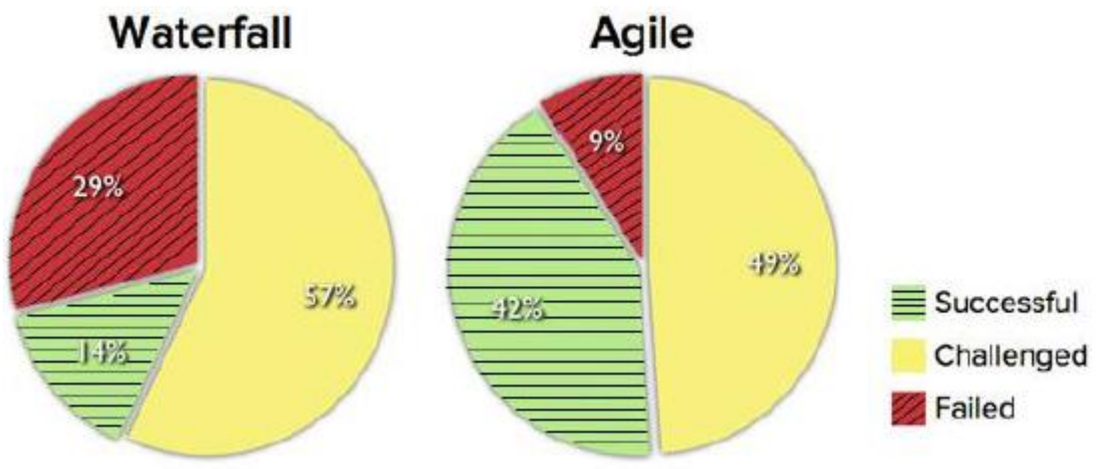


Figure 5. Success comparison of waterfall vs. agile models[19]

2.3.3. Continuous Integration

The term of *"continuous integration*" was the idea of Grady Booch and he placed the term in his method at 1991. Continuous Integration briefly can be defined as productive, beneficial and accommodating way of developing software and applications. It includes elevated automation technology processes with accommodating tools and environments.[20] Briefly, it is a way of preparing releases for applications and systems.

According to Martin Fowler, it is explained as: *"continuous integration is a software development practice where members of a team integrate their work frequently; usually each person integrates at least daily – leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."*[21]

[19] Cohn, M. (2012). Agile Succeeds Three Times More Often Than Waterfall.
[20] Vöst, S. (2016). Towards Continuous Integration and Continuous Delivery in the Automotive Industry, p. 1-5.
[21] Fowler, M. (2006). Continuous Integration, p.1.

Before continuous integration, development teams used to consist of less people. Source code used to be shared by USB keys and/or shared drives. Software development was barely able to be completed at the end of the project deadlines; there was no bandwidth for software testing. Every process of software life-cycle used to do manually.

When continuous integration became a common practice, problems about code versioning and servers emerged. To solve those problems, new terminology is created: Version Control Systems (VCS) and Automated Testing.

With the digitalization and technological improvements, competition and pressure from sales teams increased and this situation created the necessity for projects to be completed in a short time with numerous features. Enterprise companies enlarged their IT departments and agile teams to compete and survive in their sectors. Software teams included more than 1000 engineers and it was not possible to work together by sharing projects via USB keys or shared drives. So, they needed to work asynchronously on a single line of code at the same time with 15 other developers.

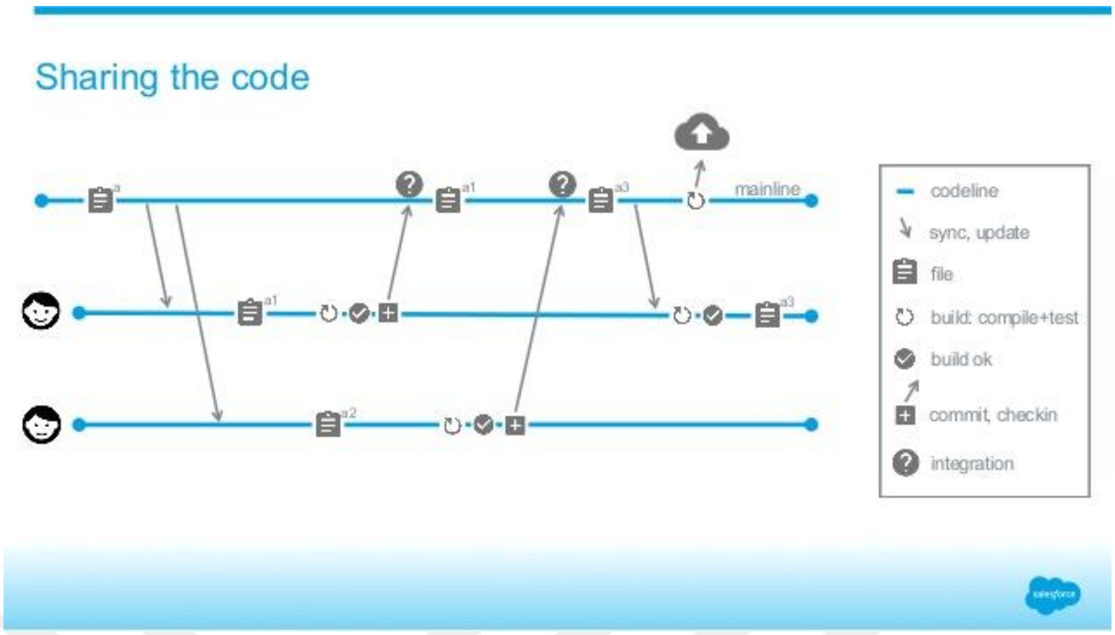


Figure 6. Continuous integration: Code sharing[22]

Primitive version of *continuous integration* is shown in *Figure 6*. In this figure, it is explained that there was no problem while sharing the code and working on the same project together because no one starts working on a file which is being edited by another person, so that there is no need to merge anything in the project. In this context, it is easier way that every worker uses their local project then pushes the changes on the master baseline.

In the *Figure 7*, it is explained that merge conflicts always occur in this primitive version but not on the local machine, on the master baseline. Because of that, after every push to the master baseline, someone has to fix the problems and make the project working as well as before.

[22] Mourot, C. (2014). A Story of Continuous Integration, p.6.

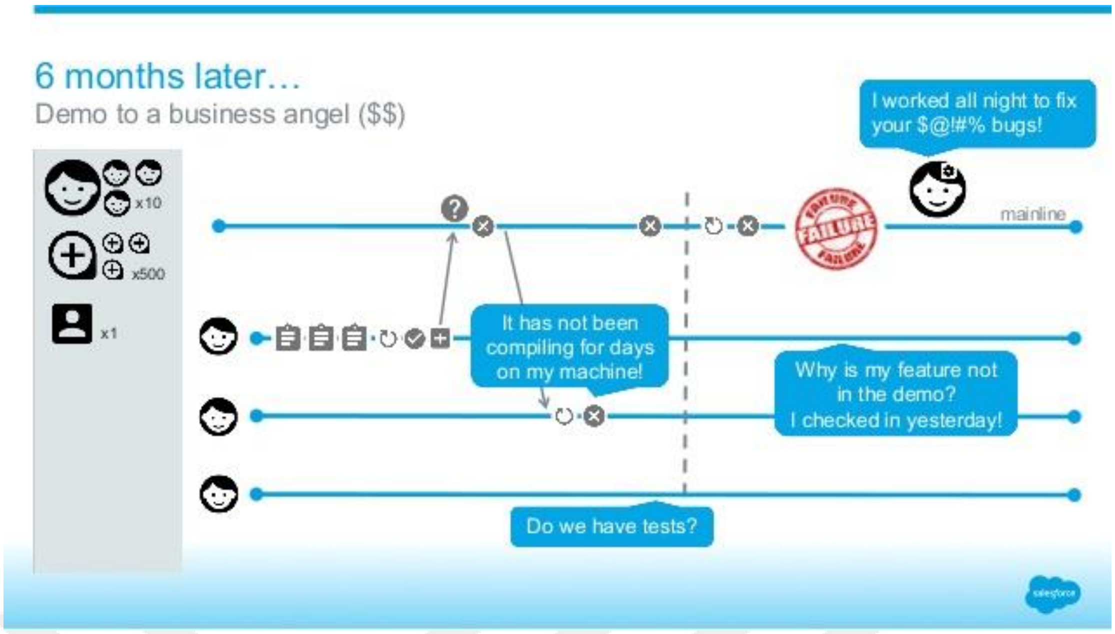
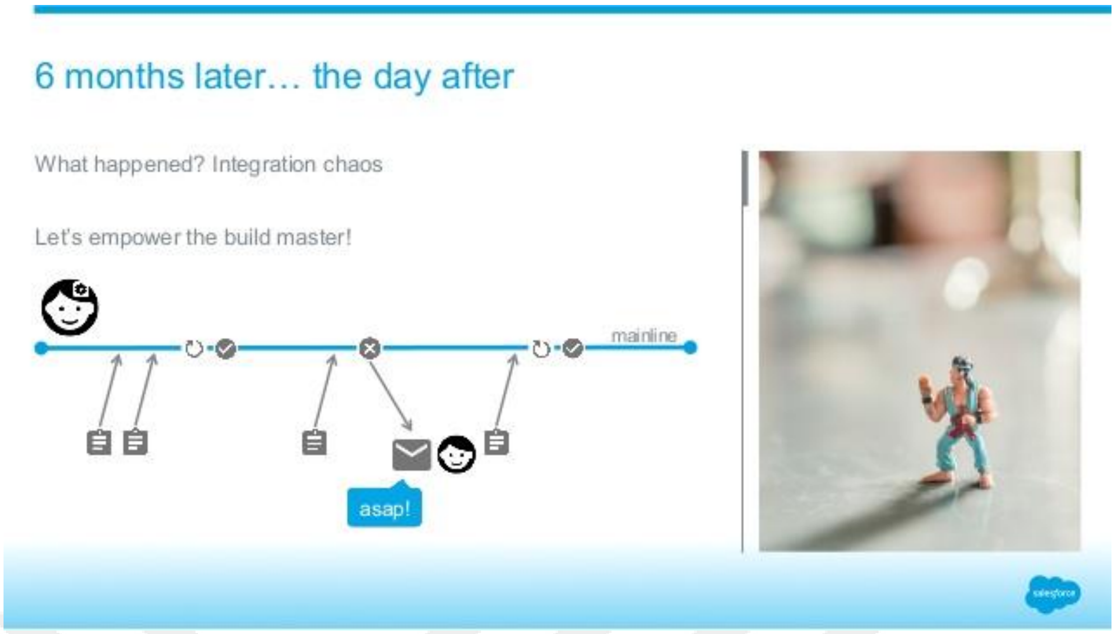

6 months later…
Demo to a business angel ($$)
x10
x500
x1
I worked all night to fix your $@!#% bugs!
mainline
FAILURE
It has not been compiling for days on my machine!
Why is my feature not in the demo? I checked in yesterday!
Do we have tests?
salesforce

Figure 7. Continuous integration: Merge problems[23]

In the *Figure 8*, an environment with a single mainline is demonstrated. Everyone works on their own local machine but they check in and check out files on the only baseline. This method solves the merge conflicts but creates more problems like inability for more than one person to work on the same file. Because there is only one baseline and if someone wants to edit some file, it is necessary to make sure no one is checked that file in before.

[23] Mourot, C. (2014). A Story of Continuous Integration, p.8.



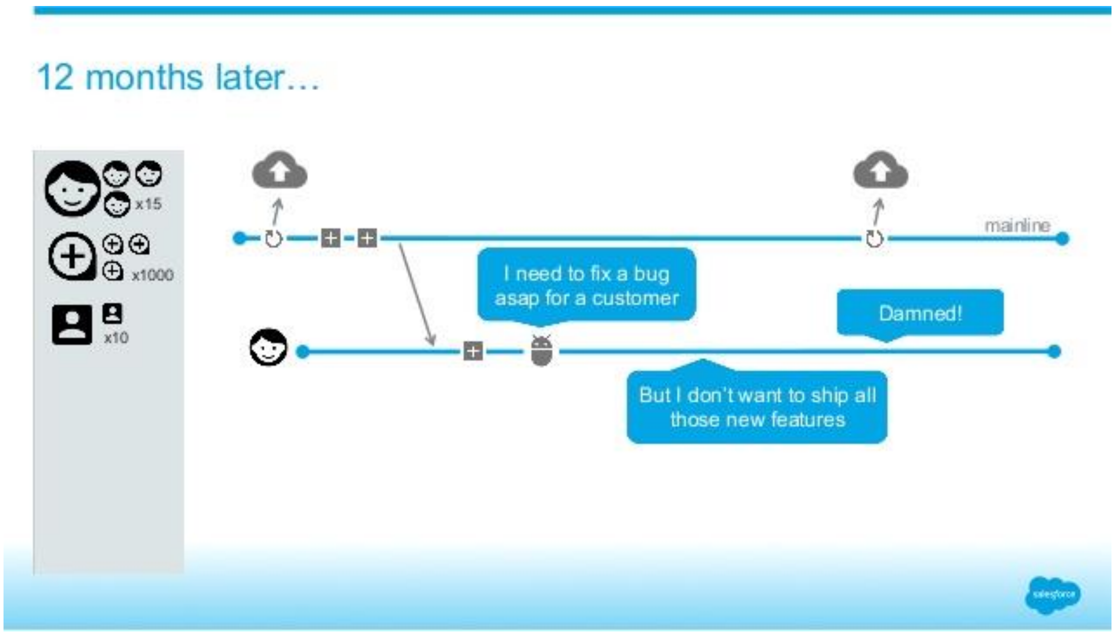
6 months later… the day after

What happened? Integration chaos

Let's empower the build master!

Figure 8. Continuous integration: Only one code line[24]

In the single line version there wasn't a way for bug fixes and new features to be separated. Solution of this problem led to a versioning system. Need for versioning explained in the *Figure 9*.



12 months later…

x15
x1000
x10
mainline
I need to fix a bug asap for a customer
Damned!
But I don't want to ship all those new features

Figure 9. Continuous integration: Version system[25]

[24] Mourot, C. (2014). A Story of Continuous Integration, p.10.
[25] Mourot, C. (2014). A Story of Continuous Integration, p.12.

In the *Figure 10*, it is explained why and how branch term created. For both versioning and monitoring different releases, branches make an easier work environment. With the help of these branches, bug fixed and new features were able to be separated from each other.

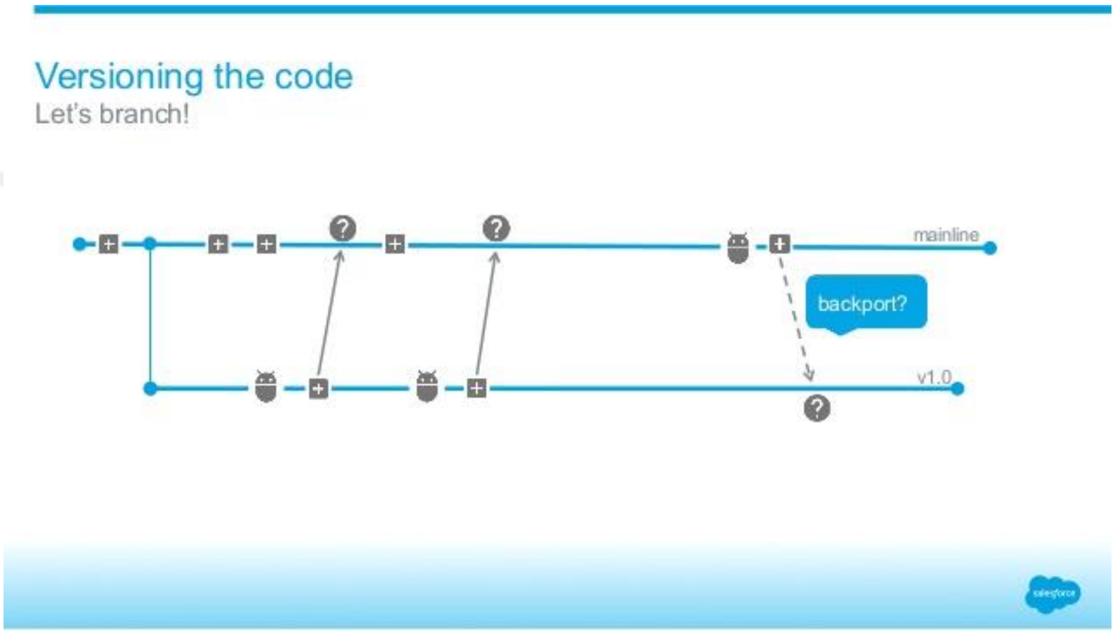


Figure 10. Continuous integration: Branch term[26]

After every commit and push to master branch, a control system was needed to check if the product is still working or is it still satisfies the business requirements. This situation is shown in *Figure 11*. This problem could be solved by testing the product after each change,

[26] Mourot, C. (2014). A Story of Continuous Integration, p.13.

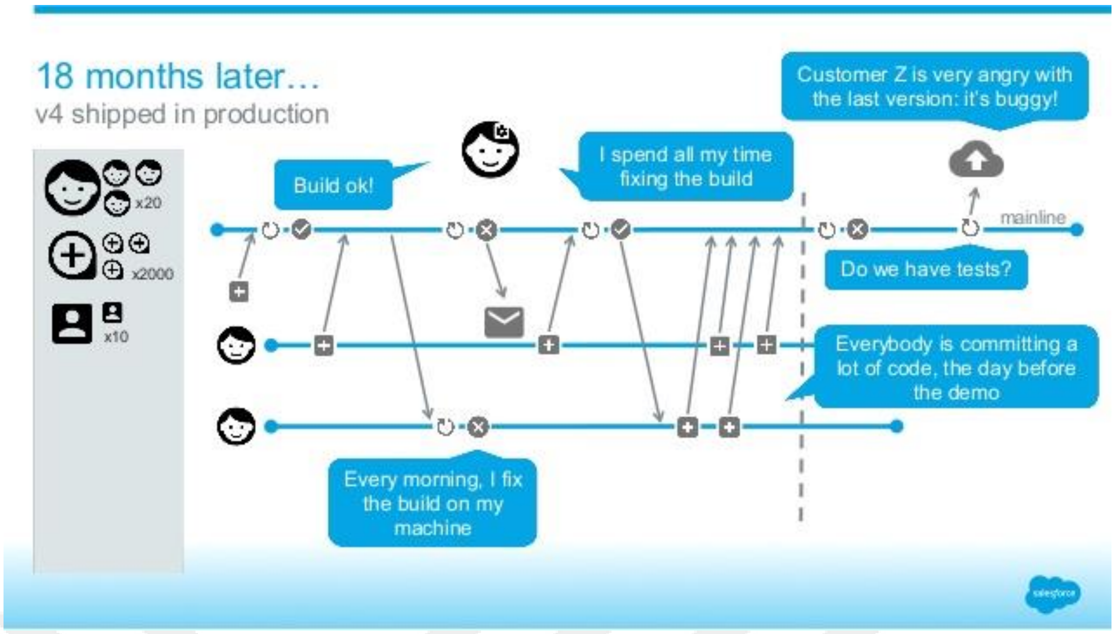

18 months later…
v4 shipped in production
x20
x2000
x10
Build ok!
I spend all my time fixing the build
Customer Z is very angry with the last version: it's buggy!
mainline
Do we have tests?
Everybody is committing a lot of code, the day before the demo
Every morning, I fix the build on my machine
salesforce

Figure 11. Continuous integration: Need for test[27]

According to *Figure 12*, it is explained why there is need for automation. It is similar to need for test but automation is an essential part of the system because it was impossible to extensively test a product takes too much time without automation.

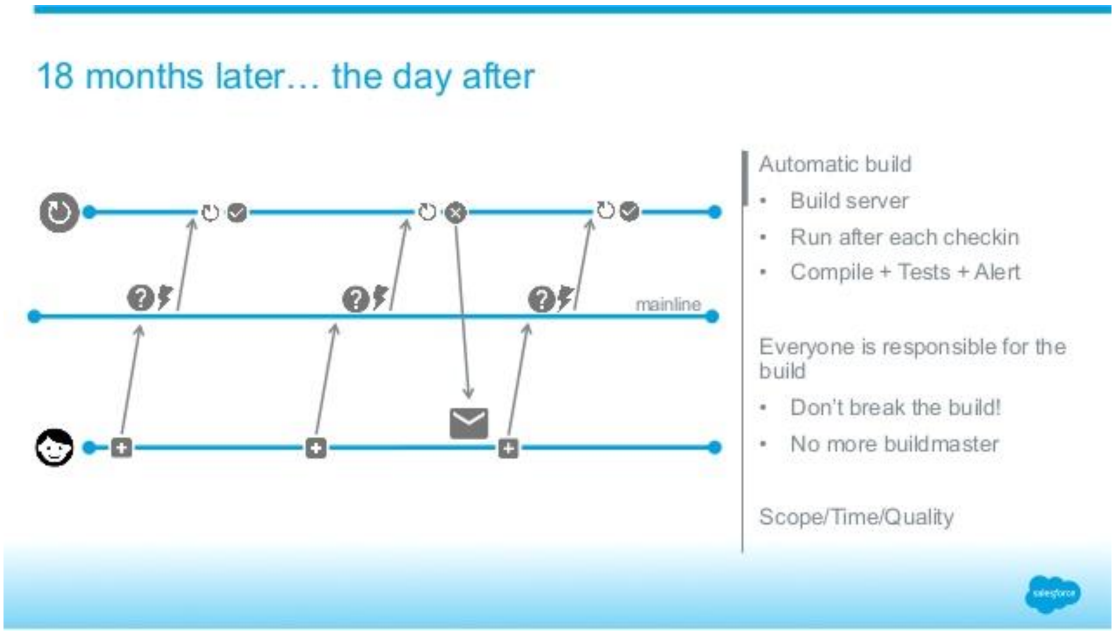


Figure 12. Continuous integration: Automation[28]

[27] Mourot, C. (2014). A Story of Continuous Integration, p.13.
[28] Mourot, C. (2014). A Story of Continuous Integration, p.14.

2.3.4. Continuous Delivery

Continuous delivery is a guarantee for the version/release of the application or the system that works well without any merge problems and bugs. In addition, continuous delivery can be able to decide if release version is clear and well enough to be implemented. According to Atlassian's definition for the term is as follows: "*Continuous delivery is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button. In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements*."[29]

On the other hand, there is a huge relevance between Continuous Integration (CI) and Continuous Delivery (CD). CD is a method that should be applied after CI transformation is complete. So that CD is a method which takes the CI as foundation to come into existence. In terms of implementation, it adds automated tests with high coverage and different aspects and automated deployment procedures to CI. This process, at the end, creates a product that is tested and ready for production use. To prevent confusion, differences between CI and CD is explained in *Figure 13*:

[29] Pittet, S. (2017). Continuous integration vs. continuous delivery vs. continuous deployment, p.5.

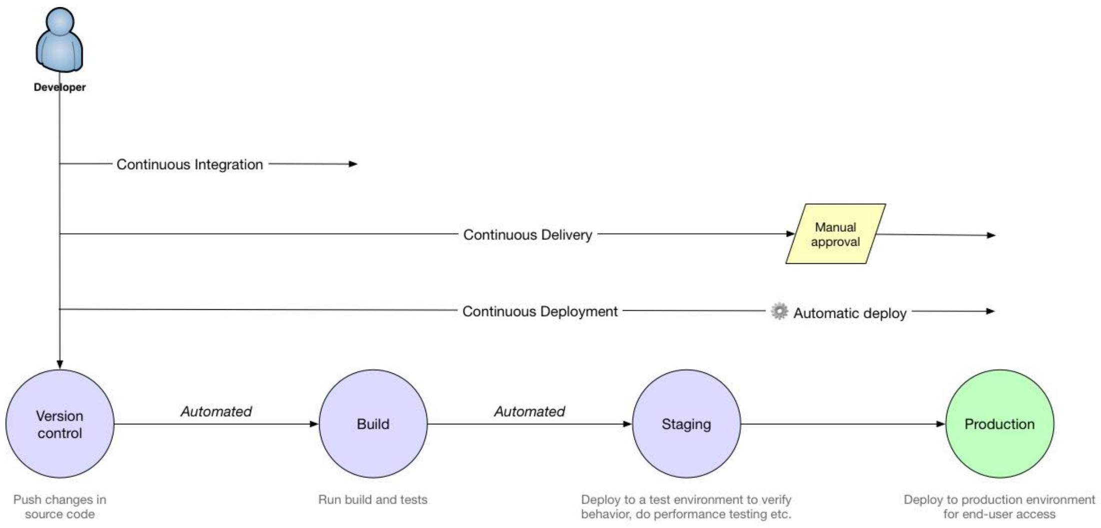


Figure 13. Difference between continuous integration and continuous deployment[30]

2.3.4.1. Need for Continuous Delivery

Software development lifecycle is getting shorter and shorted because of fast-changing market needs. Development processes cannot stay in their current positions in this environment because market is developing so that changing. When there is a need for deployment, test or any change, to perform that operations continuous delivery being a must for business models.

According to Microsoft Development Team, following *Figure 14* shows that the lifecycle of software development for DevOps (development & operations).

[30] Anastasov, M. (2017). What's the Difference Between Continuous Integration, Continuous Deployment and Continuous Delivery.

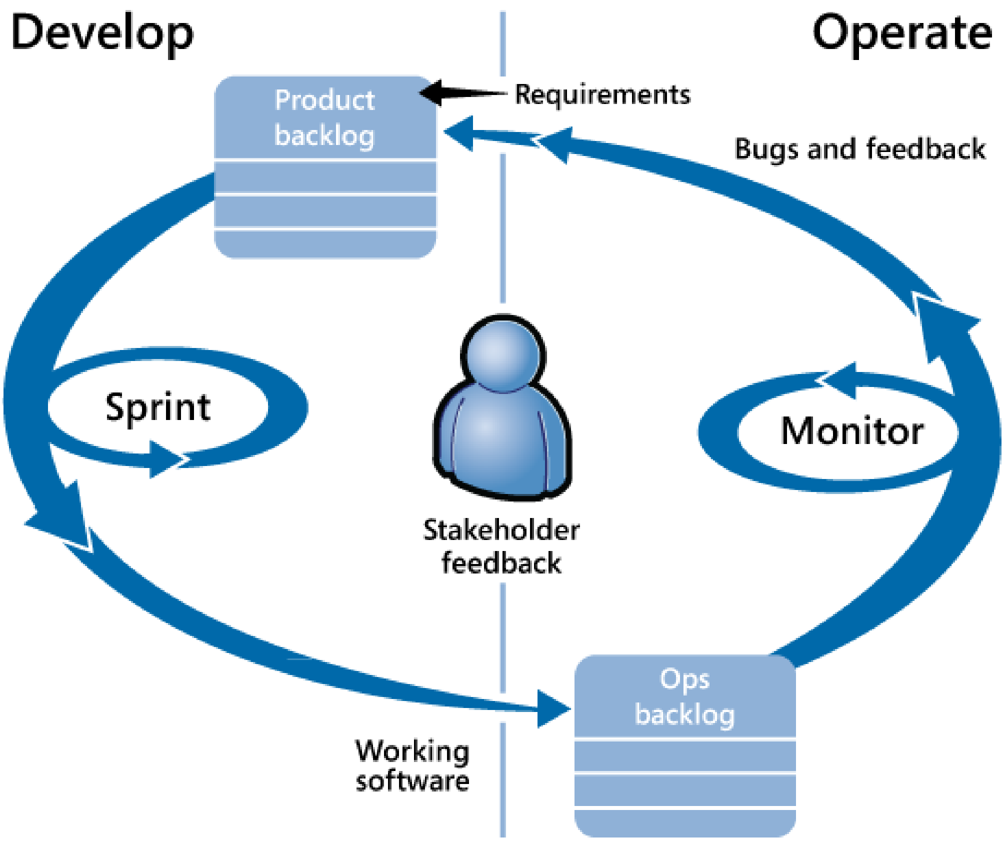


Figure 14. Software development lifecycle[31]

Software development processes and business models need continuous delivery according to following characteristics of Continuous Delivery:[32]

- **Faster reaction times:** In usually, for changes during or after the development it is always hard to pull or push the modifications and demands. Competitive business rules force the companies to make changes or fix bugs faster than “as fast as they can” so that operations in demand could be developed, tested, deployed in a very short time. Continuous Delivery supplies software development process the tools to come up to that speed and react faster.

[31] Microsoft Development Team (2012). Online Articles: Testing in the Software Lifecycle.
[32] Reed, J. P. (2018). The business case for continuous delivery.

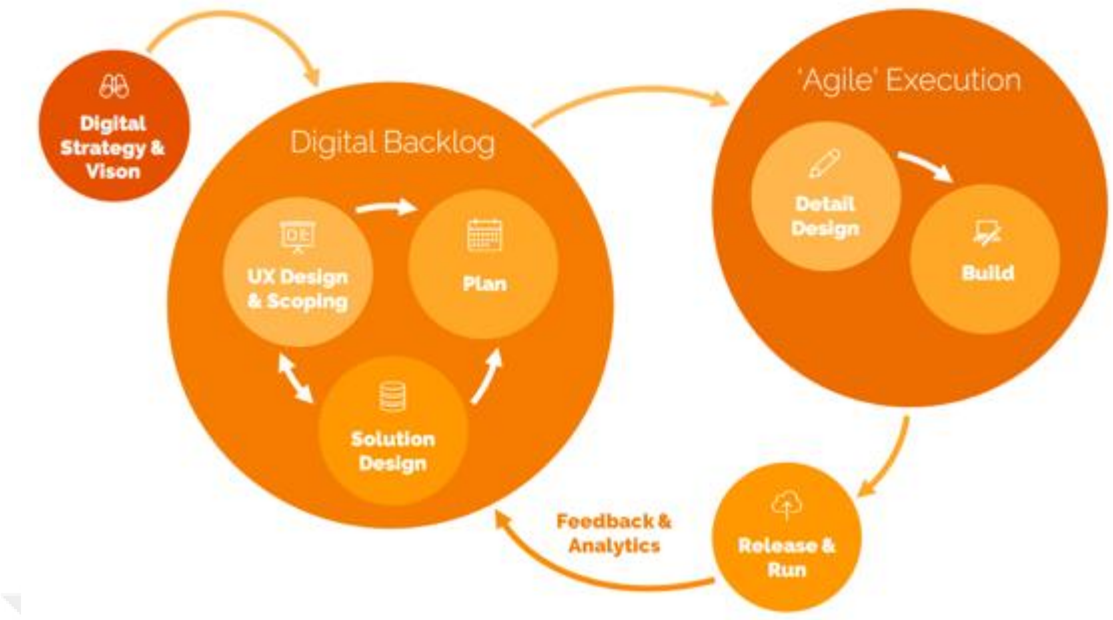


Figure 15. Continuous delivery business model[33]

- **Reduced risk:** When development for a demand is completed, it should be tested and simulated to see if there is a problem. When these test and simulations are automated, results of these operations are stable and trusted. On the other hand, manual tests, simulations and merge operations are high likely to end up in a defective product. A well-developed CD infrastructure might be able to cover most of the cases with a very low level of an error-generating characteristic.

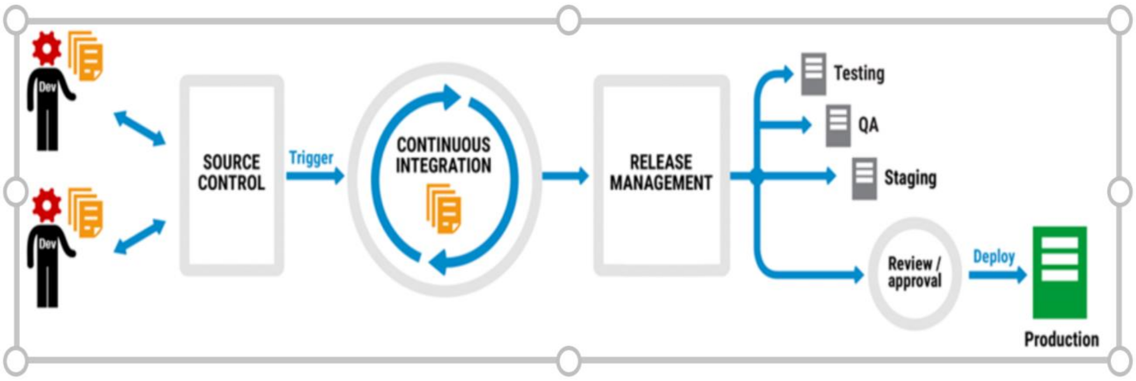


Figure 16. Reduced risk in continuous delivery[34]

[33] ClearPoint (2018). Digital Continuous Delivery.

- **Exposed inefficiencies and costs:** CD is an expensive concept to implement in business models. CD requires automated testing. For companies there is need for people to write automated tests and automate the configuration for tests. Moreover, developers must be trained for the use and develop the testing environments. However, these costs are calculable and base for the responsibility of "release- ready code and features".[35]

- **Flexible release options:** Companies can choose which features of their product are going to be released to their customers. This is also called feature toggling. According to Atlassian explanations written by J. Paul Reed flexible release options are explained as *"Specific sets of features can be released to specific customers, or released to a subset of customers, to ensure they function and scale as designed."*[36]

- **Automated testing:** Automated tests are essential for CD operations. Test automation can be implemented on different levels of the software. Thus, there are different types of tests that could be automated: Unit tests, integration tests, acceptance tests, etc. Unit tests are tests of smallest pieces of software, it takes the piece and tests it in an isolated environment. Integration tests include tests of the whole thing after parts are integrated. On the other hand, acceptance tests take testing to a different level. Each test represents a customer requirement. If all acceptance tests are successful, it means that the product is accepted by the customer. All these

[34] Hilbert, M. (2017). Bringing Continuous Delivery to the Database.
[35] Maxos (2017). Continuous Delivery.
[36] Reed, J. P. (2018). The business case for continuous delivery, p.7.

different tests are automated as a requirement of CD and save the company a huge time.

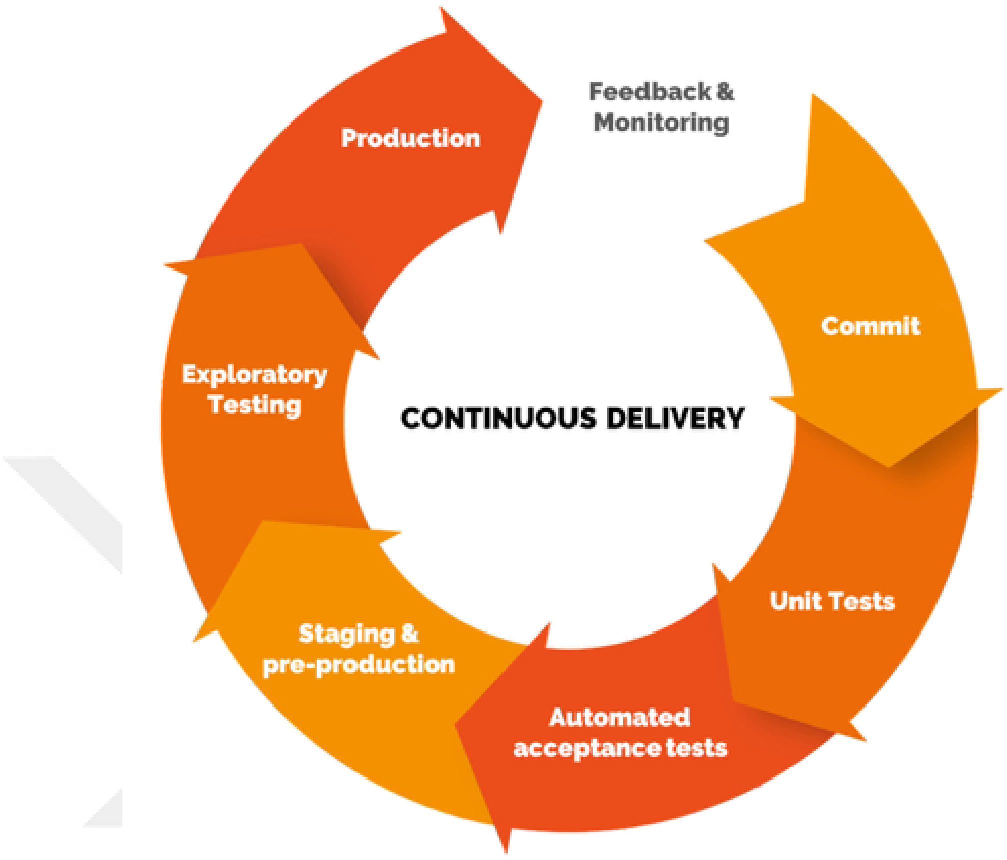


Figure 17. Continuous delivery lifecycle[37]

2.3.4.2. Digital Enterprise Solutions

According to Enterprise Management Associates (EMA) reports, companies, where practices of agile methodologies are applied in business models and IT, are growing faster than the others. Continuous Delivery and DevOps (Development and Operations) are most popular techniques in most modern companies. Digital enterprises companies are willing to both manage and deliver product or services in an automatic and risk-free way. However, enterprises do not have positive approach

[37] ClearPoint (2016). Digital Continuous Delivery.

to ways which are exhausting and require lots of exertion. So that Continuous Delivery and DevOps are proper for digital enterprises.[38]

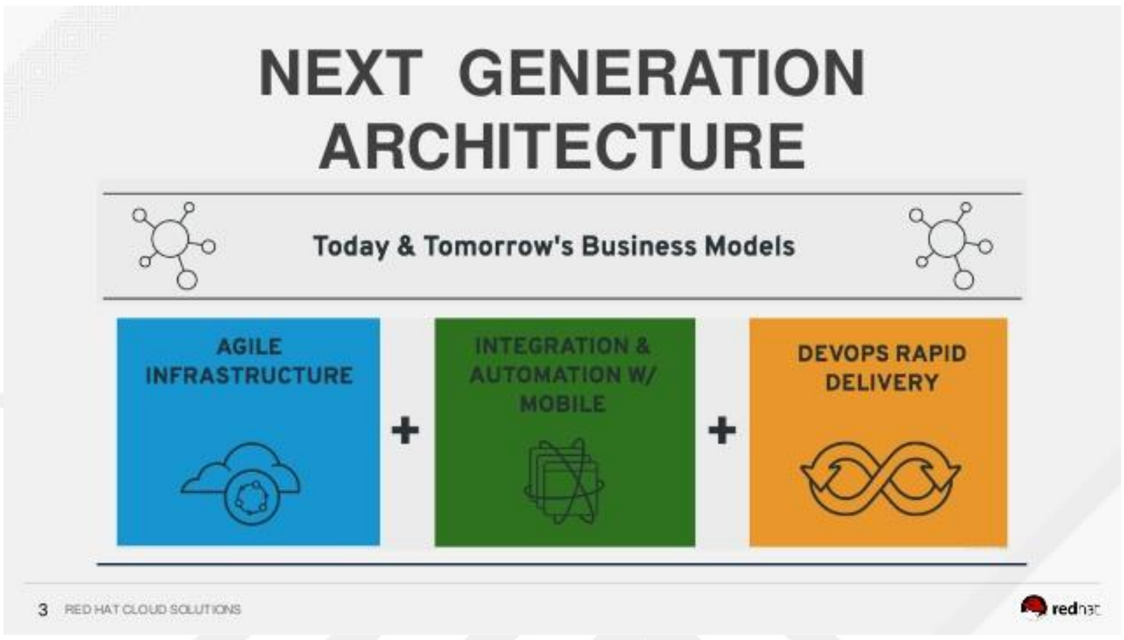

NEXT GENERATION ARCHITECTURE
Today & Tomorrow's Business Models
AGILE INFRASTRUCTURE
+
INTEGRATION & AUTOMATION W/ MOBILE
+
DEVOPS RAPID DELIVERY
3 RED HAT CLOUD SOLUTIONS
redhat

Figure 18. Next generation architecture[39]

According to the EMA report, enterprises should implement the next generation architecture to be successful and survive in the competitive world which is currently shaped by digitalization and the technology. By occurrence of next generation architecture, lifecycle of DevOps/Continuous Delivery is being more important than ever. The new lifecycle shows endless operations and processes should be done periodically.

[38] Craig, J. (2017). DevOps/Continuous Delivery Tooling, Launchpad for the Digital Enterprise.
[39] Kuncoro, A. (2016). Oracle Database Consultant. Solutions for IT Organizations on The Journey to The Digital Enterprise.

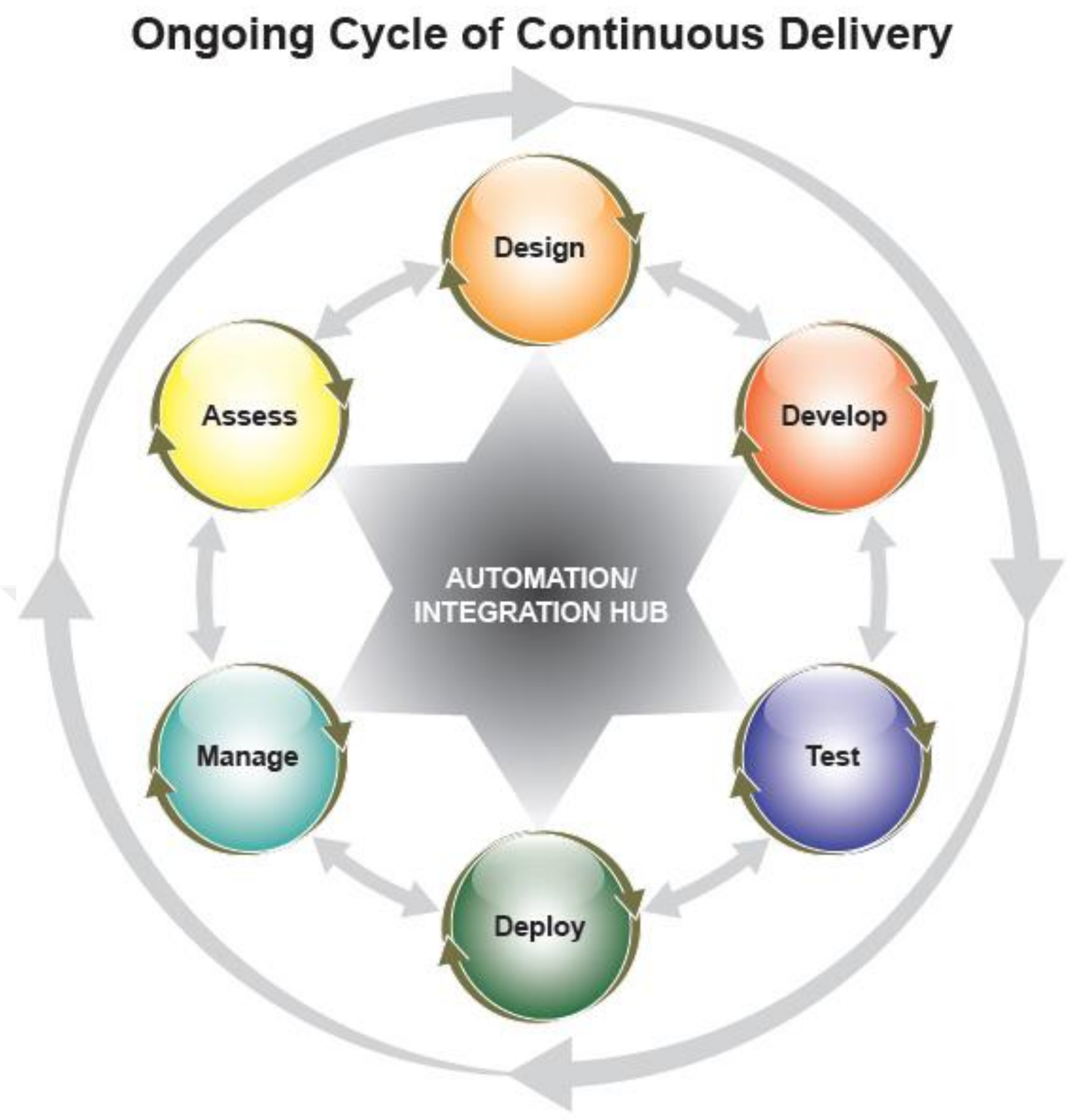


Figure 19. Next generation DevOps lifecycle[40]

The life cycle is explained from EMA report as: "*The reality is that Continuous Delivery can only be accelerated to the degree that each stage of the underlying lifecycle is accelerated and data sharing between stages is optimized. In other words, the Continuous Delivery process is only as fast as its slowest link, and data sharing is only as efficient as the integration capabilities of each tool supporting*

[40] Craig, J. (2017). DevOps/Continuous Delivery Tooling, Launchpad for the Digital Enterprise, p.3.

the lifecycle."[41] EMA report highlights the importance of the technology and digital economy on Continuous Delivery for business models. When digital economy is mentioned, technology is the first thing that comes to mind. However, digitalization is a culture; it does not only refer to technology. Deloitte explained the digital economy with details as: *"It's the economic activity that results from billions of everyday online connections among people, businesses, devices, data, and processes. The backbone of the digital economy is hyper connectivity which means growing interconnectedness of people, organizations, and machines that results from the Internet, mobile technology and the internet of things (IoT)."*[42] When the explanations are merged, the relation between digital economy and Continuous Delivery is unrolled: Technology affects the economy so much that importance of business and management data, etc. is extreme. Traditional management methodologies are not agile. So, they are not suitable for Continuous Delivery and DevOps concepts either. Thus, to be successful, digital transformation is a key word for business models. Briefly, companies should complete their digital transformation and apply modern techniques like agile methodologies, continuous delivery and DevOps. *Figure 20* presents the relationship between digital economy/digital transformations and DevOps.

[41] Craig, J. (2017). DevOps/Continuous Delivery Tooling, Launchpad for the Digital Enterprise, para.3, p.4.
[42] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, para.3.

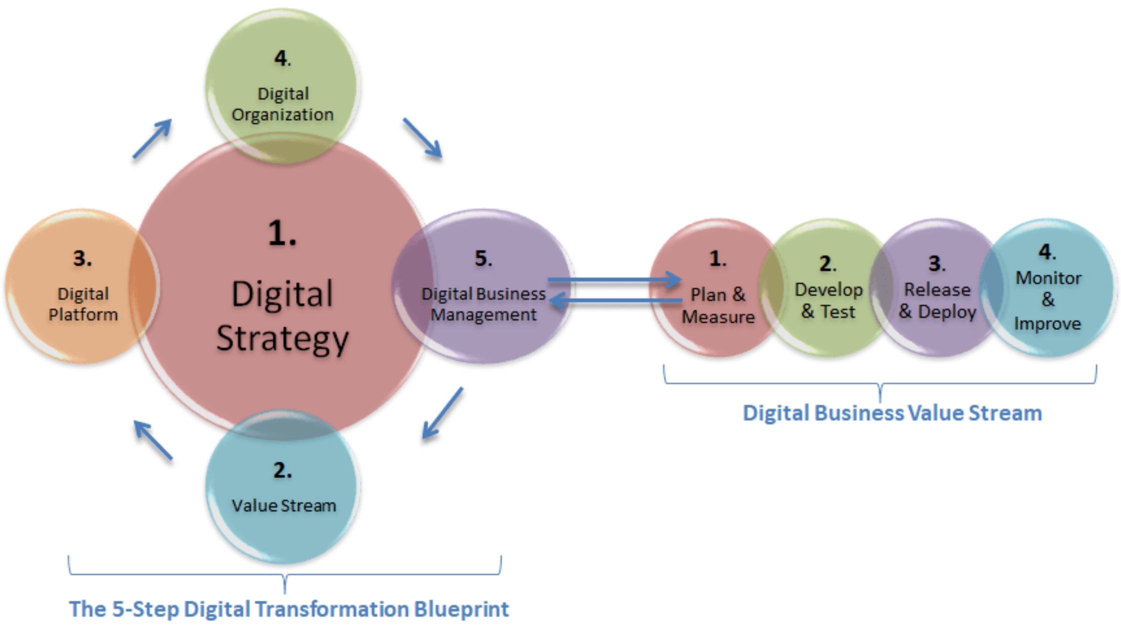


Figure 20. Relation between digital transformation and DevOps[43]

When technology is related the whole system, first of the main questions is how DevOps is related to business models. Second question is that why it is extremely crucial for enterprises. Briefly, DevOps automates the operations for every business model. According to Philippe Abdoulaye DevOps is responsible for following operations of companies[44]:

- Enterprise collaboration
- Business agility
- Short time-to-value
- Innovation culture
- Culture & Values
- People & Skills
- Methods & Practices
- Tools & Infrastructure

[43] Abdoulaye, P. (2017). Digital Transformation: How DevOps Could Have Revamped Toys R Us.
[44] Abdoulaye, P. (2017). Digital Transformation: How DevOps Could Have Revamped Toys R Us.

2.3.5. Continuous Deployment

Continuous deployment is the extension of continuous integration. The goal is to minimize the time between new code line development and new code line writing for live users in production. In continuous deployment, an infrastructure is required for all steps to be automated. Once the integration is successfully done in this way, the code in the live application is updated.

2.3.6. Being Agile and Implementing Requirements

In order to understand the meaning of being Agile, companies need a complex infrastructure which agile requirements such as continuous integration, continuous delivery and DevOps methods are implemented. In addition to this infrastructure, they need to get through a cultural change. Business leaders do not always manage these change processes well enough because there are always restrictions and resistance to any kind of change. To cope with the resistance, firstly, there is need for practices to find out what should be improved in the organization. Following figure shows the reasons behind this resistance in companies:

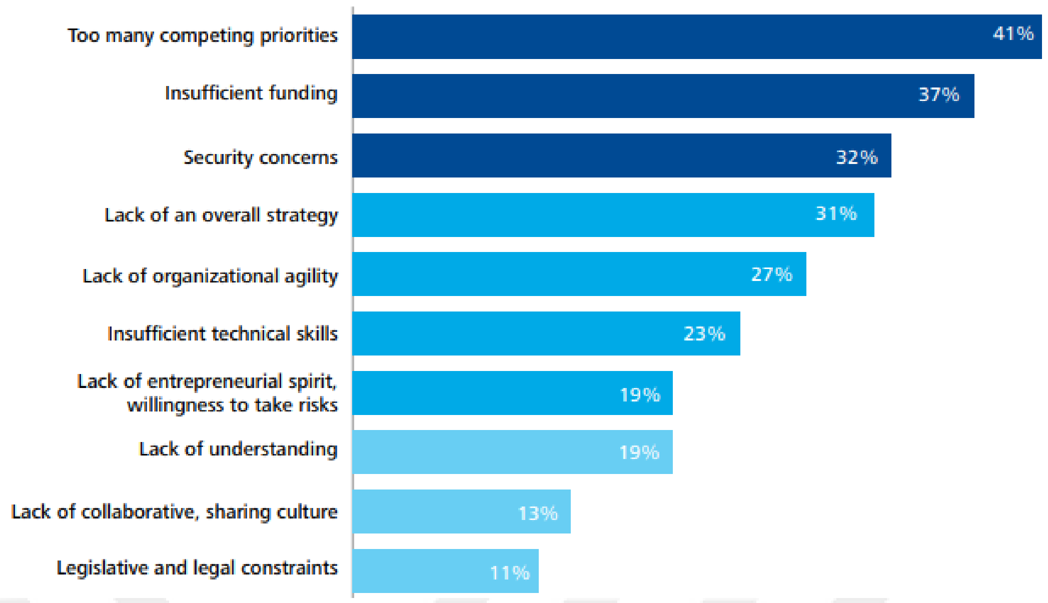
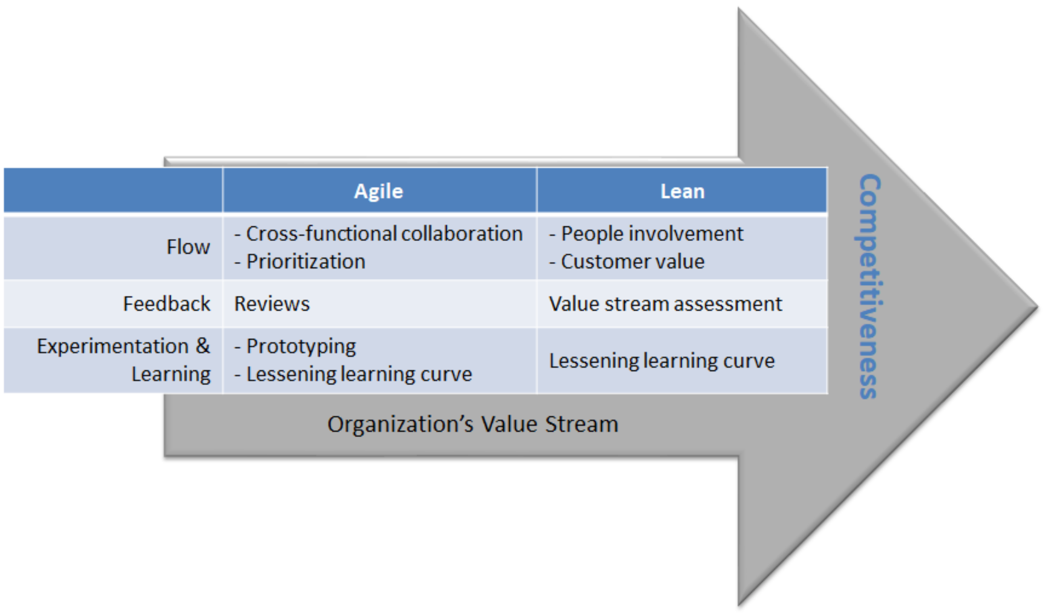


Figure 21. Reasons for resistance to agile transformation[45]

Software developers of the organizations should be encouraged and supported to embrace Agile transformation by the managers and leaders. When there is a defect that should be fixed, software developers probably are distressed about them. During agile and DevOps transformation, these problems will come to light and solved in the process. So, agile and DevOps methodologies will make happy employees (both managers and software developers).[46]

[45] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p. 12.
[46] Aiello, B. & Sachs, L. (2016). Drive adoption and overcome resistance to change.



	Agile	Lean
Flow	- Cross-functional collaboration - Prioritization	- People involvement - Customer value
Feedback	Reviews	Value stream assessment
Experimentation & Learning	- Prototyping - Lessening learning curve	Lessening learning curve

Figure 22. Agile methodologies in Competitive Business Word[47]

Managers and leaders should have clear goals and share these goals with details to each worker. Otherwise, it may cause a great risk to business operations and the way of completing the operations. Managers and leaders with certain decisions about the changes should not present funky approaches to change business flows and hold risks on their own hands without sharing.

To understand benefits of Agile & DevOps operations companies should understand the importance of "digital transformation". According to Deloitte University Press report, some questions are asked to companies to highlight the importance of new management techniques like Agile and automation systems like DevOps.[48] Following figure present the consciousness of the digital impact:

[47] Abdoulaye, P. (2017). Digital Transformation: How DevOps Could Have Revamped Toys R Us.
[48] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press.

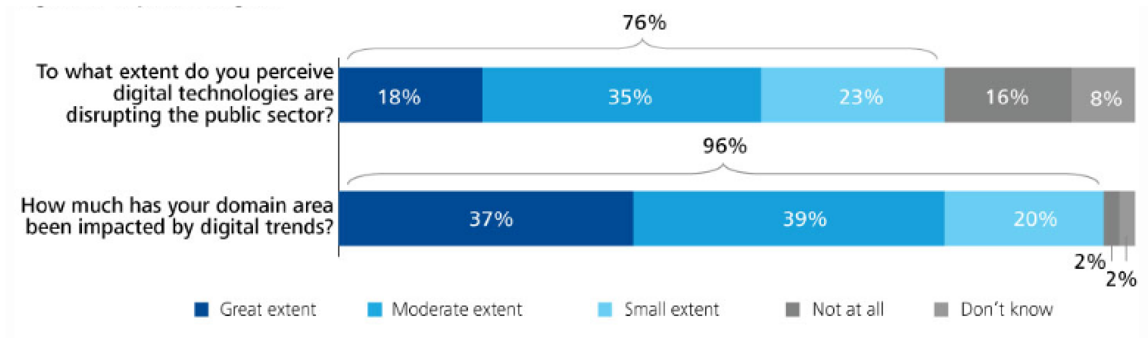


Figure 23. Digital Impact[49]

To be successful in terms of Agile and DevOps requirements, companies must reach to a certain digital maturity level. According to Deloitte report companies should implement following digital maturity chart for their operations[50]:

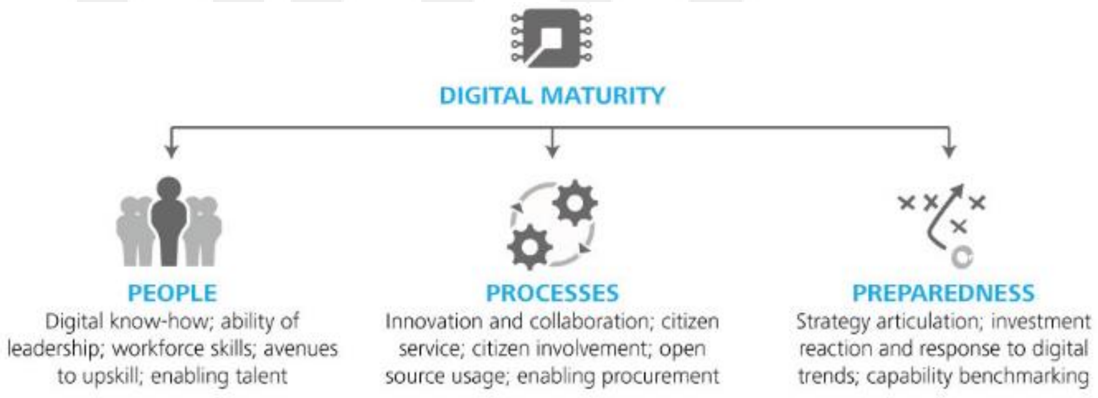
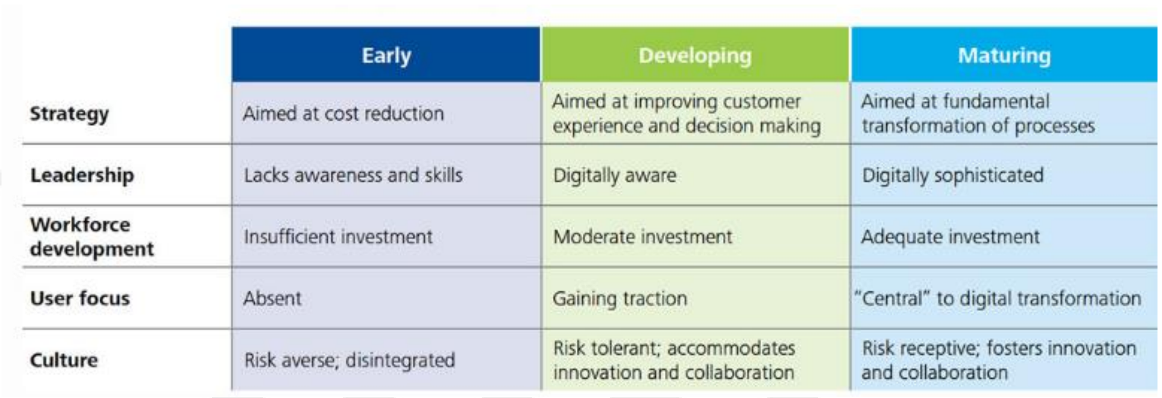

DIGITAL MATURITY
PEOPLE
Digital know-how; ability of leadership; workforce skills; avenues to upskill; enabling talent
PROCESSES
Innovation and collaboration; citizen service; citizen involvement; open source usage; enabling procurement
PREPAREDNESS
Strategy articulation; investment reaction and response to digital trends; capability benchmarking

Figure 24. Digital Maturity[51]

Digital strategies driven by managers might change the destiny of companies. If companies aim to get favorable results from their culture change process in terms of agile and DevOps, they must work on characteristics of digital maturing. When digital maturity is achieved, in terms of information technologies, it means that software

[49] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.5.
[50] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press.
[51] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.6.

delivery lifecycle is completely automated with minimized risks and also usage of agile and DevOps techniques are maximized. It is the most efficient way to increase the quality of the services/products companies provides. So that company-customer relationships will be greater than ever.[52] Characteristics of a digitally maturing organization are given from the following figure:

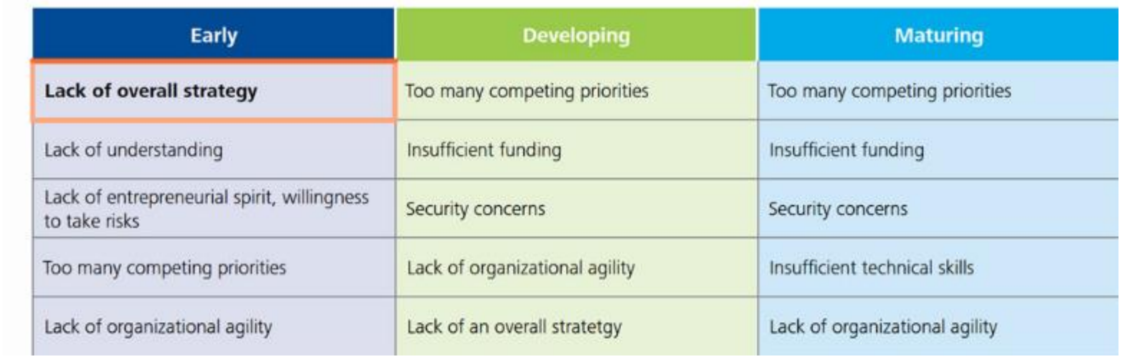


	Early	Developing	Maturing
Strategy	Aimed at cost reduction	Aimed at improving customer experience and decision making	Aimed at fundamental transformation of processes
Leadership	Lacks awareness and skills	Digitally aware	Digitally sophisticated
Workforce development	Insufficient investment	Moderate investment	Adequate investment
User focus	Absent	Gaining traction	"Central" to digital transformation
Culture	Risk averse; disintegrated	Risk tolerant; accommodates innovation and collaboration	Risk receptive; fosters innovation and collaboration

Figure 25. Characteristics of Digital Maturing [53]

According to Deloitte report, digital strategies, being digital, digital culture, improvements and developments do not only refer to fixing websites or presentations of fancy animations. Being digital, transforming to Agile and building DevOps techniques for whole of the company have objectives and barriers to these objectives as well. Following figure presents top barriers for digital transformation, being agile and implementing DevOps procedures:

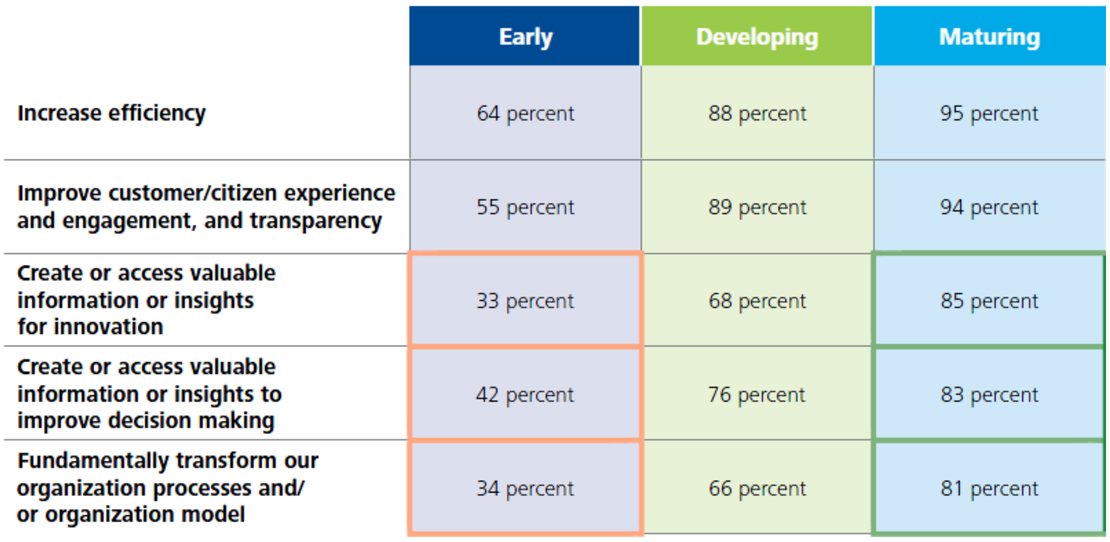
[52] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.9, para.3-5.
[53] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.8.



Early	Developing	Maturing
Lack of overall strategy	Too many competing priorities	Too many competing priorities
Lack of understanding	Insufficient funding	Insufficient funding
Lack of entrepreneurial spirit, willingness to take risks	Security concerns	Security concerns
Too many competing priorities	Lack of organizational agility	Insufficient technical skills
Lack of organizational agility	Lack of an overall stratetgy	Lack of organizational agility

Figure 26. Barriers of Digital Transformation [54]

To blot out the barriers in the transformation process, objectives should be clearly defined. In Deloitte Report, it is explained as: *"The importance that organizations place on using digital technology to improve innovation and decision making also varies by digital maturity level."*[55] Following figure presents the objectives of digital transformation, being agile and implementing DevOps:



	Early	Developing	Maturing
Increase efficiency	64 percent	88 percent	95 percent
Improve customer/citizen experience and engagement, and transparency	55 percent	89 percent	94 percent
Create or access valuable information or insights for innovation	33 percent	68 percent	85 percent
Create or access valuable information or insights to improve decision making	42 percent	76 percent	83 percent
Fundamentally transform our organization processes and/ or organization model	34 percent	66 percent	81 percent

Figure 27. Objectives of Digital Transformation, Being Agile, Implementing DevOps[56]

[54] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.9.
[55] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.9, para.1.
[56] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.10.

According to these objectives, it is clear that at maximum maturity level, companies would become game changers in their own sectors. According to Deloitte Report Summary it is summarized as: *"The exponential changes that drive digital transformation challenge the established models of leadership and governance. Before the ascent of digital technologies, new projects could be assessed through exhaustive analysis, investment decisions could be based on cost-benefit guidance, and the end destination of most plans was a fixed point. In the new digital era, leaders are required to make decisions more quickly in the face of a constant evolution in the art of the possible."*[57]

Outcomes make good show when requirements are applied properly in the companies. According to report, it is proven that lacking of technical skills is not a big problem, actually technical skills are unlimited. The most important skills to be successful in competitive world for companies are related to being agile. Following figure presents the necessary skills to be successful in terms of digital competitive rules and being agile:

[57] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.10, para.3.

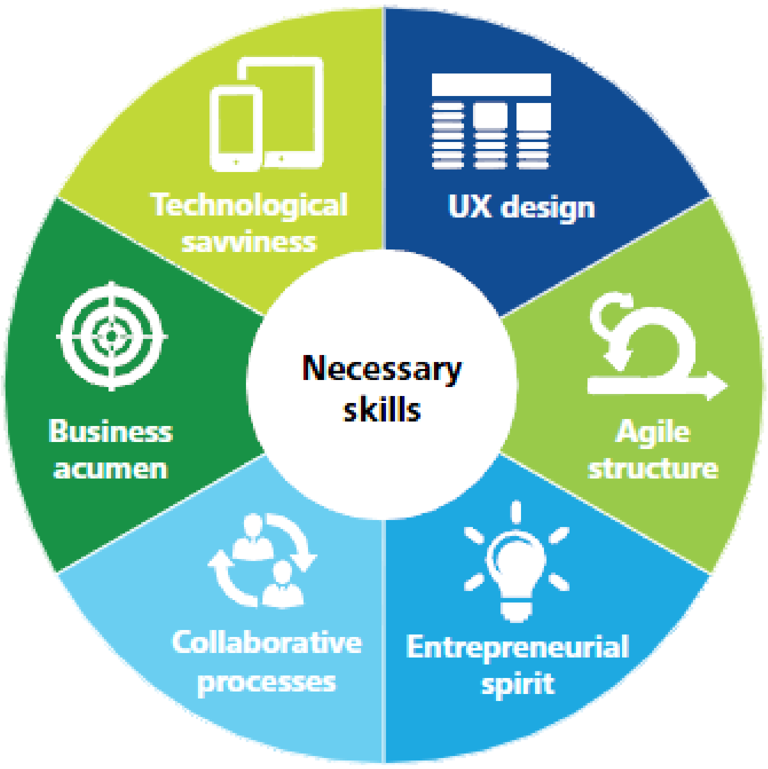


Figure 28. Skills for being successful in terms of being Agile[58]

Necessary skills are not only enough to be successful in agile processes. There is also need for drivers in the companies for being agile and implementing agile requirements. Following figure presents the necessary drivers to be successful in terms of being agile:

[58] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.15.

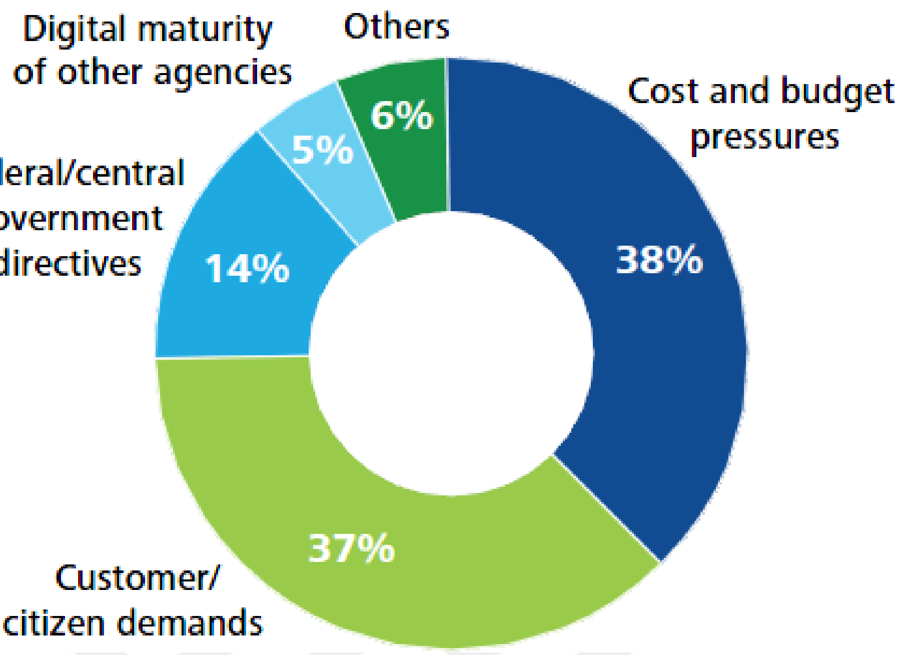


Figure 29. Drivers for being successful in terms of being Agile[59]

Moreover, it is proven that being agile and implementing agile processes are the biggest game changers in order to enable digital transformation in the company. Following figure presents the ways and their percentage to enable digital transformation in the company:

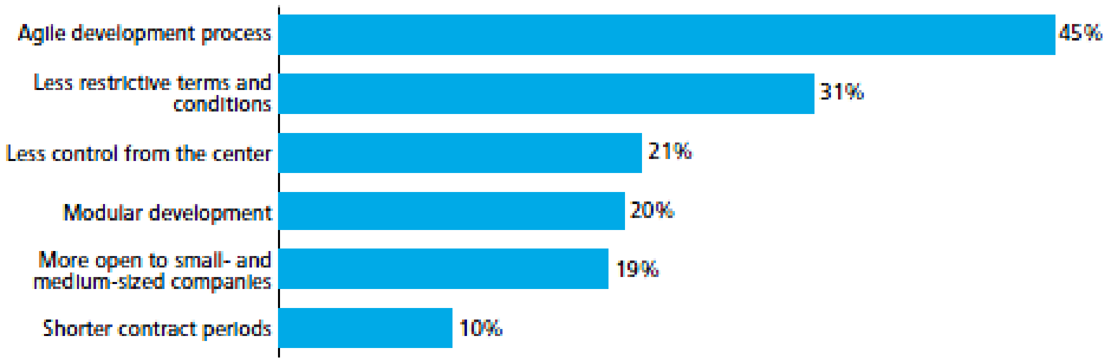


Figure 30. Ways and their percentage to enable digital transformation[60]

[59] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.18.
[60] Eggers, W. D., & Bellman, J. (2015). The journey to government's digital transformation. Deloitte Touche Tohmatsu Limited. Deloitte University Press, p.20.

As a brief summary, companies which aim to be successful in digital transformation and agile processes must have strong infrastructure in terms of agile culture and technologies. Skills and drivers must be a priority for them to create a strong approach in order to compete in digitalization.

2.4. Automation in Agile

To be a part of Agile, companies should take into consideration that all processes are dynamic. In methodologies like Agile, DevOps, CD and CI there is a secret ingredient called automation. Automation is not a stationary concept that only needs one-time only implementation. It is a continuous set of processes, a point of view, an approach and a living system.

Famous opposite view to agile is waterfall methodology. To understand the importance of automation agile and waterfall methodologies should be compared.

- Agile includes high level of whole system automation and integration, Waterfall does not.
- Agile development lifecycle is automated, Waterfall is not.
- Agile is faster way to provide demands of customers, Waterfall is not.
- Agile is flexible in terms of customers' demands, Waterfall is not.
- Agile provides higher performance than Waterfall.
- Agile includes automated tests, Waterfall does not.
- Agile provides higher quality of services and products, Waterfall does not.

Following figure presents the comparison of lifecycles of agile methodologies and Waterfall methodologies. It is clear that agile methodology lifecycle is continuous and automated:

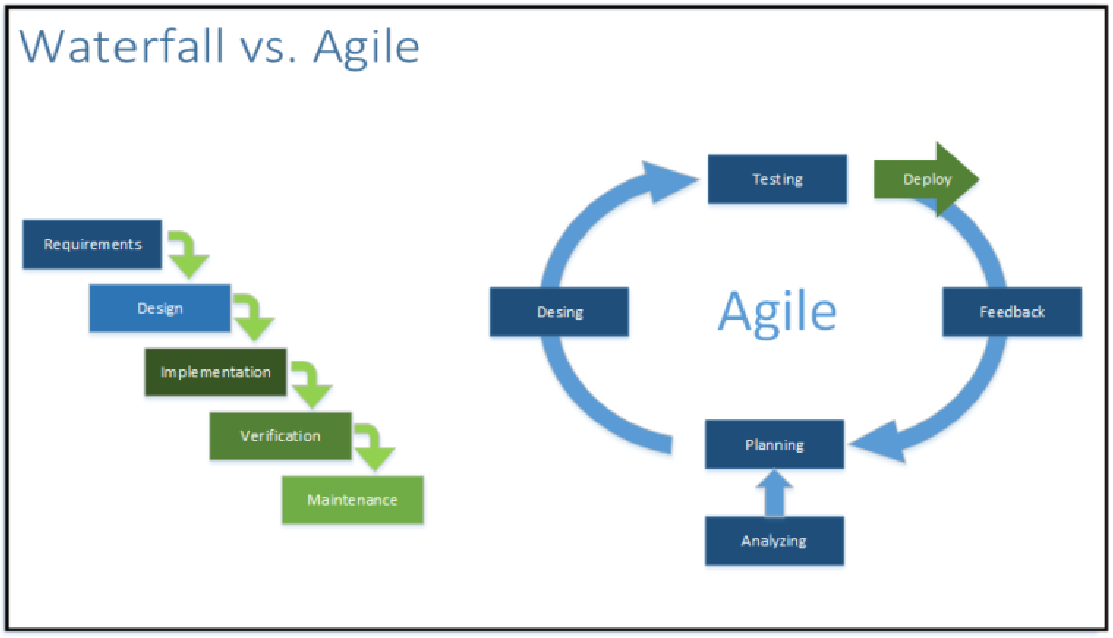


Figure 31. Comparison of Lifecycles of Agile Methodologies and Waterfall Methodologies[61]

Automation for whole system is an expensive strategy for companies. To be successful and fully agile, most of processes must be automated. Automation is only way to find out the defects before they cause any problem in product or services even for business and, or managers. It is a good strategy to solve the problems before they are presented to customers or end users as bugs. However, there are challenges to implement automation for whole systems in the companies. Automation systems are responsible to respond faster than they responded before. Teams must respond to automation systems' feedback as soon as possible and act accordingly.[62] So that when companies implement automation systems, they need dynamic and hardworking teams. Following figure shows feedback loops that are used in agile and continuous delivery techniques.

[61] Radwan, M. (2016). Introduction to software testing.
[62] Smith, D. (2015). Agile Feedback Loops.

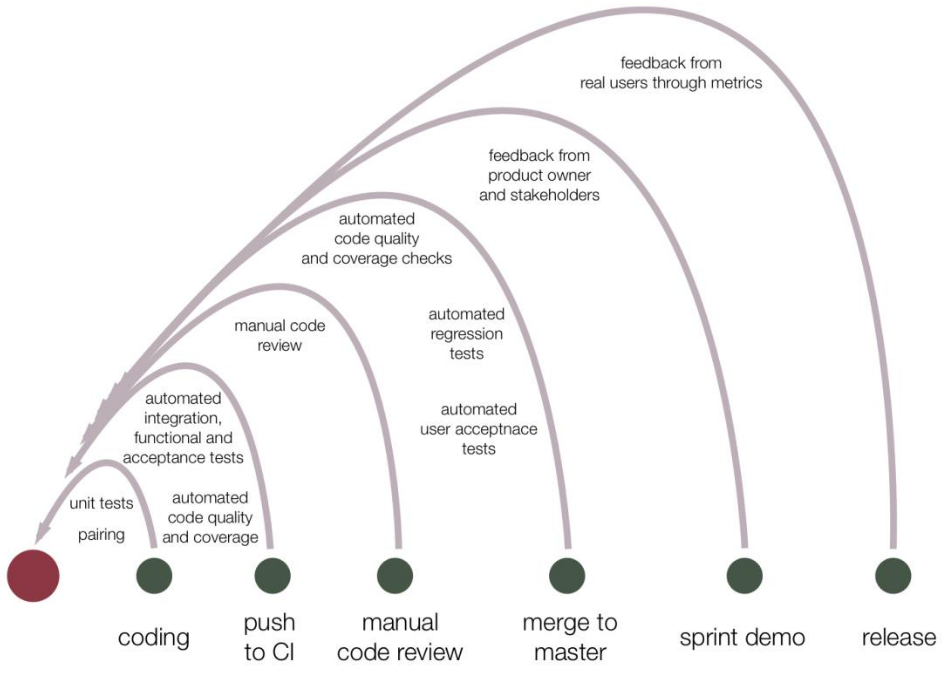

feedback from real users through metrics
feedback from product owner and stakeholders
automated code quality and coverage checks
manual code review
automated regression tests
automated user acceptnace tests
automated integration, functional and acceptance tests
unit tests
pairing
automated code quality and coverage
coding
push to CI
manual code review
merge to master
sprint demo
release

Figure 32. Feedback loops[63]

2.5. Project Success

It is crucial for project managers to include success definitions and consistent criteria for project managers. At the same time, these criteria must be accepted by project stakeholders.[64] According to Ika, the success of the project is the quality, time and cost success criteria.[65] In addition, Rosenau defined this set of triple criteria as a "triple constraint".[66] Sommer achieved success criteria; definition of budget factors, schedule, sponsor targets, defined requirements of features and functions, and customer satisfaction.[67]

[63] Smith, D. (2015). Agile Feedback Loops.
[64] Watson, P. (2009). A methodology for attaining and assessing project success for rehabilitation projects. Journal of Building Appraisal, 4(3), 181.
[65] Ika, L. (2009). Project success as a topic in project management journals. Project Management Journal, 40(4), 6-19.
[66] Rosenau, M. D. (1984). Project management for engineers. New York, NY: Lifetime Learning.
[67] Sommer, D. (2004). Project failure—12 mistakes to avoid. Paper presented at the PMI Global Congress. North America, Atlanta, GA.

In Section 2.3.2, we stated that agile projects are three times more successful than non-agile projects, according to the Standish Group (2011) report. The report showed, “*The agile process is the universal remedy for software development project failure. Software applications using agile process have three times the success rate of the waterfall method and a much lower percentage of time and cost overruns*”.[68] The project success for Standish Group is stated as follows: Projects completed with timely, budgeted and planned features are successful.

2.6. Project Failure

Project failure is a good opportunity for significant learning if assessments are made. This issue has been the focus of many researchers. In the CHAOS report of the Standish Group (2008), it has been found that software projects frequently use expressions "project challenged" (p.4) and "project impaired" (p.4). The prevalence of these failures is specified.[69] The success rate in the related report was only 16%. On the other hand, challenged projects have a rate of 53% and impaired projects have a rate of 31%. Significant conclusions can be drawn from the errors if the projects are evaluated despite the high rate of failure.

[68] The Standish Group. (2011). CHAOS. West Yarmouth, MA: The Standish Group International, p.25.
[69] The Standish Group. (2008). The trends in IT value. Yarmouth, MA: The Standish Group International.

3. ANALYSIS OF AGILE METHODS AND RESULTS

In this section, analyzes were made on the agile reports organized in the sponsorship of VersionOne between 2014 and 2016.[70,71,72] The reports used in the analysis were made with the participation of individuals in the global software development community. In the analyzes Agile was examined from seven different aspects. These are respectively: respondent demographics, company experience and adoption, benefits of agile, agile methods and practices, agile success and metrics, scaling agile and agile project management tools.

3.1. Respondent Demographics

Table 4 indicates the comparison of the size of the participants of the reports. Accordingly, when annual developments are examined, by 2016, the size distribution of firms has become more inclusive of all types of companies. Participants in large organizations with more than 20,000 employees increased by 20% in 2014 to 26% in 2016. On the other hand, there are 39% of the participants in 2016, although there are no participants in 2014 from organizations employing less than 1,000 people. This result provides us with the traces of the reports that all firms have become covered in time.

[70] VerisonOne. (2014). The 9th Annual State of Agile Report. VerisonOne.
[71] VerisonOne. (2015). The 10th Annual State of Agile Report. VerisonOne.
[72] VerisonOne. (2016). The 11th Annual State of Agile Report. VerisonOne.

Table 4. *Size of organization*

	2014	2015	2016
<1000 people		44%	39%
1,001 – 5,000 people	53%	17%	18%
5,001 – 20,000 people	35%	15%	17%
20,001+	20%	24%	26%

The intensity comparison of the participants employed in the groups working with software groups is presented in Table 5. The intensity among the participants of the survey seems to be generally stable even though they have slight changes over the years.

Table 5. *Size of software organization*

	2015	2016
<100 people	38%	32%
1,01 – 1,000 people	31%	34%
1,001 – 5,000 people	15%	15%
5,001+	16%	19%

Country comparisons of the survey participants are given in Table 6. For the first time in 2016, half of the survey participants came from outside North America. Thus, the North American dominance of the reports has been reduced over the years. In the report of 2016, it is seen that the participants became more distributed around the world

Table 6. *Respondent locations*

From	2014	2015	2016
North America	65%	56%	50%
South America		4%	5%
Europe	21%	26%	28%
Africa		1%	2%
Asia		11%	10%
Oceania		2%	4%

Table 7 provides a comparison of participants' internal duties with respect to years. Over the years there has been no significant change in the selection of participants according to their duties. This allows us to make more meaningful comments about the changes in value over the years in which the answered questions are. On the other hand, most participants have been project / program managers every year. However, the second and third orders are changing according to years. People working as development staff are in second place in 2014, but by 2016 they have only 15% of the total. By 2016, however, the second place was the Scrum Master of internal coach with 19%.

Table 7. *Respondent roles*

Responded	2014	2015	2016
Project/program manager	24%	24%	23%
Development staff	22%	18%	15%
Development leadership	14%	14%	13%
Other (Scrum Master or	13%	16%	19%

internal coach)			
Consultant/trainer	13%	10%	14%
Product owner	8%	7%	6%
Business analyst		5%	5%
IT staff	3%	3%	2%
C-Level	3%	3%	3%

Comparisons of participants' sectors are presented in Table 8. Despite small changes over the years, they are said to be stable for sector rates. Most of the participants in the year 2016 are again in the previous years as software (ISV) companies. In 2015, the share of those who participated in this sector decreased from 26% to 23% in 2016 with a slight decline.

Table 8. *Industries respondents worked in*

Responded	2014	2015	2016
Software (ISV)	25%	26%	23%
Financial services	12%	14%	14%
Professional services	11%	11%	12%
Healthcare	7%	6%	6%
Government	6%	6%	5%
Insurance	3%	4%	6%
Telecom	4%	4%	4%
Retail	4%	3%	3%
Manufacturing	4%	3%	4%
Media & Entertainment	3%	3%	3%
Internet services	3%	3%	3%

Transportation	3%	2%	4%
Consumer products	2%	2%	2%
Utilities	1%	2%	2%
Public services	1%	1%	1%
Other	11%	10%	11%

Not. Other is consist of Education, Consulting, Aerospace, Automotive, Hospitality, and Non-Profit.

3.2. Company Experience and Adoption

Participants' agile experience and adoption comparisons are given in Table 9 and the results of the year 2016 are shown in *Figure 33*. Over the years, the number of participants over five years of experience is higher than the other groups. In addition, the highest experienced participants in 2016 have broken record for all time with 44%. On the other hand, the total rate of participants who have exceeded three years experiences in 2016 is 72%. Thus, it turns out that almost two-thirds of the data used in these analyzes were generated by highly experienced people on the agile. Also, it is clear that personal experience of the participants increases throughout the years.

Table 9. *Personal experience with agile development practices*

Experience	2014	2015	2016
5+ years	28%	33%	44%
3 – 4 years	30%	30%	28%
1 – 2 years	32%	21%	7%
<1 year	10%	16%	11%

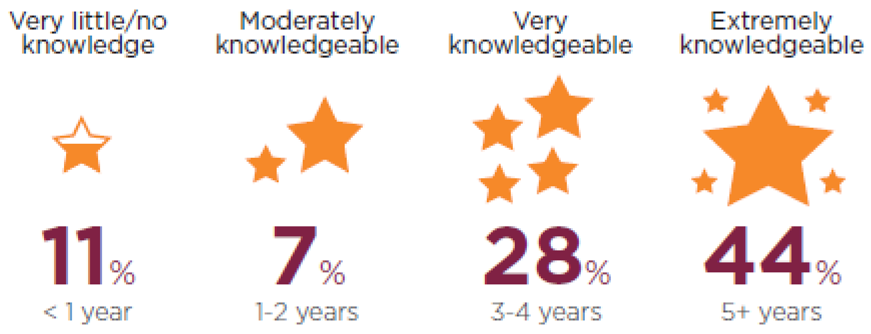


Figure 33. Agile development practices in 2016[73]

The comparison of the rates of agile treatment in the organizations where the participants are working is presented in Table 10. Agile practice is seen in the 94-95% band over the years. Thus, the analysis concludes that the subjects of the participants are experienced in agile.

Table 10. *Percentage of practicing agile based on count*

Year	Agile Practice
2014	94%
2015	95%
2016	94%

Figure 34 and Table 11 indicate that the comparison results of aging practice periods of organizations. According to the duration of the experience, there is no significant change in the rates over the years. On the other hand, the rate of organization with more than 5 years experiences in 2016 is 28%.

[73] VerisonOne. (2016). The 11th Annual State of Agile Report, p.7.

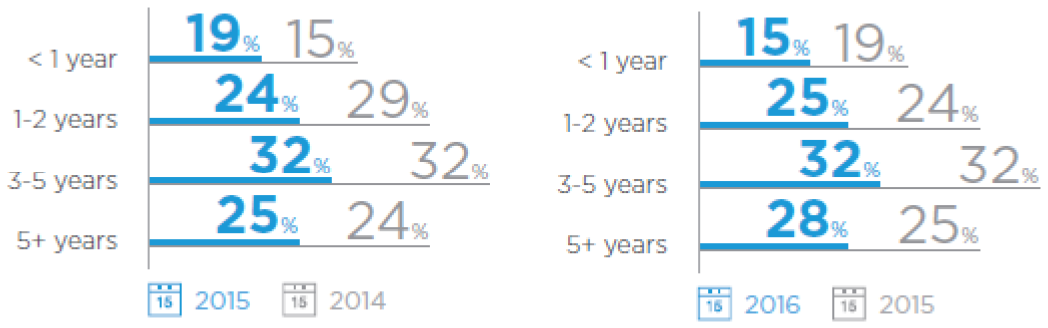


Figure 34. Comparison of agile experience[74,75]

In addition, the share of more than 3 years experienced organizations in total organization is 60%. Therefore, in addition to our previous findings, it is seen that both the organizations and the employees involved in the organization are experienced in agile.

Table 11. *Percentage of practicing agile based on the length of time*

Experience	2014	2015	2016
5+ years	24%	25%	28%
3 – 5 years	32%	32%	32%
1 – 2 years	29%	24%	25%
<1 year	15%	19%	15%

The proportion of teams using agile in 2016 is presented in *Figure 35* and in Table 12 with the comparison table according to years. In 2016, all teams in the organization use agile applications is 8%. In addition, 60% of the participants stated that they use less than half of the teams in their organizations. In general, the proportion of teams using agile does not increase even though personal experience

[74] VerisonOne. (2016). The 11th Annual State of Agile Report, p.7.
[75] VerisonOne. (2015). The 10th Annual State of Agile Report, p.6.

increases. This situation may be caused by struggles in change management and resistance to agile.

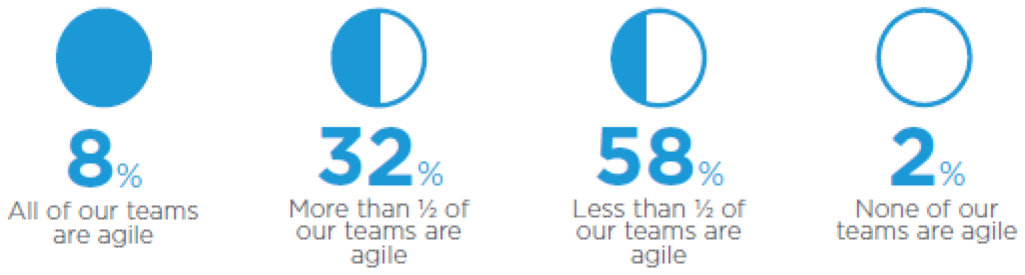


Figure 35. Percentage of teams using agile in 2016[76]

Table 12. *Percentage of teams using agile*

Teams	2014	2015	2016
All teams	9%	9%	8%
More than ½ of teams	36%	34%	32%
Less than ½ of teams	50%	53%	58%
None of teams	5%	4%	2%

Figure 36 indicates the results of the DevOps initiative by entrepreneurs in 2016 or the following period. Half of the respondents said that there is already a DevOps, and 21% said planning for the coming years. This is also an indication for increasing DevOps initiatives in upcoming years.

[76] VerisonOne. (2016). The 11th Annual State of Agile Report, p.7.

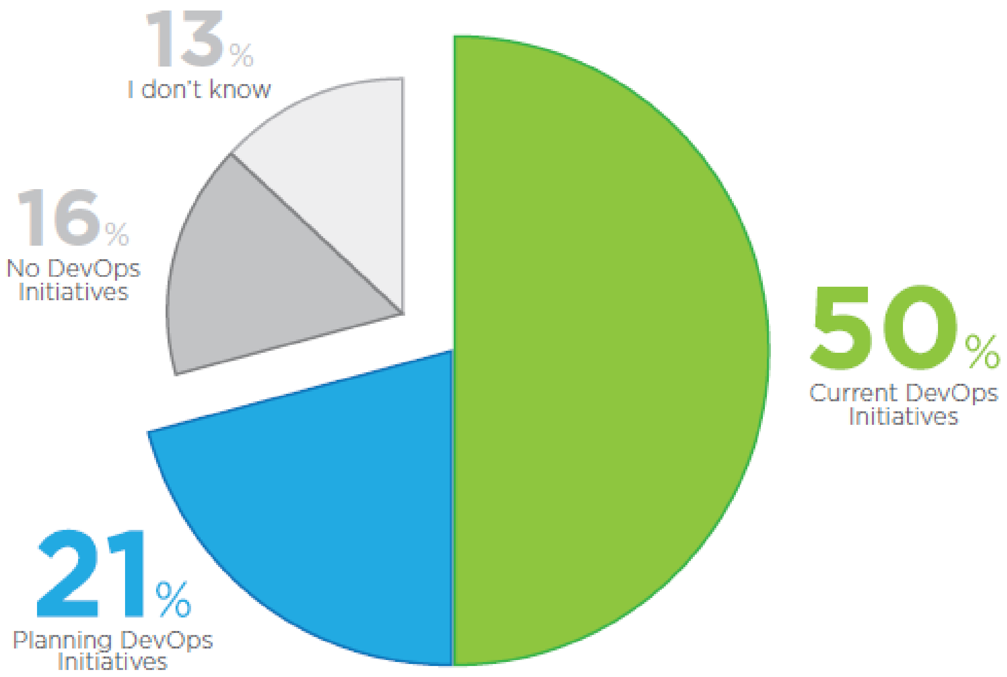


Figure 36. DevOps initiatives[77]

Table 13 gives the reasons for adopting agile comparisons over the years. Accelerate product delivery is the most important reason for increasing agility for agile adopting is hitting the bone. From 2014 to 2016, it increased from 59% to 69%. In the second and third place in the table, there is enhance ability to manage changing priorities and increase productivity. In addition, the improving project visibility increased three places to become the fourth most important reason for adopting agile in 2016.

Among the reason for adopting agile, 3 of top 5 reasons, accelerate product delivery, increase productivity, and enhance software delivery, are directly related to Continuous Delivery. These benefits are achieved by implementing Continuous Delivery on the software delivery life cycle.

[77] VerisonOne. (2016). The 11th Annual State of Agile Report, p.7.

Table 13. *Reasons for adopting agile*

Reasons	2014	2015	2016
Accelerate product delivery	59%	62%	69%
Enhance ability to manage changing priorities	56%	56%	61%
Increase productivity	53%	55%	53%
Improve project visibility	40%	40%	43%
Enhance software quality	46%	47%	43%
Improve business/IT alignment	40%	44%	42%
Reduce project risk	38%	40%	37%
Improve team morale	26%	29%	31%
Enhance delivery predictability	44%	44%	30%
Improve engineering discipline	25%	24%	21%
Better manage distributed teams	20%	21%	20%
Reduce project cost	23%	23%	18%
Increase software maintainability	22%	22%	18%

Organizations' agile maturity levels for the years 2014, 2015 and 2016 are presented in

Figure 37, *Figure 38*, and *Figure 39* respectively. In 2014, %94 percent of the respondents stated that there is some level of agile adoption in their software organizations. Also, %60 percent of organizations had passed the early adoption phase, including mid-level and above. %18 percent of the participants said that their organizations had mature adoption level.

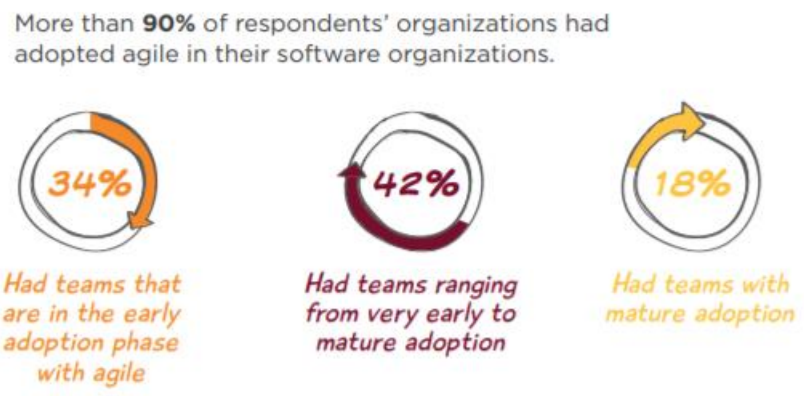


Figure 37. Agile maturity in 2014[78]

When returned to 2015, %17 percent of organizations had teams with mature agile. 44% percent of organizations had teams ranging from very early to mature adoption. 33% percent of organizations had teams that are in the early adoption phase with agile. Accordingly, 95% of the participants stated that there are agile practices in their organizations. Only 1% of the participants said that the result of agile implantation was unsuccessful. In comparison with 2014 report, percentages remained almost the same in 2015, while general adoption rate slightly increased.

[78] VerisonOne. (2014). The 9th Annual State of Agile Report, p.7.

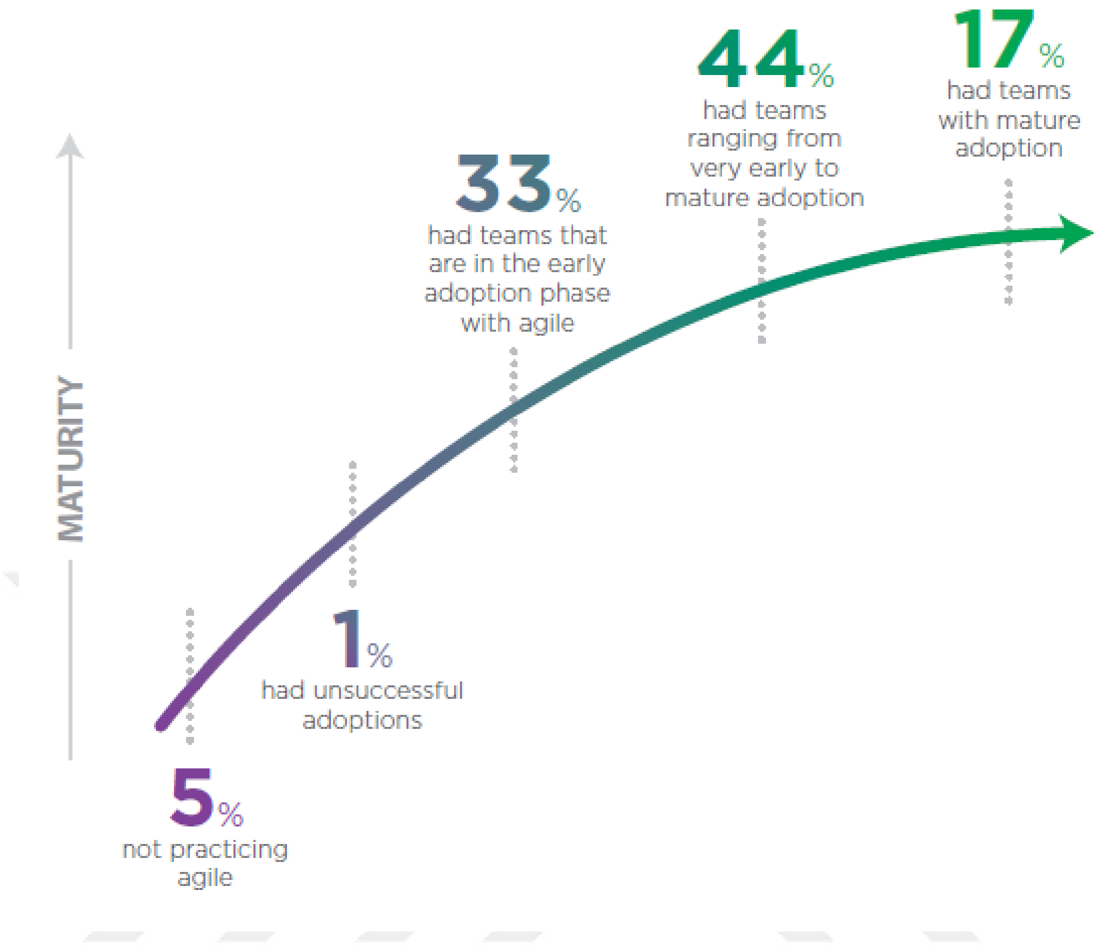


Figure 38. Agile maturity in 2015[79]

In 2016 report, adoption levels are questioned with a different terminology. %7 percent of organizations is enable greater adaptability to market with agile practices. In addition, %11 percent of organizations is high level of competency with agile practices across the organization. %60 percent of organizations use agile practices and techniques but still maturing. It means that the vast majority of organizations (79%) are still mature or below this level. 3% of the organizations do not have any initiative for agile. Comparing with the previous years, organizations are marching through complete maturity over the years.

[79] VerisonOne. (2015). The 10th Annual State of Agile Report, p.7.

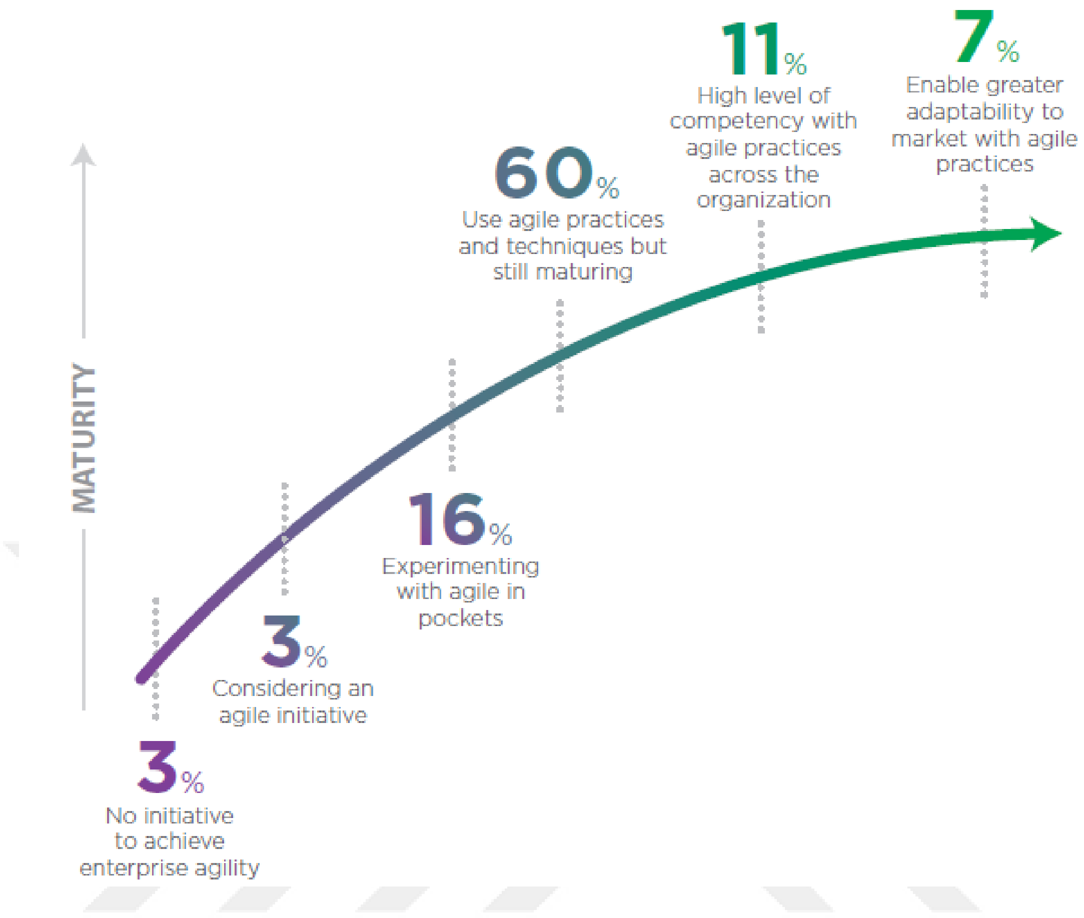


Figure 39. Agile maturity in 2016[80]

3.3. Benefits of Agile

The benefits of agile implements in 2016, 2015, and 2014 are presented in *Figure 40*, *Figure 41*, and *Figure 42*, respectively. The top three benefits of adopting agile have remained steady for the past five years: manage changing priorities (88%), team productivity (83%), and project visibility (83%). Even if there are many different benefits, some of them are quite in the foreground. These are: ability to manage changing priorities (88%), project visibility (83%), increased team productivity (83%), delivery speed/time to market (81%) and team morale (81%).

[80] VerisonOne. (2016). The 11th Annual State of Agile Report, p.8.

According to these figures showing benefits of adopting agile over the years, each year, continuous delivery related benefits sit on the top of the list: increased team productivity, delivery speed/time to market, enhanced software quality and business/IT alignment are stated to be actual improvements after implementing agile and continuous delivery. These improvements prove that implementing continuous delivery made positive effects during the agile adoption.

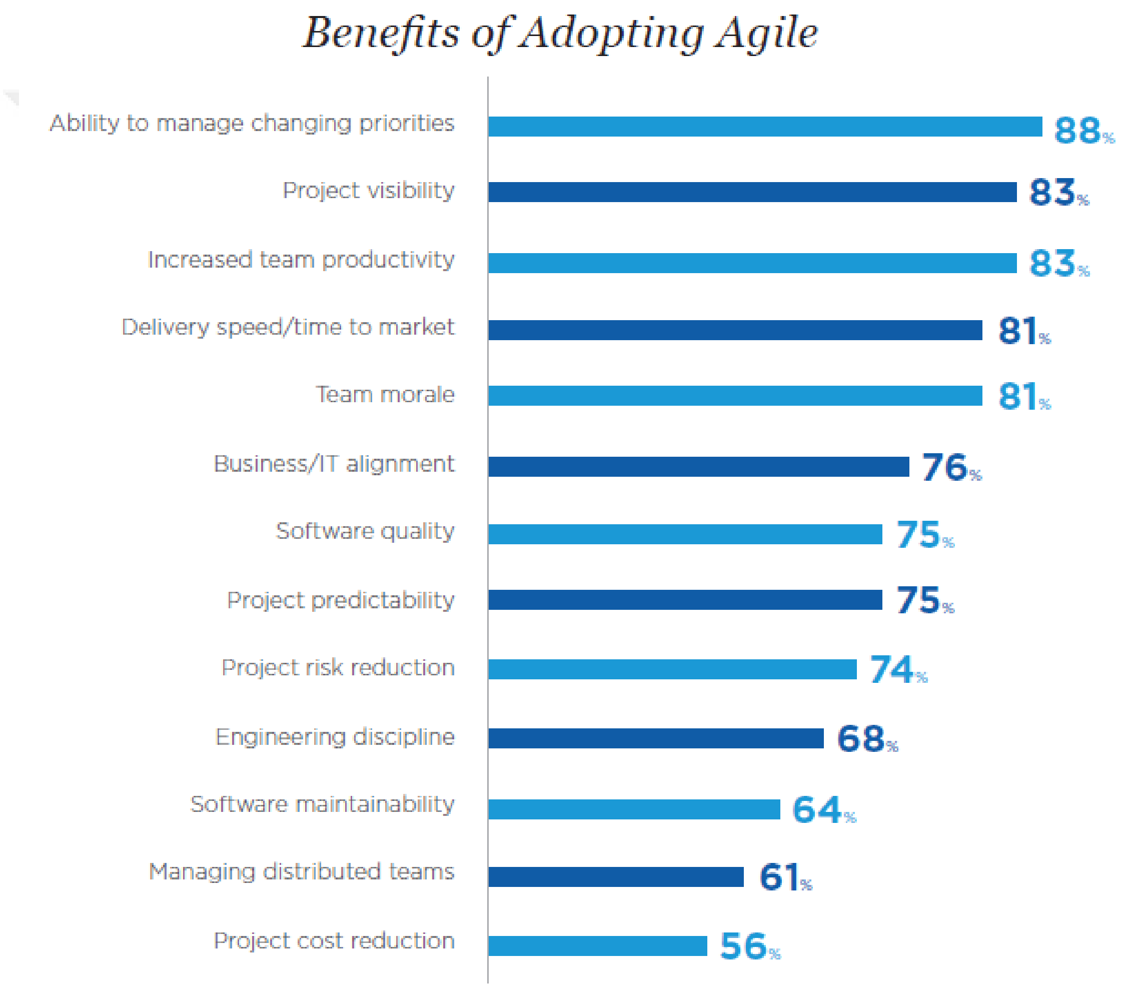
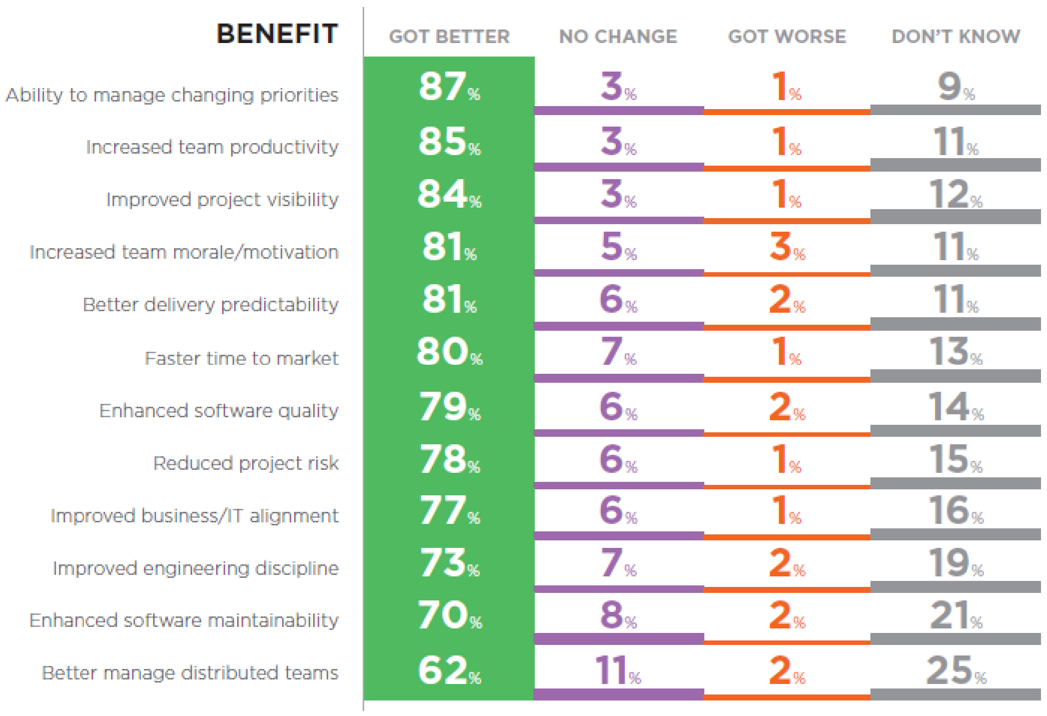


Figure 40. Benefits of implementing agile in 2016[81]

[81] VerisonOne. (2016). The 11th Annual State of Agile Report, p.8.



BENEFIT	GOT BETTER	NO CHANGE	GOT WORSE	DON'T KNOW
Ability to manage changing priorities	87%	3%	1%	9%
Increased team productivity	85%	3%	1%	11%
Improved project visibility	84%	3%	1%	12%
Increased team morale/motivation	81%	5%	3%	11%
Better delivery predictability	81%	6%	2%	11%
Faster time to market	80%	7%	1%	13%
Enhanced software quality	79%	6%	2%	14%
Reduced project risk	78%	6%	1%	15%
Improved business/IT alignment	77%	6%	1%	16%
Improved engineering discipline	73%	7%	2%	19%
Enhanced software maintainability	70%	8%	2%	21%
Better manage distributed teams	62%	11%	2%	25%

Figure 41. Benefits of implementing agile in 2015[82]

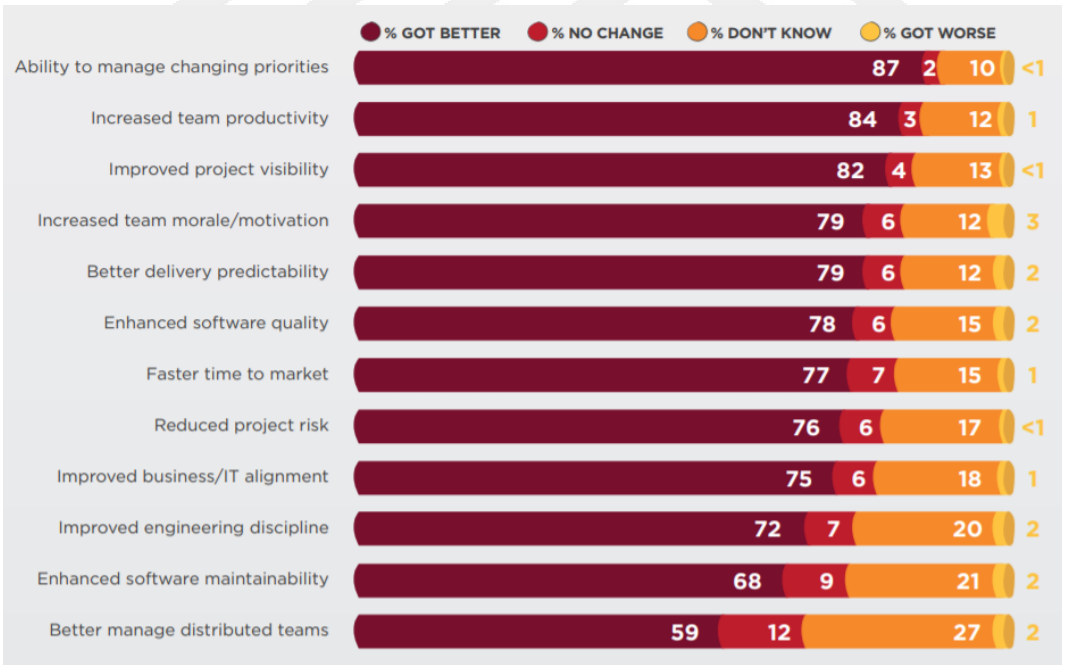


Figure 42. Benefits of implementing agile in 2014[83]

[82] VerisonOne. (2015). The 10th Annual State of Agile Report, p.8.
[83] VerisonOne. (2014). The 10th Annual State of Agile Report, p.8.

3.4. Agile Methods and Practices

The comparisons of the agile methodologies used for years and the results of the year 2016 are given in Table 14 and *Figure 43*, respectively. The main picture has not undergone many changes over the years. Scrum and Scrum / XP hybrid methodologies were the most preferred agile methods with 68% between the years 2014-2016.

Table 14. *Comparison of agile methodology used*

Methodologies	2014	2015	2016
Scrum	56%	58%	58%
Scrum/XP hybrid	10%	10%	10%
Custom hybrid	8%	8%	8%
Scrumban	6%	7%	8%
Kanban	5%	5%	5%
Iterative development	4%	3%	2%
Lean development	2%	2%	1%
Agile modeling	1%	1%	
Lean startup			<1%
Feature-driven development (FDD)	1%	1%	<1%
DSDM / Atern	<1%	1%	<1%
XP	<1%	1%	<1%
Agile unified process (AgileUP)	<1%	<1%	<1%
Other	2%	3%	5%
I do not know	3%	2%	2%

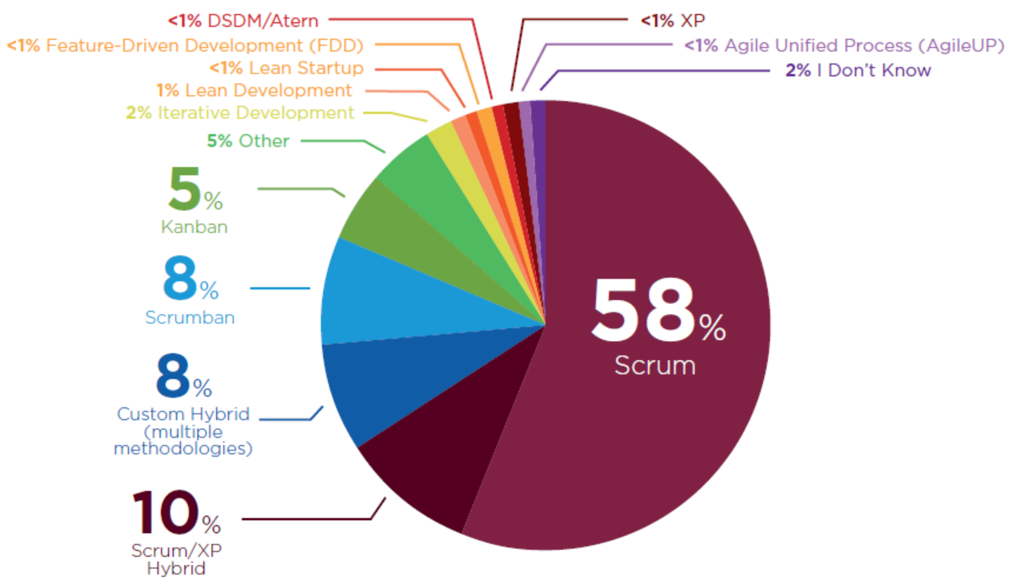


Figure 43. Agile methodology used in 2016[84]

Table 15 contains merged data from agile techniques and related engineering techniques used compared to the years. In 2016, the iteration planning technique showed a significant increase compared to other years and rose to the first place. The daily standup technique, which ranks first in 2014 and 2015, falls to second place in 2016. On the other hand, prioritized backlogs are the second most preferred technique in 2014 and 2015, but not in 2016. In addition, the main techniques that have significantly increased from 2015 to 2016 are: iteration review 54% to 81%, Kanban to 39%, and iteration planning from 69% to 90%.

Within the agile techniques shown in Table 15 there are some techniques that are implemented while implementing continuous delivery. Continuous integration and continuous deployment, which increased over the years, are continuous delivery's predecessor and successor and could be considered as the same. Unit testing and automated acceptance testing are about test automation and are very important components for a continuous delivery infrastructure, which also made significant

[84] VerisonOne. (2016). The 11th Annual State of Agile Report, p.10.

progress over time. These findings showed that majority of the participants used these CD-related techniques and a successful agile implementation is only achievable when continuous delivery is implemented. Also, importance of continuous delivery while implementing a successful agile software development infrastructure is increasing constantly as the time goes by.

Table 15. *Agile techniques and engineering techniques employed*

Techniques	2014	2015	2016
Iteration planning	71%	69%	90%
Daily standup	80%	83%	88%
Prioritized backlogs	79%	82%	
Retrospectives	69%	74%	83%
Iteration reviews	53%	54%	81%
Unit testing	65%	63%	74%
Short iterations	79%	79%	71%
Release planning	65%	63%	66%
Team-based estimation	56%	59%	62%
Taskboard	53%	55%	
Continuous integration	50%	50%	61%
Coding standards	43%	44%	56%
Dedicated product owner	48%	49%	55%
Single team (integrated dev & testing)	46%	45%	54%
Refactoring	36%	37%	52%
Frequent releases			50%
Kanban	34%	39%	50%
Open work area	38%	38%	45%
Test-driven development (TDD)	34%	33%	40%

Product roadmapping			38%
Automated acceptance testing	24%	28%	36%
Continuous deployment	24%	27%	35%
Story mapping	29%	30%	35%
Pair programming	22%	24%	32%
Collective code ownership	27%	25%	31%
Agile portfolio planning			25%
Agile/lean UX			22%
Behavior-driven development (BDD)	9%	10%	16%

3.5. Agile Success and Metrics

Participants indicated in 2016 that their organizations had achieved 98% success in the projects they were using. On the other hand, many organizations are aware of the difficulties of measuring success even if they express the success of mass applications. At the beginning of these difficulties are organizational culture at odds with agile values (63%), lack of skills or experience with agile methods (47%) and lack of management support (45%).

The comparison results of reasons why the agile projects fail is presented in Table 16. Participants' answers include a significant difference between 2014-2015 and 2016. In 2016, the level of "organizational culture at odds with core agile values", which is at a lower level in 2014 and 2015, has risen to 63%.

Results of the data in Table 16 give the opportunity to make comments about the potentials risks in agile software development. According to the participants, top causes for failed agile projects are related to company culture, company structure and experience. Thus, company and management itself play a great role in agile. Lack of support and agile experience may lead project to failure. Even though agile software development experience increase over the years, it also requires a lot experience and effort in change management to succeed in projects.

Table 16. *Leading causes of failed agile projects*

Causes	2014	2015	2016
Company philosophy or culture at odds with core agile values	42%	46%	63%
Lack of experience with agile methods	44%	41%	47%
Lack of management support	38%	38%	45%
General organization resistance to change			43%
Lack of business/customer/product owner			41%
Pervasiveness of traditional development			34%
Lack of support for cultural transition	37%	38%	
Inconsistent agile practices and process		38%	31%
Fragmented tooling, data, and measurements			20%
Regulatory compliance and governance			15%
External pressure to follow traditional waterfall processes	37%	36%	
Ineffective management collaboration		34%	
A broader organizational or communications problem	33%	30%	
Unwillingness of team to follow agile	33%	30%	
Inability to continuously prioritize work		28%	
Insufficient training	30%	27%	34%
Ineffective collaboration		25%	19%
Don’t know	6%	5%	2%

Comparisons of organizations' agile adoption barriers between 2014 and 2015 are presented in Table 17. As in previous periods, participants view the organizational culture and the general resistance of change as the greatest obstacle to further agile adoption. Concerns about organizational culture increased from 44% in 2014 to 55% in 2015, and concerns about a general resistance to change increased from 34% in 2014 to 42% in 2015.

Table 17. *Barriers to further agile adoption*

Barrier	2014	2015
Ability to change organizational culture	44%	55%
General organizational resistance to change	34%	42%
Pre-existing rigid/waterfall framework	32%	40%
Not enough personnel with the necessary agile experience	35%	39%
Management support	29%	38%
Business/user/ customer availability	23%	28%
Concerns about a loss of management control	22%	27%
Management concerns about lack of upfront planning	24%	25%
Confidence in ability to scale agile methodologies	15%	18%
Concerns about the ability to scale agile	14%	18%
No barriers	16%	17%
Perceived time and cost to make the transition	12%	15%
Development team support	13%	14%
Regulatory compliance	11%	13%

The comparative agile initiatives for the years are given in Table 18 and the results for 2016 are given in *Figure 44*. The business value, which is seen as a success criterion, ranks fourth in 2014 and 2015, and ranks second in 2016. On-time delivery appears to have fallen by 5% in 2016, despite being ranked first in all time. On-time delivery, business value, customer satisfaction and product quality remain in the top four measures of agile initiative's success as they have in the past few years.

Among the participants' answers about measures of an agile initiative's success, as in Table 18, 4 of top 6 measures are also measures for a successful continuous delivery implementation: On-time delivery, business value, product quality, and productivity. It means that an agile initiative also requires a continuous delivery initiative in place.

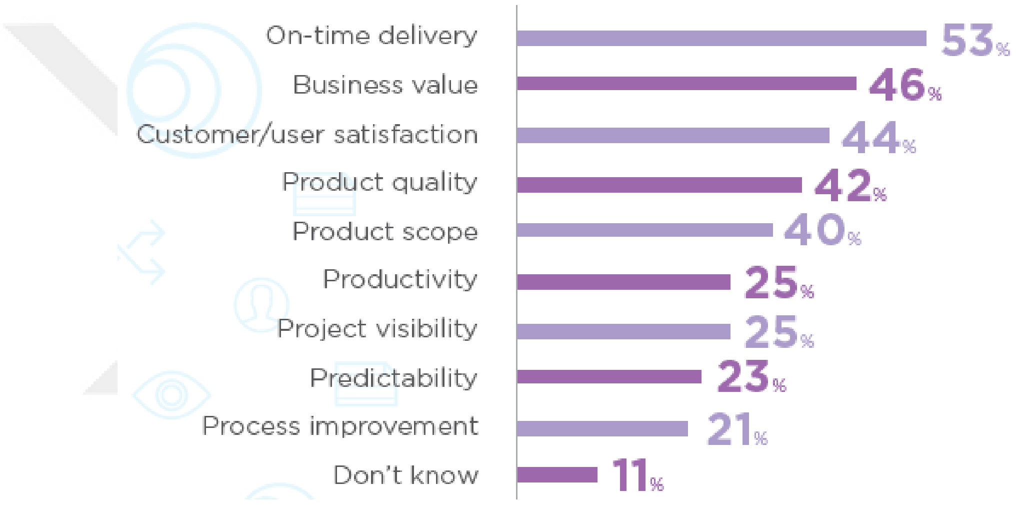


Figure 44. Measures of an agile initiative's success in 2016[85]

Table 18. *Measures of an agile initiative's success*

Initiative	2014	2015	2016
On-time delivery	58%	58%	53%
Business value	44%	46%	46%
Customer/user satisfaction	44%	46%	44%
Product quality	48%	48%	42%
Product scope	39%	36%	40%

[85] VerisonOne. (2016). The 11th Annual State of Agile Report, p.13.

Productivity	29%	31%	25%
Project visibility	30%	30%	25%
Predictability	25%	26%	23%
Process improvement	23%	24%	21%
Don’t know	11%	11%	11%

The comparison of the measures of an agile project’s success between 2014 and 2016 and the results of the year 2016 are given in Table 19 and *Figure 45*, respectively. Between 2014 and 2016 Velocity emerged as the first measure. As the years progress, its popularity seems to increase. Business value, despite being ranked second with 46% in the agile initiative's success ranking in 2016, could be ranked eleventh (23%) among the most popular measurement methods. On the other hand, there is no observable change in the values of the other measurement methods in the first fifteen.

Velocity is the measure that is affected the most by a successful continuous delivery implementation. It is also stated the most used measure for an agile project’s daily success. Thus, continuous delivery and automation is strictly related to an agile project’s success.

Table 19. *Measures of an agile project’s success*

Measured	2014	2015	2016
Velocity	59%	57%	67%
Iteration burndown	51%	51%	51%
Release burndown	39%	41%	38%
Planned vs. actual stories per iteration	35%	37%	37%
Burn-up chart	29%	34%	34%

Work-in-Process (WIP)	27%	30%	32%
Defects into production	26%	28%	30%
Customer/user satisfaction	28%	29%	28%
Planned vs. actual release dates	29%	29%	26%
Cycle time	18%	19%	23%
Defects over time	22%	28%	23%
Business value delivered	19%	21%	23%
Budget vs. actual cost	22%	23%	22%
Defect resolution	21%	20%	20%
Cumulative flow chart	12%	15%	19%

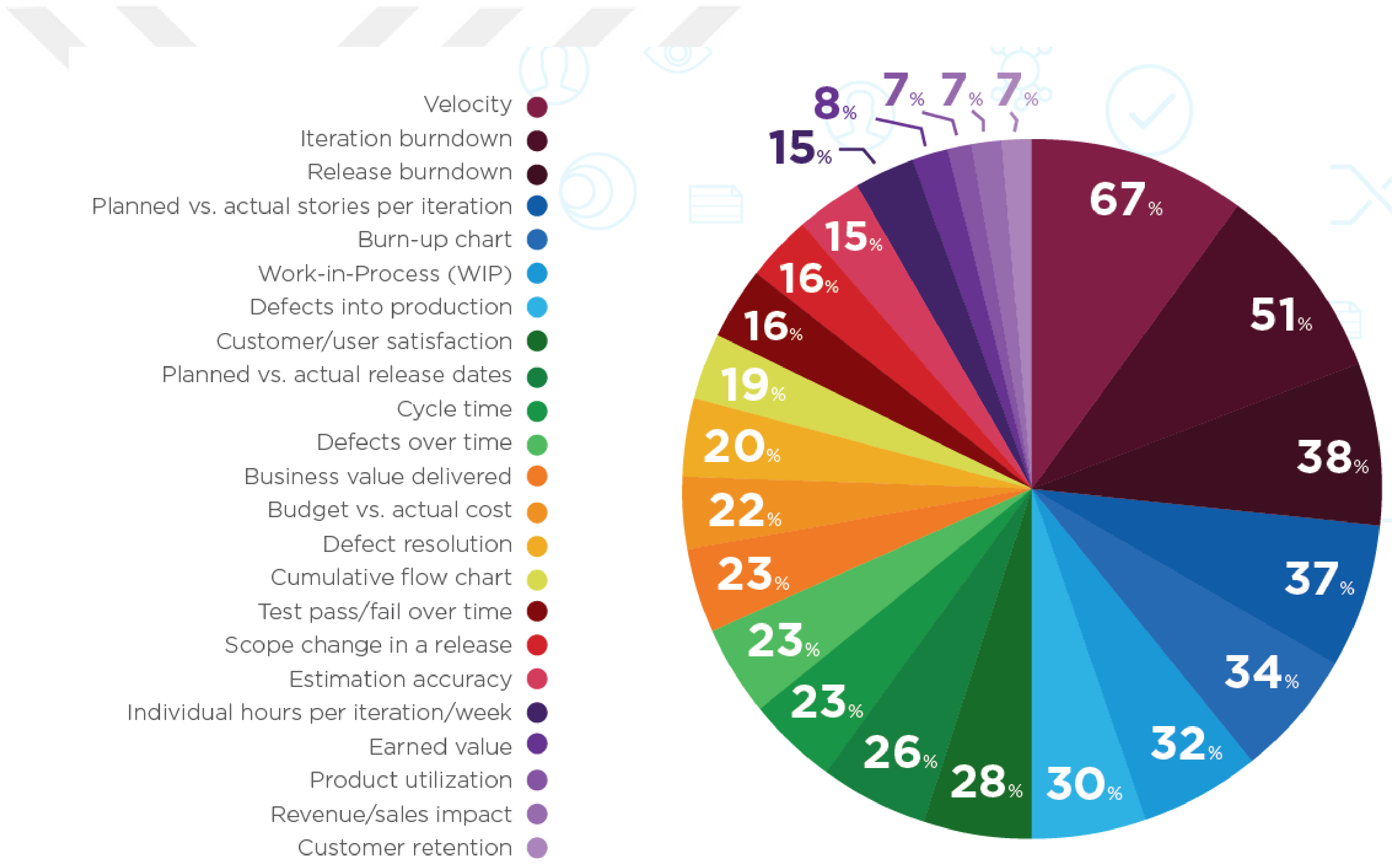


Figure 45. Agile initiative's success measured in 2016[86]

[86] VerisonOne. (2016). The 11th Annual State of Agile Report, p.13.

3.6. Scaling Agile

The scaling methods and approaches graph of 2016 is presented in *Figure 46*. SAFE and Scrum / Scrum of Scrums are the most popular scaling methods in 2016 according to the participants, with 28% and 27% respectively.

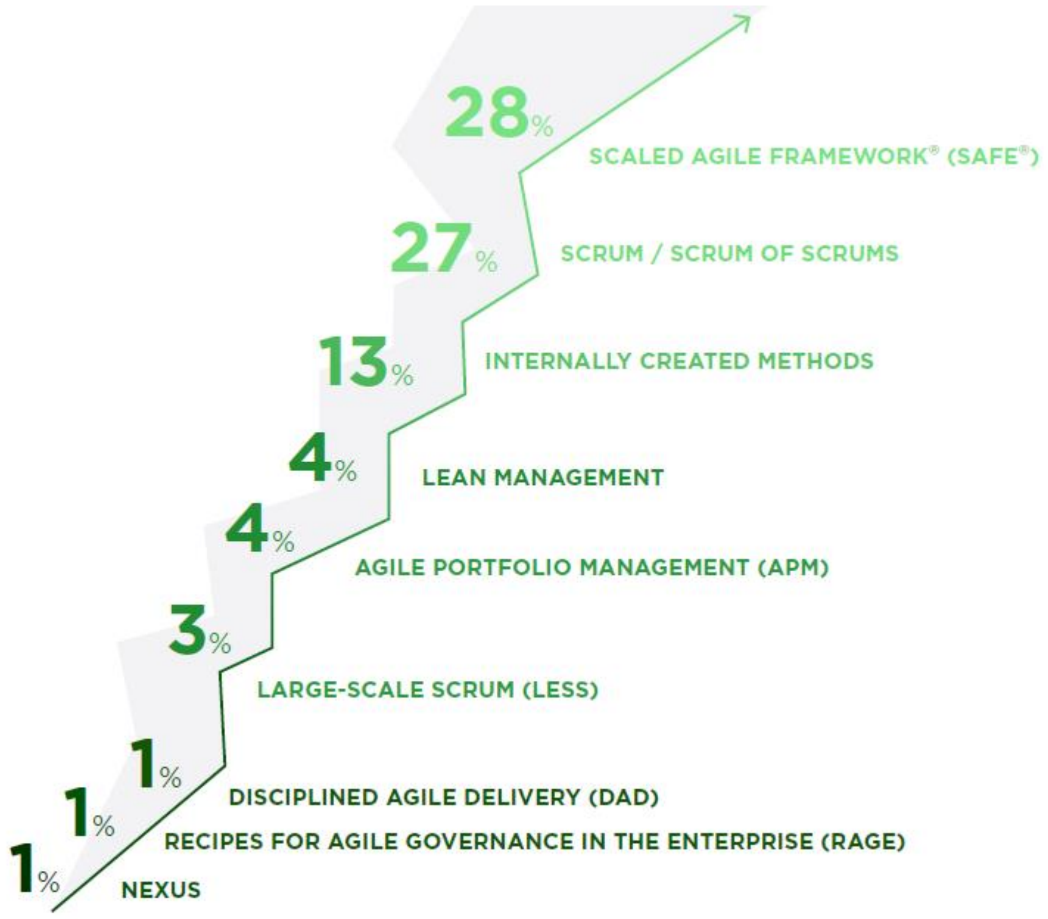


Figure 46. Scaling methods and approaches in 2016[87]

The comparison results according to years are presented in Table 20. It should be noted that, in 2016, participants were not able to make multiple selections. The results are quite intriguing, and the scaling method of Scrums / Scrums of Scrums seems dominate the list in 2014 and 2015, but is also one of the most popular methods

[87] VerisonOne. (2016). The 11th Annual State of Agile Report, p.14.

in 2016. On the other hand, SAFE is stable at 28% over the years. In addition, it can be said that internally created method is popular when it comes to scaling agile. Enterprise agile and scrum methods have not been found in the reports of the year 2016. On the contrary, despite being a tableland in 2014 and 2015, Nexus found itself in the list, despite having a small share in 2016.

Table 20. *Scaling methods and approaches*

Methods & Approaches	2014	2015	2016
Scrums / Scrums of Scrums	69%	72%	27%
Scaled agile framework (SAFE)	19%	27%	28%
Internally created methods	25%	23%	13%
Lean management	18%	17%	4%
Agile portfolio management	9%	9%	4%
Large-scale scrum (LeSS)	3%	6%	3%
Disciplined agile delivery (DAD)	4%	4%	1%
Recipes for agile governance in the enterprise (RAGE)	1%	1%	1%
Nexus			1%
Enterprise agile	10%	9%	
Enterprise Scrum	9%	9%	

The comparison of top five tips for success with scaling agile is presented in Table 21. Executive sponsorship (48%), consistent process and practices (41%), implementation of a common tool across teams (36%), and agile consultants or trainers (36%) continue to be cited in the top five tips for successfully scaling agile for the past few years and likely points to the long-term importance of self-sufficiency when scaling agility. The most mentioned tip in 2016 was internal agile coaches with

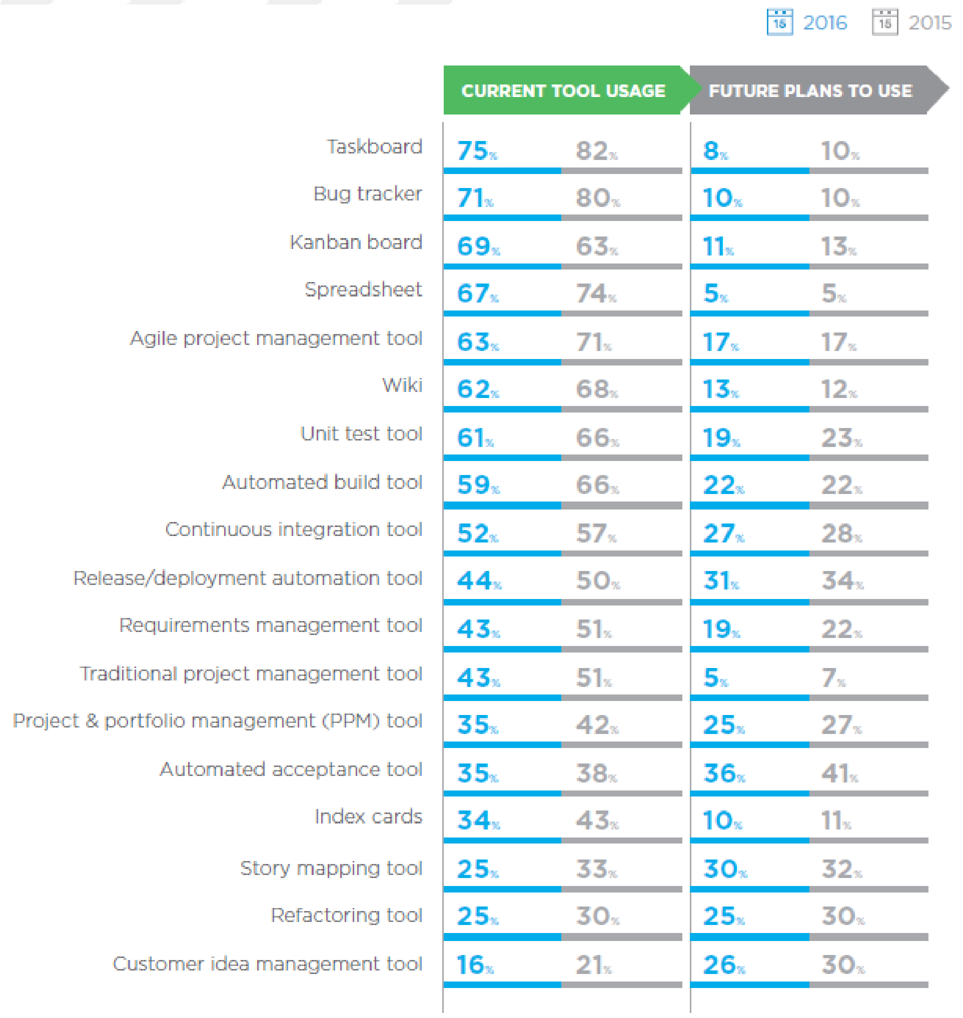
52%. In addition, some of other important factors are externally attended classes or workshops, company-provided training program, online training, and webinars.

Table 21. *Comparison of top five tips for success with scaling agile*

Tips	2014	2015	2016
Internal agile coaches	31%	55%	52%
Executive sponsorship	40%	37%	48%
Consistent process and practices	42%	43%	41%
Implementation of a common tools across teams	39%	40%	36%
Agile consultants of trainers	35%	40%	36%

3.7. Agile Project Management Tools

The overall tool comparisons for the years 2015 and 2016 are given in *Figure 47*. The taskboard is predominantly in the first place. However, when the year 2015 is passed to 2016, a meaningful decrease is observed (-7%). The use of taskboard in future plans is targeted at 8%. On the other hand, the usage rates of all tools except Kanban are decreasing. The number of Kanban increased by 6% from 2015 to 2016. Results in general also show that automation tools are a very important elements of the agile tool palette.



🟦 2016 ⬜ 2015

	CURRENT TOOL USAGE		FUTURE PLANS TO USE	
	2016	2015	2016	2015
Taskboard	75%	82%	8%	10%
Bug tracker	71%	80%	10%	10%
Kanban board	69%	63%	11%	13%
Spreadsheet	67%	74%	5%	5%
Agile project management tool	63%	71%	17%	17%
Wiki	62%	68%	13%	12%
Unit test tool	61%	66%	19%	23%
Automated build tool	59%	66%	22%	22%
Continuous integration tool	52%	57%	27%	28%
Release/deployment automation tool	44%	50%	31%	34%
Requirements management tool	43%	51%	19%	22%
Traditional project management tool	43%	51%	5%	7%
Project & portfolio management (PPM) tool	35%	42%	25%	27%
Automated acceptance tool	35%	38%	36%	41%
Index cards	34%	43%	10%	11%
Story mapping tool	25%	33%	30%	32%
Refactoring tool	25%	30%	25%	30%
Customer idea management tool	16%	21%	26%	30%

Figure 47. General tool usage in 2016[88]

[88] VerisonOne. (2016). The 11th Annual State of Agile Report, p.15.

The agile management tool list used in 2016 is shown in *Figure 48* and the comparison of the ten most used tools is presented in Table 22. Although there are many options among the management tools, there are usually a few tools in the foreground. These can be listed as Atlassian / JIRA, Microsoft Excel, Microsoft TFS, Microsoft Project, and VersionOne. In 2014 and 2015, the most commonly used management tool was Microsoft Excel at 60%, but in 2016 it fell to 46%, dropping to second place. On the other hand, the most preferred tool in 2016 was Atlassian / JIRA with 53%. In addition, there has been a considerable increase over time in the Agile Central.

Results show that usage of old and non-agile oriented tools like Excel and Microsoft Project is decreasing. That means companies are making investment in tools to be more agile.

Table 22. *Comparison of use of top ten agile management tools*

Tips	2014	2015	2016
Atlassian/JIRA	45%	51%	53%
Microsoft Excel	68%	60%	46%
Microsoft TFS	24%	24%	22%
Microsoft Project	46%	33%	20%
VersionOne	33%	28%	20%
Google Docs	24%	18%	14%
CA Agile Central	3%	3%	13%
HP Quality Center/ALM	23%	18%	12%
In-house/home-grown	19%	14%	10%
Bugzilla	19%	10%	5%

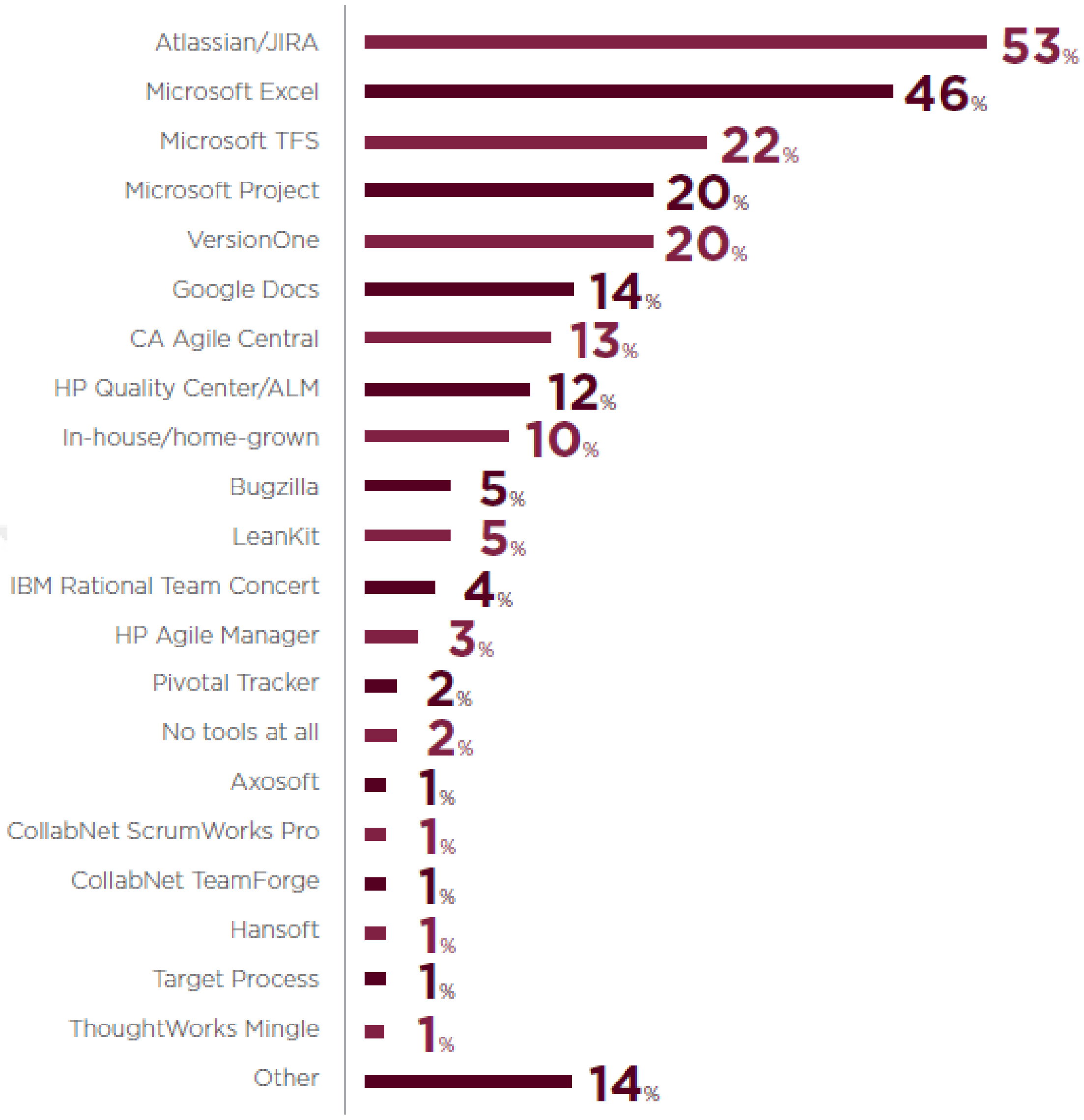


Figure 48. Use of agile management tools in 2016[89]

Based on past or current experience of participants, management tool recommendations are presented in *Figure 49*. According to the result, VersionOne, (87%) Atlassian/JIRA (81%) and CA Agile Central (76%) have the top three management tool.

[89] VerisonOne. (2016). The 11th Annual State of Agile Report, p.16.

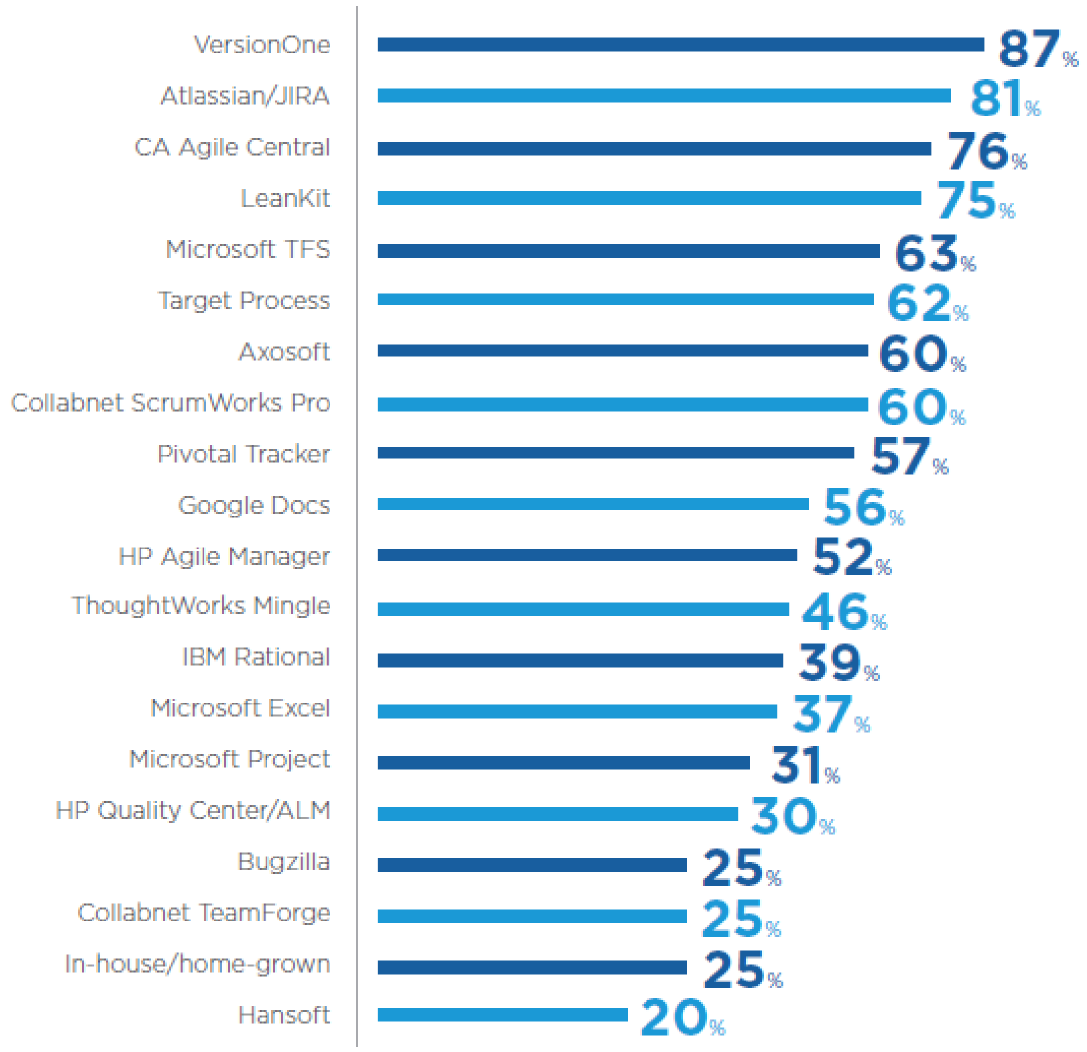


Figure 49. Recommended agile project management tools in 2016[90]

[90] VerisonOne. (2016). The 11th Annual State of Agile Report, p.17.

4. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

Automation in software development sector has been too long developed and too long neglected. As technology integrated in every business model, continuous integration and continuous deployment became heart of the computer science and software development. However, it is not limited; business models and mechanism of work changed faster than people think. It is clear to see that agile and automation culture, especially in project management, plays outstanding role on project success in a positive way. There are outstanding, limitless subunits that affects development, project success and business models so that management methodologies. With the rise of digitalization, management models have drastically changed and they made huge impact on every sector.

It is not totally true to consider only locally because methodologies are in a global scale, life-long processes. In this thesis, agile management methodologies in scope of continuous integration and continuous delivery were analyzed and found out that managing projects by agile methodology with implemented continuous integration and continuous delivery have positive effects on project success. According to the survey and data analysis, advantages of the continuous delivery, continuous integration and agile management methodology are proven:

- Continuous Delivery has a great meaning for business models in terms of technology and project success.
- Using Continuous Delivery methodologies and agile management style makes software and projects more successful, easy to use and integrated.
- In terms of computer science, software development, business models and project management styles Continuous Delivery methodology and agile

management style generate more successful and faster results than other methodologies.

- When Continuous Delivery methodologies are used, software development and management of business processes have better quality than others.

In addition, as a result of this research, emerging technology leads to development of automation systems. It also helps to improve the business models which consist of glamorous features. Thanks to these features, companies have to compete and survive with these challenges. As a result, automation systems especially continuous integration and continuous delivery are must-to-haves for agile management style.

As an attachment, the thesis describes the analysis of the agile development data of countries collected from a survey made by “versionone.com”.

Moreover, findings proved that there is strong relationship between agile management style and project success. Detailed comparison of agile project management style with others is given in the research. Competitive rules have changed and these changes are leaded by agile management techniques. Benefits are obtained from implementing Agile are explained according to the data analysis. However, there are also failed agile projects. This thesis by data analysis also explains the reasons of the failures.

As a result, when data is analyzed, results proved that agile methodologies highlight the importance of the continuous integration and continuous delivery. Agile management style and their effects on business models and project success are completely positive.

In today’s world, reality and virtuality are connected. The findings present possible and current agile management, continuous integration and continuous delivery effects on business models. In this manner, with these positive effects of agile and technologies, business models are being improved.

REFERENCES

Abdoulaye, P. (2017, December 05). *Digital Transformation: How DevOps Could Have Revamped Toys R Us.* Retrieved 2018, from devops.com: https://devops.com/digital-transformation-devops-revamped-toys-r-us/

Aiello, B., & Sachs, L. (2016). *Drive adoption and overcome resistance to change.* Retrieved 2018, from IBM: www.ibm.com/developerworks/library/d-drive-devops-adoption/index.html

Anastasov, M. (2017). *What's the Difference Between Continuous Integration, Continuous Deployment and Continuous Delivery?* Retrieved 2018, from Semaphore CI: https://semaphoreci.com/blog/2017/07/27/what-is-the-difference-between-continuous-integration-continuous-deployment-and-continuous-delivery.html

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). *Manifesto for agile software development.* Retrieved 2018, from http://www.agilemanifesto.org/principles.html: http://www.agilemanifesto.org/principles.html

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. *Computer, 36*(6), 57-66.

Cho, J. J. (2010). An exploratory study on issues and challenges of agile software development with scrum. *All Graduate theses and dissertations,* 599.

ClearPoint. (2016). *Digital Continuous Delivery*. Retrieved from ClearPoint: http://www.clearpoint.co.nz/blog/digital-continuous-delivery/

Cohn, M. (2012). *Agile Succeeds Three Times More Often Than Waterfall.* Retrieved 2018, from Mountain Goat Software: http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall

Craig, J. (2017). *DevOps/Continuous Delivery Tooling, Launchpad for the Digital Enterprise.* EMA™, Application Management. EMA™, IT & DATA MANAGEMENT RESEARCH, INDUSTRY ANALYSIS & CONSULTING.

Deloitte Touche Tohmatsu Limited. (2018). What is Digital Economy. *Technology Articles*.

Eggers, W. D., & Bellman, J. (2015). *The journey to government’s digital transformation.* Deloitte Touche Tohmatsu Limited. Deloitte University Press.

Elmquist, N. (n.d.). *Agile 101: A Brief Intro to Agile Development*. Retrieved 2018, from Digital Crafts: https://www.digitalcrafts.com/blog/agile-101-brief-intro-agile-development

Fowler, M. (2005, December 10). *The new methodology*. Retrieved from http://martinfowler.com/articles/newMethodology.html

Fowler, M., & Foemmel, M. (2006). *Continuous integration.* Retrieved 2018, from Thought-Works: http://www. thoughtworks.com/Continuous Integration.pdf

Fruhling, A., & De Vreede, G. J. (2006). Field experiences with extreme programming: Developing an emergency response system. *Journal of Management Information Systems, 22*(4), 39-68.

Hilbert, M. (2017). *Bringing Continuous Delivery to the Database*. Retrieved from Database Zone: https://dzone.com/articles/bringing-continuous-delivery-to-the-database-1

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and Agile methods. *Proceedings of the annual International conference on Computer software and applications.*(1), 520-525.

Ika, L. (2009). Project success as a topic in project management journals. *Project Management Journal, 40*(4), 6-19.

Kuncoro, A. (2016, October 12). Oracle Database Consultant. *Solutions for IT Organizations on The Journey to The Digital Enterprise*.

Leffingwell, D. (2007). *Scaling software agility: best practices for large enterprises.* NJ: Pearson Education.

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series).* Boston: Pearson Education, LTD.

Maxos. (2017). *Continuous Delivery*. Retrieved 2018, from Continuous Agile: http://www.continuousagile.com/unblock/cd_costs_benefits.html

Microsoft. (2012). *Testing in the Software Lifecycle*. Retrieved 2018, from MSDN: https://msdn.microsoft.com/en-us/library/jj159342.aspx

Mourot, C. (2014). *A Story of Continuous Integration.* Retrieved 2018, from Salesforce Engineering: https://www.slideshare.net/salesforceeng/a-story-of-continuous-integration

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. *In Enabling Technologies: Infrastructure for Collaborative Enterprises*, 308-313.

Paulk, M., Weber, C., Curtis, B., & Chrissis, M. (1995). *The Capability Maturity Model:Guidelines for Improving the Software Process.* Indianapolis: Addison-Wesley.

Pittet, S. (2017). *Continuous integration vs. continuous delivery vs. continuous deployment*. Retrieved 2018, from Atlassian: https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

Radwan, M. (2016, 12 3). *Introduction to software testing*. Retrieved 01 01, 2018, from Automation Planet:

http://mohamedradwan.com/2017/07/01/introduction-to-software-testing/1-waterfall-vs-agile-4/

Reed, J. P. (2018). *The business case for continuous delivery*. Retrieved 2018, from Atlassian: https://www.atlassian.com/continuous-delivery/business-case-for-continuous-delivery

Rosenau, M. D. (1984). *Project management for engineers.* New York, NY: Lifetime Learning.

Schwaber, K. (1996). Controlled chaos: Living on the edge. *American Programmer, 9*, 10-16.

Singleton, A. (n.d.). Retrieved 2018, from www.continuousagile.com: http://www.continuousagile.com/unblock/cd_costs_benefits.html

Smith, D. (2015, February 24). *Agile Feedback Loops*. Retrieved 2018, from Medium: https://medium.com/rootpath/agile-feedback-loops-by-danny-smith-64f6f14894bc

Sommer, D. (2004). Project failure—12 mistakes to avoid. *Paper presented at the PMI Global Congress.* North America, Atlanta, GA.

The Standish Group. (2008). *The trends in IT value.* Yarmouth, MA: The Standish Group International.

The Standish Group. (2011). *CHAOS.* West Yarmouth, MA: The Standish Group International.

VerisonOne. (2014). *The 9th Annual State of Agile Report.* VerisonOne.

VerisonOne. (2015). *The 10th Annual State of Agile Report.* VerisonOne.

VerisonOne. (2016). *The 11th Annual State of Agile Report.* VerisonOne.

Vöst, S., & Wagner, S. (2016). Towards continuous integration and continuous delivery in the automotive industry. *arXiv preprint*, 1-5.

Watson, P. (2009). A methodology for attaining and assessing project success for rehabilitation projects. *Journal of Building Appraisal*, *4*(3), 181.