BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF BUSINESS AND MANAGEMENT
FAKULTA PODNIKATELSKÁ

INSTITUTE OF INFORMATICS
ÚSTAV INFORMATIKY

USAGE OF AGILE METHODOLOGY IN SOFTWARE DEVELOPMENT MANAGEMENT
VYUŽITÍ AGILNÍ METODIKY PŘI ŘÍZENÍ VÝVOJE SOFTWARU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Zuzana Mazáková**
AUTOR PRÁCE

SUPERVISOR — **Ing. Lenka Smolíková, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Studentka:	**Bc. Zuzana Mazáková**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Lenka Smolíková, Ph.D.**
Akademický rok:	2017/18

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Využití agilní metodiky při řízení vývoje softwaru

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Návrh řešení a přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je zefektivnění procesu vývoje softwaru integrováním agilní metodiky do řízení projektu.

Základní literární prameny:

APELLO, Jurgen. Management 3.0: leading Agile developers, developing Agile leaders. 1st edition. Upper Saddle River, NJ: Addison-Wesley, 2011. 458 p. ISBN 0-321-71247-1.

BURKE, Rory. Introduction to Project Management. 1st edition. Norwich: Burke Publishing, 2007. 283 p. ISBN 0-9582733-3-2.

LOCK, Dennis. The Essentials of Project Management. 2nd edition. Great Britain: Gower, 2001. 254 p. ISBN 978-0-566-08805-6.

RUBIN, Kenneth S. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Upper Saddle River, NJ: Addison-Wesley, 2012. 452 p. ISBN 978-0-13-704329-3.

SHORE, James and Shane WARDEN. The Art of Agile Development. 1st edition. Sebastopol, CA: O'Reilly Media, 2007. 432 p. ISBN 978-0-596-52767-9.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2017/18

V Brně dne 28.2.2018

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstract

This master's thesis is focused on improvement of the management process in software development by implementing the agile methodology Scrum into the project. The proposed solution contains core principles and elements of the methodology and their recommended application in the project ABC with the accordance to project's unique characteristics.

Key words

agile methodologies, Scrum, Waterfall model, software development, distributed team, management style

Abstrakt

Diplomová práce se zaměřuje na zefektivnění procesu řízení ve vývoji softwaru, a to integrováním agilní metodiky Scrum do projektu. Navrhnuté řešení obsahuje stěžejní principy a součásti metodologie a jejich konkrétní doporučenou aplikaci v projekte ABC v soulade s jeho specifikami.

Klíčová slova

agilní metodiky, Scrum, Waterfall model, vývoj software, distribuovaný tým, styl řízení

Rozšířený abstrakt

Diplomová práce je zaměřená na efektivní využívání agilních metod v procesu řízení projektu v oblasti vývoje software. Práce se zabývá konkrétním projektem ve firmě SDE Software Solutions s. r. o., na kterém se podílejí pracovníci z USA ale i z Brna a tvoří společný tým. Spolu se podílí na vývoji několika produktů, především pro americké zákazníky.

V minulosti byl tento projekt převážně řízený na základě modelu Waterfall. Postupem času se firma rozhodla, pro potřebu pravidelného dodávání produktu v co nejlepší kvalitě, implementovat do projektu i metodiku Scrum. Tato implementace proběhla neúspěšně a konkrétní principy a změny se v projektu nikdy úplně neukotvili ve své doporučené formě. Vzhledem k tomu, že v tomto projektu existuje mnoho problémů s vývojem produktů včas a v požadovaném rozsahu, přirozeně se členové týmu a managment podniku snaží je vyřešit v co nejkratším čase a s co nejnižšími náklady.

Pro vytvoření efektivního, nákladově nenáročného a proaktivního řešení této situace, je nutné důkladně zanalyzovat vnitřní prostředí projektu. Vzhledem k tomu, že se jedná o tým rozložený v Brně i v USA, je nutné nejprve přesně zanalyzovat současný stav definice rolí a zodpovědnosti konkrétních členů týmu Force. Z této analýzy vyplývá, že role a s nimi spojené činnosti nejsou přesně naplánované, což v reálném příkladu může znamenat, že testeři aplikací mají někdy velmi málo času na otestování funkcionality aplikace. Naopak v některých případech nemají většinu času v jedné iteraci vývoje produktu přesně vyčleněnou práci z důvodu zpoždění vyvíjených částí aplikace.

Další oblastí analyzovanou v projektu Force je plánování a rozvrhnutí činností v jedné iteraci vývoje produktu. Analýza odhaluje, že oblast plánování aktivit na základě požadavků zákazníka je neefektivní z důvodu nezapojování celého týmu. Nedostatečná informovanost je jedním z klíčových aspektů dezinformací a neshod, které v týmu vznikají. Provázanost konkrétních fází vývoje aplikace, včetně samotné implementace a testování, je nedostatečně definovaná, což vede k tomu, že členové týmu často nevědí, v jakém stavu se vyvíjená funkcionalita nachází.

Co se týká používaných informačních technologií na projektu, tým Force disponuje svým vlastním informačním systémem, který poskytuje mnoho užitečných funkcí, jako například verzování softwaru. Jeho neefektivní využívaní spočívá především

v nedostatečném popisu práce v systému, včetně zodpovědných osob a aktuálním stavu položek. Tým proto často netuší, zda část aplikace je již připravená na testování, nebo se jedná již o finální vydání pro zákazníka. Většina těchto neshod se řeší na každodenním meetingu, který byl původně určený na rychlé střetnutí členů týmu, informování o aktuálním stavu položek, na kterých se pracuje a blokujících elementech, které zabraňují členům týmu pokračovat v práci. Namísto tohoto původního účelu se nyní tým zabývá neefektivním řešením problémů způsobených tím, že neexistují jednotlivá pravidla pro vykonávané procesy v projektu.

Z analýzy také vyplývá, že v týmu není kladen důraz na zaznamenávání dokumentace a její aktualizace. To vede k dalším problémům v týmu, protože developeři si nepamatují verzi software vydanou před několika lety a vzhledem k tomu, že nebyla zdokumentovaná, je velmi těžké informace získat zpětně. Při rozsáhlosti produktů vyvíjených v tomto projektu, je zaznamenávání dokumentace architektury aplikace a funkcionality, včetně návaznosti hlavních prvků systému, velmi důležitá. Zdlouhavé zpětné dohledávaní klíčových informací a vzájemné obviňování vytváří v týmu nepříznivé prostředí.

Řízení projektu ABC se vyznačuje aplikováním především Waterfall modelu a některými, neefektivně využívanými praktikami z agilní metodiky Scrum. Tradiční přístup k řízení projektu není nejvhodnějším řešením, z důvodu často se měnících požadavků ze strany zákazníků na výsledný produkt. Tradiční navázání jednotlivých fází vývoje aplikace neumožňuje dostatečně promptně reagovat na změny a adaptovat se na ně. Jak již bylo zmíněno, artefakty a principy z metodiky Scrum, které byli v minulosti implementované, nejsou v současnosti vhodně využívané. Příkladem jsou Sprinty, teda jednotlivé iterace vývoje, které nemají přesně definované doby trvání, dále chybějící role Scrum Mastera, nebo plánovací meetingy bez účasti celého týmu.

Pro potřeby zanalyzování projektu ABC a určení stavu, v kterém se z hlediska složitosti a způsobu vykonávaní činností nachází, je využitá v práci metoda Cynefin Framework, která napomáhá určit doménu, v kterém se projekt nachází. Projekt ABC je identifikovaný jako typický příklad projektu v komplexní doméně. To znamená, že je vhodným kandidátem na využití agilní metodiky Scrum, případně její částí pro to, aby se

zefektivnilo procesní řízení projektu a práce celého týmu. Závěrečné zhodnocení slabých míst v projektu, silných stránek, příležitostí a hrozeb je zahrnuté ve SWOT analýze.

V následující části práce se nachází návrh řešení, které mají přinést zlepšení v problematických oblastech projektu, které byli identifikované v analytické části. Z analýz vyplynul požadavek na implementaci agilní metodiky Scrum do procesu řízení projektu ABC. Nejedná se o kompletní přeměnu projektu na Scrum projekt, ale o výběr nejdůležitějších a nejpotřebnějších principů a nástrojů z metodiky a jejich implementaci do projektu.

V prvé řadě se jedná o přeměnu dosavadní struktury týmu. Návrh začlenění role Scrum Mastera do týmu má dopomoct odstranit komunikační problémy týmu, ulehčit organizací meetingů, a rozložit proces správy požadavek a plánovací činnosti mezi Scrum Mastera a Product Ownera. Další změna spočívá v kladení důrazu na tvorbu a správu dokumentace, efektivnější využívání nástrojů, které nabízí informační systém a vytvoření Scrum Board pro lepší orientaci v práci rozložené do dané iterace vývoje produktu. Tým si taktéž vytváří svou vlastní stupnici hodnocení množství práce, pro potřebu ulehčení procesu plánování a zvládání náporu proměnlivosti požadavků zákazníka. Definováním konkrétního stavu vykonaných položek se odstraní nejasnost v dalším postupu práce a tým získá přehled o množství práce, kterou je potřeba vykonat. Pomocí začlenění retrospektivy do průběhu Sprintu se tým podílí na nepřetržitém zdokonalování procesů, které se vykonávají při vývoji produktů. Zhodnocením toho, co tým zvládl v dané iteraci včas, které procesy potřebují vylepšit a definováním konkrétních kroků pro odstranění problémů, se tým neustále posouvá k vytýčeným cílům, a to za využití těch nástrojů Scrumu, které přináší v projektu zlepšení.

Součástí práce je taktéž ekonomické zhodnocení navrhované změny, která obsahuje přehled o počátečních nákladech na zavedení Scrumu a přírůstku v týmu o Scrum Mastera. Náklady na přetransformování procesů v projektu, především v oblasti komunikace a sounáležitosti týmu, nejsou přesně vyčíslitelné, avšak přínos těchto změn spočívá v příbytku nových zákazníků, včasném dodávání funkcionalit a řešení, kvalitních produktech, a získání konkurenční výhody na trhu.

Bibliografická citace

MAZÁKOVÁ, Z. *Využití agilní metodiky při řízení vývoje softwaru.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2018. 93 s. Vedoucí diplomové práce Ing. Lenka Smolíková, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracovala jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušila autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 15. května 2018 ……

podpis autora

Poděkování

Touhle cestou bych se ráda poděkovala Ing. Lence Smolíkové, Ph.D. za odborné vedení mé diplomové práce, její cenné rady a pomoc s řešením problémů. Dále bych ráda poděkovala své nejbližší rodině, která mě ve studiu podporovala.

CONTENTS

INTRODUCTION

Project management in software development is a specific domain and often requires a different approach to processes and resources management. Software development heavily depends on the ability to react and adapt, since it is an environment of ever-changing requirements. Classic models, such as Waterfall are still widely used and have their place in software development projects, however, modern approaches are offering variety of principles crafted namely for the software development. With the uniqueness of every single project, it is up to the project manager, team members and a customer to pick and refine the most suitable and cost-effective solution that all interested parties can benefit from. Agile methodologies have become very popular thanks to their adaptive nature, simplicity and proven efficiency. Using their core principles and best practices can significantly improve the development process in any project.

The main objective of this master's thesis is to help a software development project implement and use the agile methodology Scrum to their advantage. Adopting Scrum or any other methodology is an evolving process, not a single activity. Therefore, it takes time and effort of the whole team and management to completely grasp all the core principles and best practices and apply them in the unique environment of every project.

The first part of the thesis contains all necessary information about classical and agile approaches to the software development management, as well as the breakdown of the software development life cycle. The analysis is the second part of the thesis and its main purpose is to help to execute the change effectively. It contains a thorough analysis of the concurrent situation and unsuccessful past attempts to adopt Scrum. The third main part focuses on the proposed solution based on all the information collected from the analysis.

GOALS OF THESIS AND METHODS

The main goal of the thesis is to improve the efficiency of the software development process by integrating the Agile methodology into the project management. Usage of agile methodology is meant to help the project ABC to promptly react to changes in the software development environment and adapt flexibly and quickly with minimal costs to the company and its customers. Gaining an advantage against competitors in the software development sphere and becoming a supplier of high-quality and on-time solutions, are all perks of using continuous improvement and iterative approach in the development process.

In order to achieve the main goal of this thesis, there is a need to fulfil several complemental goals. The first one is to layout the necessary theoretical foundation of the problematics, such as traditional development model, agile models and methodologies, software development process and life cycle. The theoretical review is created using available sources of knowledge dedicated to the best practices used in the field of software development management, as well as recommendations and guides to improve the efficiency of the project management.

Based on this theoretical background, another complemental goal is to analyze the concurrent situation in detail, such as processes executed in the project, team members roles and responsibilities, used information technologies and a problematics analysis. Cynefin Framework and the SWOT analysis are used to determine the current state of the project and help to identify all weaknesses and strengths of the project, as well as all opportunities and threats affecting the project from the outside.

The final proposal of the most suitable solution for this project is then created with the accordance to all the gathered information about the concurrent situation of the project. Apart from the solution with all steps that are necessary to be executed, the thesis contains a cost analysis and a conclusion of the most significant benefits of the proposed changes.

1 THEORETICAL REVIEW OF PROBLEM

This chapter contains the necessary knowledge regarding project management, software development methods and mainly Scrum, as an agile approach to software development life cycle. This theoretical background is a foundation for analysis and further proposal of improvements in the dedicated chapter.

1.1 Software development process and life-cycle

Various processes and methodologies selected to develop the project according to its purpose and objectives are called development models. Their key role is to help improve the software quality as well as the overall development process. These models are suitable for software development life-cycles (SDLC), each crafted to fulfill different objectives. SDLC is an environment representing a set of activities performed in each stage of the software development process (Figure 1). It includes a detailed plan for conducting the development, maintenance and replacement of a specific software. SDLC is also known as software development process. The international standard for SDLC is ISO/IEC 12207, aiming to define all the activities required to develop and maintain software [13].

The first and the main stage of SDLC is the analysis of requirements, often performed by senior team members, using inputs from customers, sales department, market research, and industry experts. The gathered knowledge is then used for creating a basic project plan and feasibility study from economic, operational and technical points of view. It ends with the definition of carious technical approaches that can be used in order to implement the project with the minimum of risks [13].

The second stage is definition of product requirements. Afterwards, they are documented and must be approved by the client or by market analysts through software requirement specification (SRS). SRS is a document listing all product requirements needed throughout the project life-cycle. It is also a foundation for the architects that create the best possible architecture for the product, which is the third stage of SDLC. Writing a design document specification (DDS) is another crucial part of SDLC. It often contains more than one suggested approach to choose from. This document must be revised by all the interested parties, since the goal is to select the most suitable approach.

Selection is performed based on several criteria, such as risk evaluation, product robustness, design method, budget, and time constraints [13].

The fourth stage of SDLC is the first stage of the product development itself, including writing the source code. It is crucial that the design in previous stages has been performed in a detailed and organized manner, since all he developers refer to guidelines provided by their organizations. The activity of producing source code requires choosing proper programming tools and languages according to the product that is being developed. Product testing is the only stage that is performed throughout all the other stages of the software development. It consists of reporting faults, tracking, fixing and reanalyzing in order to ensure the compliance with the SRS quality requirements. The last stage of SDLC is the market operation and maintenance. It is performed once the product has already been tested and is ready to launch on the market. Often, the product is delivered and tested in a limited segment of real business environment, and based on the final feedback, it is then launched in the whole market [13].

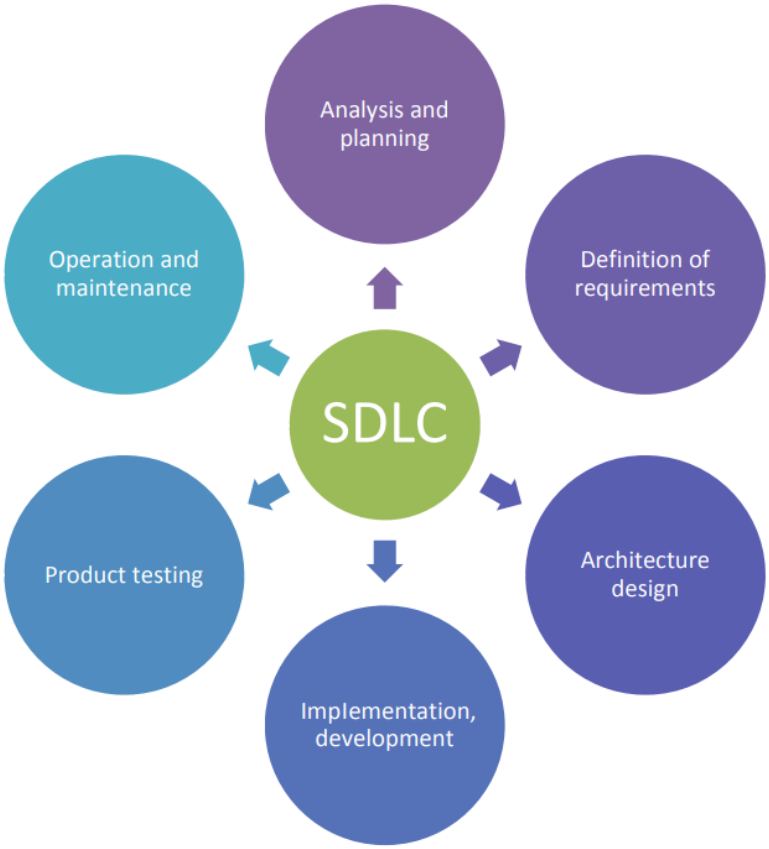


Figure 1: SDLC

(Source: [13])

1.2 Traditional approach to software development

The traditional approach to development of software is based on a predictive approach. Projects using the traditional approach always possess a highly detailed plan and have a full list of characteristics and tasks that ought to be completed in the next period of time. It is best suited for large-scale projects. Predictive methods are completely relying on the requirement analysis and careful planning at the beginning of the project. In case any change emerges, it needs to go through a thorough change control and prioritization [13].

The traditional approach is process-centric, always guided by the belief that all the sources of variations are identifiable upfront and may always be eliminated by continually controlling and refining processes [15].

1.2.1 Waterfall Model

The Waterfall approach, also known as linear-sequential life cycle model, is a traditional model of software development defined by Winston W. Royce in 1970 [13].

It describes a set of straight-through processes, such as thorough planning, specifying the way the product will behave, design the architecture, work out the detailed design and after all these activities, begin implementation including coding and testing, and debugging. The stages of Waterfall are given in Figure 2 [11].

The advantages of this model are, that documentation and structure design are already prepared when a new member joins the team. Waterfall model is very easy to understand and use. Since the model is very rigid, it is easy to coordinate expected result and an evaluation process in each stage [13].

One of the main reasons why software companies might avoid using pure Waterfall approach is that it is near impossible to completely specify and plan every aspect of the project in advance of writing any code. It is much more common to use a lifecycle model resembling Waterfall structure, but with incorporating a possibility to swim upstream the waterfall structure, hence, allowing to add or modify other stages [11].

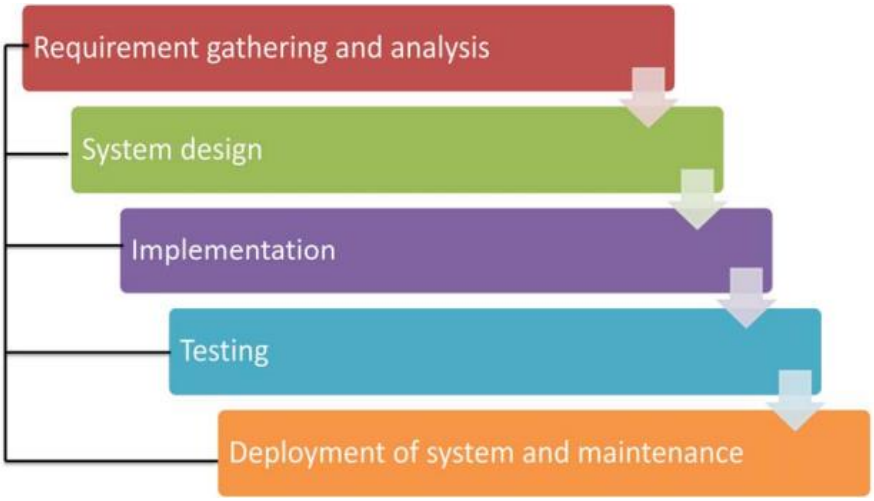


Figure 2: The Stages of Waterfall Model

(Source: [13])

Waterfall model is recommended for all the short projects with the requirements well understood, clear, and final. It is also important that product definition is stable, and technology is fully understood. There should be no ambiguous requirements and resources involving expertise should be freely available [13].

1.3 Agile approach to software development

Agile as a method was born in February of 2001 when a group of developers interested in lightweight development methodologies met to talk about their views regarding effective development of software. The developers understood the importance of a model in which every development cycle iteration would learn from the previous one. Hence, this new methodology was much more efficient, flexible and team-oriented than any previous model [2].

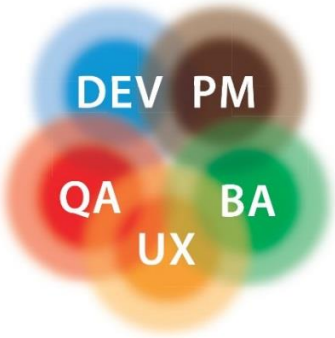


Figure 3: Agile Approach Team Roles

(Source: [6])

One of the main differences in comparison to traditional approaches, in agile environment, team roles tend to blur. Joining an agile team is a lot like working in a mini-startup. All the members do whatever it takes to make the project successful regardless of title or their role in the team (Figure 3). However, team members still have their core competencies. Another difference is, that analysis, coding, design and testing are continuous activities (Figure 4). They do not exist in isolation anymore and team members need to be working together daily throughout the project [6].

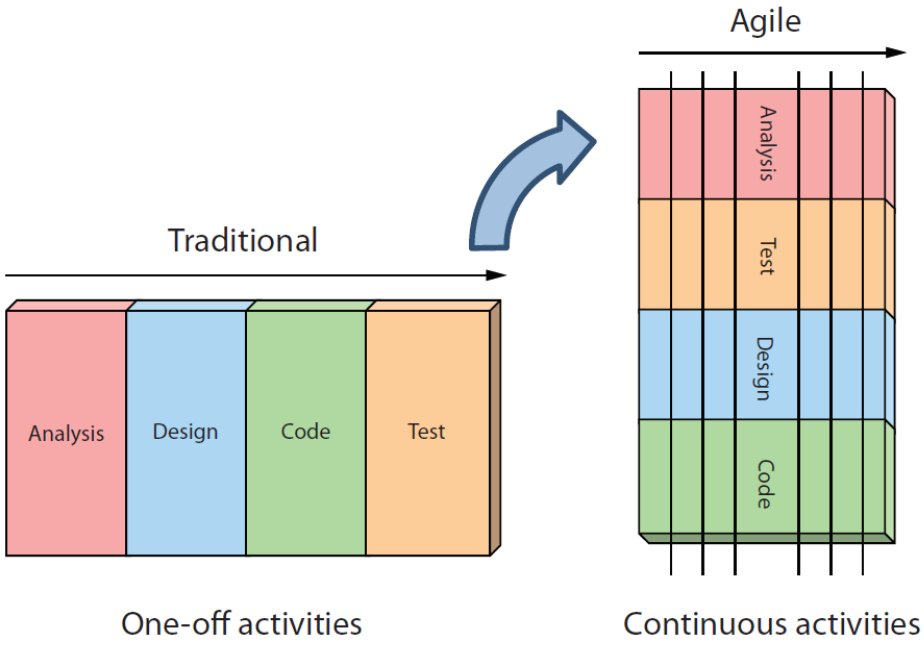


Figure 4: Activities in Traditional vs Agile Approach

(Source: [6])

As far as the quality assurance is concerned, in Agile approach it is everyone's responsibility to ensure it, whether they are working on the code, managing the project or designing the architecture (Figure 5) [6].

Mixing formal and ad-hoc development approaches resulted in creating many different agile methods, such as Evo, Scrum, DSDM, Crystal, Extreme Programming (XP), Feature-Driven Development (FDD), Pragmatic Programming, and Adaptive

Software Development [5]. Even though there are many different agile methods, they all look up to the Agile Manifesto and its 12 core principles for guidance [2].

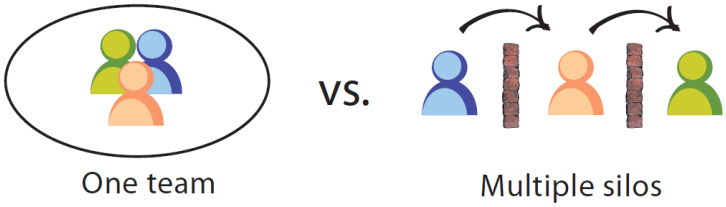
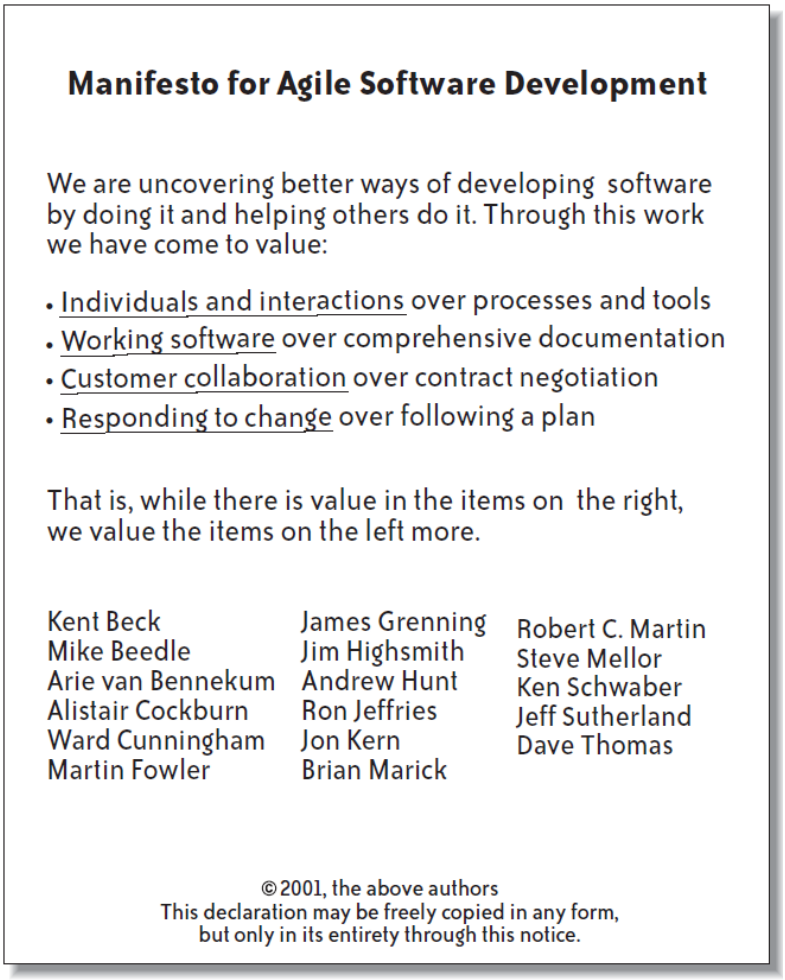


Figure 5: Quality Assurance in Agile Approach vs Traditional Approach

(Source: [6])

1.3.1 Agile Manifesto

Agile Manifesto, as published on the Agile Manifesto website, is the fundamental guidance used by Agile teams (Figure 6), with its 12 core principles (Figure 7).



Manifesto for Agile Software Development

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

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

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

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

Figure 6: Agile Manifesto

(Source: [5])

Agile approach considers people as unique individuals instead of replaceable resources and puts great emphasis on their interactions and collaboration instead of just their knowledge. Agile focuses on creating a small team with cross-functional units, that are preferably located in the same room. There is no method or process imposed on these teams. On the contrary, they are supposed to self-organize. These teams are trusted to get the workload done using approaches that they themselves think are best [4].

Principles behind the Agile Manifesto

We follow these principles:

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

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity, the art of maximizing the amount of work not done, is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 7: Principles behind Agile Manifesto

(Source: [5])

In order to achieve the best results while creating a product, agile approach focuses on including the customer in the whole process of development. The collaboration between the team and the customer means continual reprioritizing and maintaining an ever-changing backlog of features, that are described in a concise format. Their documentation begins as soon as they are selected for immediate implementation by the team. Each feature is described in compliance with simplicity to achieve the best possible design. The functionality of these features is verified by the customer as soon as the implementation process is finished [4].

One of the fundamentals of Agile approach is a focus on quality. This is crucial for creating successful products. One of the ways to ensure the quality is to use Test-Driven Development (TDD). Using TDD means writing test code before writing production code. Among other important tools, using code reviews done mostly as pair programming, Definition-of-Done checklists, iterative development as adapting code due to changes or new insights, and refactoring as continual improvement of code, all help to achieve the best possible architectures for products. High-quality architectures are never defined up-front as far as Agile approach is involved. They are allowed to form further during development of the product. However, if there is need for early definition of the architecture, then it is done only in a basic form [4].

Even though there are many useful tools described and promoted throughout the Agile literature, tools are not considered crucial contributors to successful products. Tools for daily builds, continuous integration and automated testing are the most preferred ones among experienced Agile teams [4].

Since motivation is very important for Agile team members, there is need for removing repetitiveness from daily tasks. This leads to implementation of automation processes throughout the development. Many Agile teams thrive for supportive environments, such as open office layouts and tools such as big task boards and burn charts (Figure 8). To sum up, tools in Agile context are supposed to strengthen motivation, communication, and collaboration in a team [4].

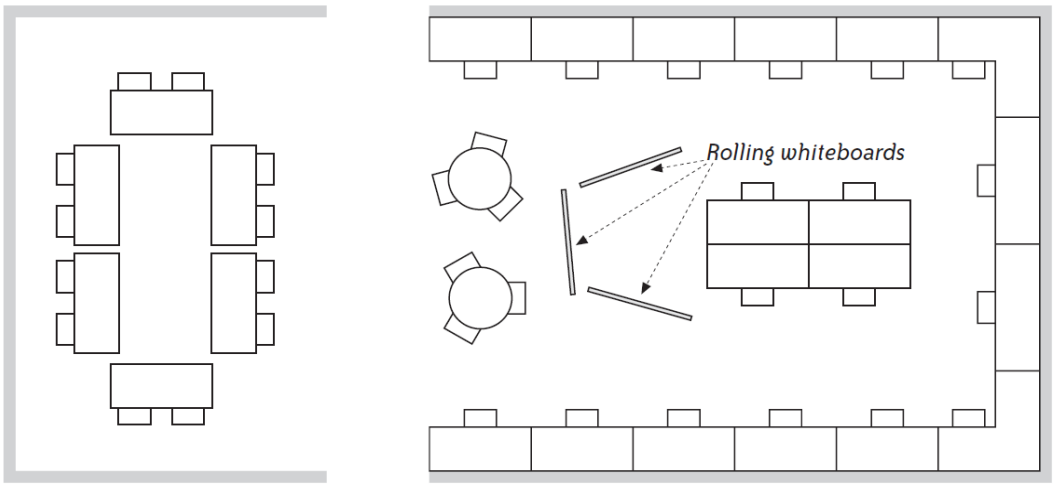


Figure 8: An Example of Agile Office

(Source: [5])

When it comes to delivery dates and deadlines in Agile projects, as much as budgets, they can be chosen almost arbitrarily. Software is being produced in short-time frames and delivered in incremental releases. Every release should present a potentially shippable product. As a result, business owners can take control of timing and even move release dates back and forth according to what features they would like to present to customers and when [4].

Agile Manifesto was, among other reasons, crafted in order to address the need of response to change, which is a common trait of software development environment. For example, features that were considered the most valuable yesterday might become useless today, including the features that had already been delivered in the past. To cope with these situations, Agile teams nurture short feedback and delivery cycles [4].

In Agile projects, there are some essential processes required, such as minimal planning, daily face-to-face meetings also known as standups, and measurement of progress by evaluating working software, hence, the features accepted by the customer [4].

1.3.2 The most common mistakes in adoption of Agile approach

Among many advantages of choosing the Agile approach in software development, there are many mistakes that teams and management can make while pursuing the perfect project management [6].

The most common pitfalls of misunderstanding or misusing the Agile approach are:

- focusing only on construction,
- blindly following new rules while ignoring unique needs of the project or company,
- improper planning,
- excluding the entire organization, existing only in a limited space of one delivery team, and not looking at the entire process around solution delivery,
- lack of executive support,
- adopting the approach too quickly leading to problems with scalability issues, extending tools, missing definitions of processes for dealing with multiple dependent or distributed teams,
- insufficient coaching,
- retaining traditional governance, such as project funding, change control, and phase gates,
- skimping on proper training and tooling in order to cut on costs.

1.4 Extreme Programming

One of the popular agile approaches is called Extreme Programming (XP). Its main focus is putting the customer first. XP teams thrive to achieve high customer satisfaction by developing all the features precisely when the customer wants them. Team's daily routine consists of handling new requests that appear frequently, always prioritizing work in accordance with the most pressing matters and solving them with high efficiency [6].

The core principles and practices of XP are [6]:

- established coding standards and guidelines that all team members should follow,
- collective ownership encouraging transparency and accountability for work quality,

- continuous integration that prompts team members to check in code changes frequently and integrating the system to ensure that all the changes work and that the rest of the team is always working with the latest version of the system,
- Test-Driven Development (TDD) that begins with coding a simple test, then continues with updating the functional code to make it pass the test and ends with getting the software running and iterating this process,
- detailed requirements captured just-in-time (JIT) in the form of acceptance tests,
- refactoring, hence, enabling the team to evolve their work slowly over time by doing minor changes in source code, database schema, or user interface etc.,
- pair programming, allowing two programmers to work together on the same task at the same time,
- guiding the product into successful delivery by applying the planning game,
- always seeking the simplest way to write their code,
- frequent deployment of valuable, working software into production, that build confidence in the team and trust from the customer,
- sustaining an energized approach encouraging to work at a constant, and gradually improving pace,
- having a team, where all members dispose of all the necessary skills to deliver a working and high-quality solution for a customer.

1.5 Lean Programming

With origins in manufacturing, Lean approach was born in the 1940s in Japan, the company Toyota. At that time, Toyota was not able to afford any major investment required for mass production. Hence, studying supermarkets and customers' needs and the way they buy the products, Toyota created a JIT process that could translate the way supermarkets operate to the factory floor. As a result, Toyota managed to achieve a significant reduction in inventory of parts, finished goods, and a major cut in investments to people, space, etc. Thanks to JIT, workers have the ability to freely make decisions about what is the next most important task to head to. All the responsibility for the results goes to workers themselves. This major success of relying on JIT processes globally affected mass manufacturing approaches [6].

The seven principles of lean manufacturing can be applied to optimize the whole IT value stream as well. Its core development principles are [6]:

- eliminate waste,
- create knowledge,
- defer commitment,
- deliver the product quickly,
- respect people, including customer and team members,
- optimize the whole, not just a part.

1.5.1 Kanban

Kanban, as a method used in software development, is a lean methodology aiming for improving on existing systems. There are two core principles that are crucial to success while applying Kanban to software development [6].

The first one is visualizing workflow. In Kanban, teams rely on using a Kanban board, often in a form of white board, corkboard, or electronic board, that displays kanbans. The term kanban is used for describing indications of where in the process a particular piece of work is. Kanban board is always split into several columns. Each of them represents a stage in the process, a work buffer, or queue. There are also optional columns tailored by team members to meet their specific needs and help them with efficiency of their work. The board is always updated by team members whenever work proceeds, and all the blocking issues are always identified on a daily meeting [6].

The second core principle emphasizes the need of limiting work in progress (WIP). By doing so, not only team reduces average lead time, but also improves the quality of the work delivered. Among other advantages, overall productivity of the team increases, and the team will be able to deliver frequent functionality, hence, build trust with all stakeholders at sustainable pace. In order to limit WIP, team needs to pinpoint exact position of blocking issues, address them as quickly as possible, reduce queue and buffer sizes whenever possible [6].

1.6 Scrum

Scrum is one of the agile approaches for developing new innovative products and services. Development using Scrum begins with creating a product backlog. This prioritized list of features and other necessary capabilities is required for delivering a successful product. Following the product backlog ensures that the team is always working on the most important and highly prioritized task [1].

Typically, work is delivered in short, timeboxed iterations. Their length ranges from one week to a calendar month. Team is self-organized and cross-functional, executing tasks such as designing, building and testing. Each iteration starts with planning a highly-prioritized version of the product backlog, since the original product backlog workload is too large to be completed in a single iteration [1].

At the end of iteration, the team and stakeholders meet to review the finished work to alter both planned work and the way the team plans to do it. In case any additional needed features appear, the Product Owner can create a new item for it and include the item in the product backlog, already ranked with corresponding priority [1].

If appropriate, a shippable version of the product is released either at the end of each iteration or after a couple of iterations. Each iteration ends with the planning of the next one [1].

1.6.1 Scrum Origins

The first breakthrough article describing the importance of empowered, self-organizing teams and outlining management's role in the development process is "The New New Product Development Game" (Takeuchi and Nonaka 1986) that appeared in the Harvard Business Review. The article describes how companies such as Honda, Canon and Fuji-Xerox delivered world-class results thanks to a scalable and team-based approach to all-at-once product development [1].

Its important influence resulted in weaving various concepts together and giving rise to what today we call Scrum. The term Scrum is borrowed from the sport of rugby. It refers to a way of restarting a game after an accidental infringement or when a ball has gone out of play. The authors of the article use this term to relate to product development process. They bring up the importance of the team acting as a unit while delivering the

product, or metaphorically passing the ball back and forth while going to the distance as a unit. According to the authors, this approach is better suited to nowadays competitive requirements [1].

In 1993, Jeff Sutherland and his team at Easel Corporation took the ideas from the 1986 article and combined it with the concepts from object-oriented development, empirical process control, iterative and incremental development, software process and productivity research, and complex adaptive systems, to create Scrum process [1].

Since then, many Scrum-specific publications have been published by authors such as Ken Schwaber and Jeff Sutherland, including Agile Software Development with Scrum (Schwaber and Beedle 2001), Agile Project Management with Scrum (Schwaber 2004), and The Scrum Guide (Schwaber and Sutherland 2011) [1].

1.6.2 Choosing Scrum

Scrum has mostly been used to develop software products, but its core values and principles can be used to develop various products, or for organizing the flow of several types of work, such as hardware development, marketing programs or sales initiatives [1].

Companies using Scrum effectively benefit from delivering exactly the results that their customers need, and not just the features they specified at the beginning of the project. Delivering smaller and more frequent releases and exposing organizational dysfunction or waste reflects in the improvement of the return on investment and the reduction of cost [1].

Using Scrum leads to creating working, integrated, tested and business-valuable features each iteration. It is well suited to help companies adapt to interconnected actions of competitors, customers, users, and stakeholders. Another positive effect can be observed on team members, as they can enjoy frequent and meaningful collaboration and hence, improved interpersonal relationships and mutual trust among all team members [1].

Applying Scrum is not suitable in every project, as Cynefin framework explains. This topic is described in the following chapter.

1.6.3 Cynefin Framework

Cynefin Framework (Snowden and Boone 2007) is often referred to as a sense-making tool that helps us to identify and understand the current situation of our project, as well as choose a situation-appropriate approach we can operate with (Figure 9). Cynefin Framework defines these five different domains: simple, complicated, chaotic, complex and disorder [1].

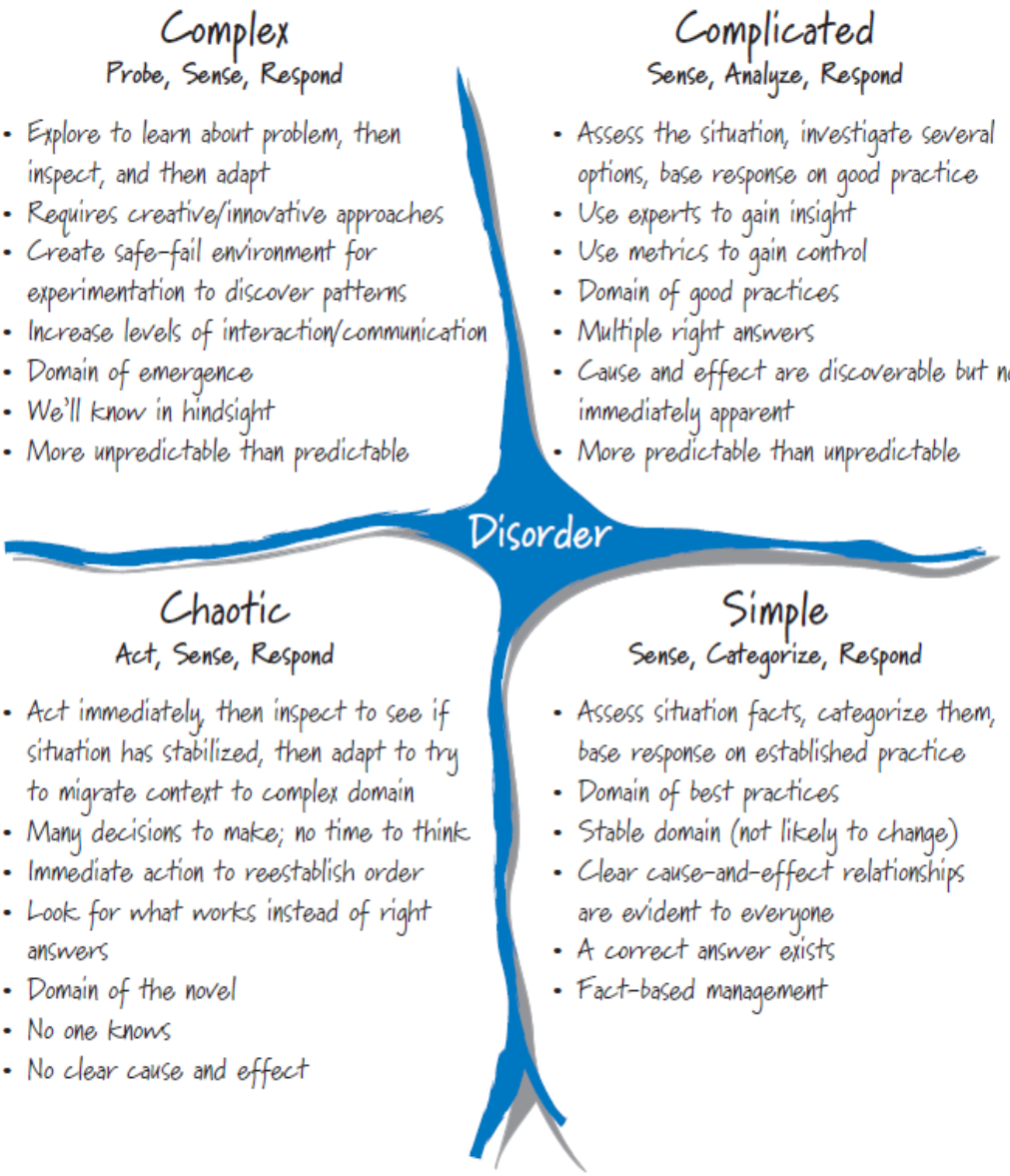


Figure 9: Cynefin Framework

(Source: [1])

Simple domain is the domain of legitimate best practices. While dealing with simple problems, everyone can see the cause and effect. Hence, the right solution is obvious most of the time. Even though Scrum can be used for small problems, it is not the most efficient tool for dealing with them. Using a process with a well-defined, repeatable set of steps is more suitable in this domain [1].

Complicated domain is dominated by experts and it is the domain of good practices. Expert diagnosis is required to figure out the right solution for a problem in the complicated domain. Although Scrum might be applicable in this domain, Six Sigma and similar quantitative approaches are preferable. A lot of problems, such as dealing with performance optimization are better served by assembling the right experts and letting them assess the situation [1].

Chaotic problems require a rapid response to prevent further harm and reestablish at least some order. For example, a chaotic problem might be having a customer filing lawsuit against a company due to faulty algorithm, causing him large damages. At the same time, the main designer of the algorithm might be unavailable. There is need to act immediately and decisively, since someone needs to take charge of the situation. Again, Scrum is not the best solution in this domain [1].

Complex domain is typical for unpredictable environments. It is the domain of emergence, since we cannot see the right solution clearly. There is need to explore to learn about the problem and then inspect and adapt. Scrum is particularly well suited for this domain, as there is need for innovative and creative approach. High levels of interaction and communication are essential. It is typical for innovative new-product development, as well as for enhancing existing products with new innovative features [1].

Project is in the disorder domain when there is uncertainty about fitting in any other domain. In this case, people tend to act accordingly to their personal preference for action. Solution for this dangerous position is to break down the situation into smaller parts and assign each of them to the other four domains. There is not any official approach recommended in this domain, as there is a need to get out of this domain as soon as possible [1].

1.6.4 Scrum Roles

Development of software while using Scrum requires one or more Scrum teams, consisting of three Scrum roles: Scrum Master, Product Owner and the development team. Even though there might be more roles included in the team, only these three are necessary [1].

Product Owner

Product owner is the crucial point of product leadership and the only authority responsible for deciding which features and functionality to build, as well as the order in which to build them. The Product Owner must maintain and communicate a clear vision of Scrum team's goal to all the participants of the development. As such, he or she is responsible for the developed and maintained solution's overall success [1].

Whether the focus is set on an external or internal product; the Product Owner is obliged to making sure that the most valuable work is always performed. In order to ensure that the team rapidly builds the correct and most desired solution, the Product Owner closely works with the Scrum Master and the development team [1].

The core responsibilities of a Product Owner are managing economics, participating in planning, grooming the product backlog, defining acceptance criteria and verification of their fulfilling, collaboration with the development team, and the stakeholders [1].

The Product Owner is a combination of several traditional roles that exist in non-Scrum teams, mainly: product manager, product marketer, project manager, business analyst, and acceptance tester [1].

Scrum Master

The significance of the role of Scrum Master lies in helping everyone involved understand and embrace the Scrum values, principles, and practices. Scrum Master is a coach that provides leadership in helping the whole organization to develop organization-specific Scrum approach. As such, Scrum Master helps the organization in difficult period of adopting Scrum [1].

Another crucial responsibility of the Scrum Master lies in protecting the team from any interference coming from outside, as well as removing any impediments that might endanger the team productivity. He or she has no authority to express control over the

team and is rather a leader than a manager which is present in the activities he or she is occupied with throughout the day in a Sprint (Figure 10). Typical Scrum activities consist of working on the product backlog grooming activities, such as writing and prioritizing new product backlog activities with the Product Owner [1].

Scrum Master helps to remove barriers between the roles in the team, allowing the Product Owner focus on driving the development directly. Even though Scrum Master is not able to solve the problems of team members, he or she always helps and guides them as they solve the problems themselves [1].

While coaching a new Product Owner, the Scrum Master assists the Product Owner in understanding the role and its core responsibilities, maximizing business outcomes using Scrum, listening to any complaints or requests for change and then parsing them to actionable improvements for the team [1].

Scrum Master is always looking for opportunities to make the team more effective. He or she is the process authority of the team, ensuring that the team adheres to the Scrum values, principles, and practices along with any team's special approach or adjustment to Scrum. In order to maximize the business value of using Scrum approach, there is need for continuous improvement of processes throughout the development [1].

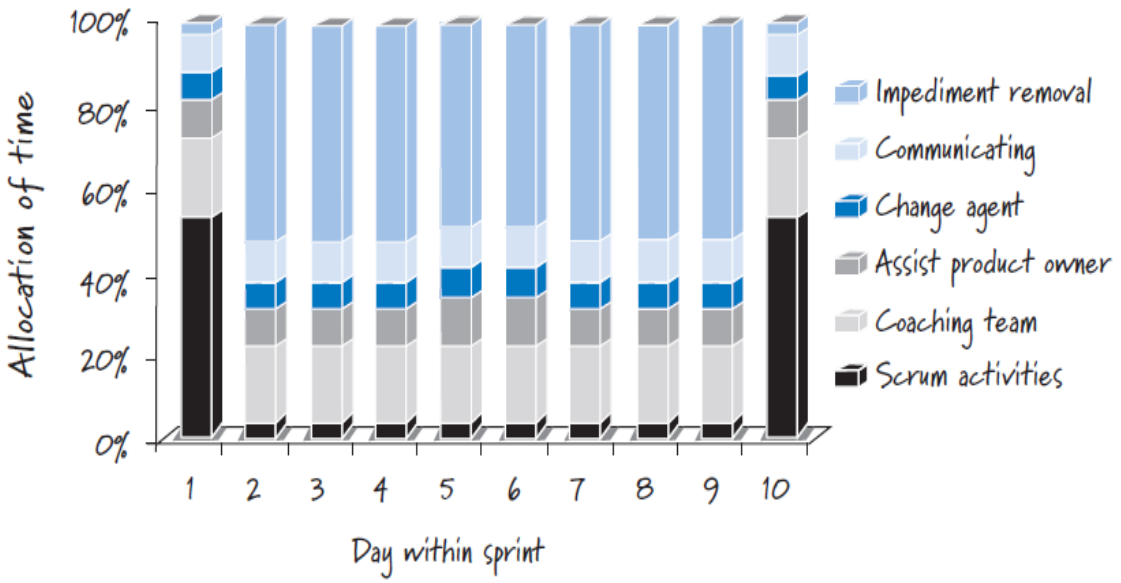


Figure 10: Daily Activities of the Scrum Master

(Source: [1])

Development Team

Typically, the development team consists of team members such as developers, designers, testers. However, in Scrum, there is need for a team skilled in all the aspects of development, including the design, development, integration, and testing of the created functionality. These teams are not necessarily role-specific, since there is no need for the team member to hand off the work to another when it is finished. Another difference in comparison to classic teams is that the development team doing the work during the Sprint should also execute the testing. Hence, it is important to create cross-functional teams while applying Scrum approach to software development [1].

Each member should have so-called T-shaped skills (Figure 11). This means, that they should have deep skills in their preferred functional area, such as User Experience (UX). At the same time, they should be able to work outside of their core specialty area, for example helping with execution of testing or maintaining documentation. On the other hand, it is not expected that every team member is able to perform any assigned task. If it is not possible to form a team using T-shaped skilled members, there should be at least the effort to evolve the skills throughout the development process.

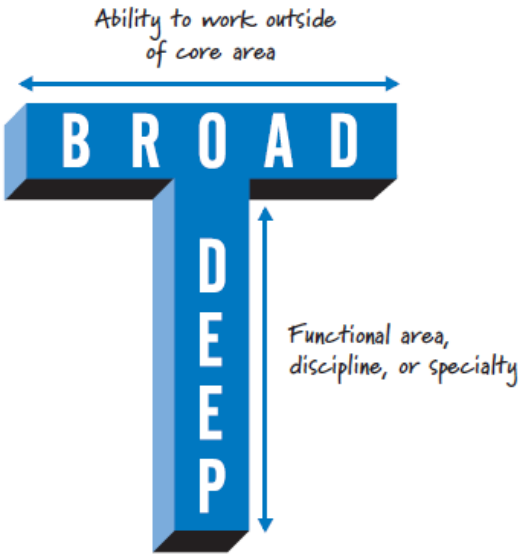


Figure 11: T-shaped Skills of the Development Team Members

(Source: [1])

The main responsibility of the development team is the Sprint execution and they spend the majority of their time performing this. They need to perform the creative work

of designing, building, integrating, and testing of the product backlog items while creating an increment of potentially shippable functionality. The team members self/organize in order to collectively plan, manage, carry out, and mainly communicate work [1].

It is expected that each team member actively participates in daily Scrum, as the team collectively inspects progress toward their Sprint goal. The Sprint goal might be ruined if any member restrains from participation. Another crucial part of the development team's responsibility lies in preparing for the next. This means focusing on product backlog. The main activities are then described as: creating and refining of the backlog items, their estimating and prioritizing. All these activities are executed together with the Product Owner [1].

Each Sprint begins with Sprint planning. With the participation of the Product Owner and the assistance of the Scrum Master, the development team helps to set the goal for the next Sprint. This activity consists of choosing high-priority subset of product backlog items that are built during the period of Sprint. If the Sprint is four-weeks long, there might be need of dedicating an entire day to Sprint planning. Typically, a two-weeks long Sprint, that is very common, requires a half-day planning. The planning must happen iteratively, so the team makes a set of more certain, smaller and more detailed plans just-in-time, instead of focusing on large, uncertain, and overly detailed plans [1].

At the end of the Sprint, the development performs two inspect-and-adapt activities. Firstly, the team, Product Owner, Scrum Master, stakeholders, sponsors, customers, and interested members of other teams review the just completed features of the current Sprint. This activity is known as Sprint review. The second activity is called Sprint retrospective and it is where the whole team its processes and practices in order to improve delivery of the business value using Scrum [1].

1.6.5 Scrum Activities and Artifacts

The Scrum framework starts with the Product Owner's vision of what is supposed to be created in the upcoming Sprint (Figure 12). Since his view can be rather wide, it needs to be cut down to smaller sets of features and then included in the prioritized product backlog. This activity is also known as product grooming [1].

The first phase of the Sprint is planning, followed by Sprint execution or simply development work and ends during Sprint review and retrospective. The size of backlog is bigger than the workload executable by the team during one Sprint. During ongoing development, the product backlog contains new features, modifications to existing functionality of the product, all the bugs found during the process of quality assurance, improvements ideas for the architecture, etc. The items in the product backlog are created by Product Owner, who collects them from the team and stakeholders. The most important and valuable work must be done first. That is why at the beginning of the Sprint, during the planning phase, it is decided what is achievable by team and the rest of items stays in the product backlog. To ensure that the team achieves its determined goals, there is one more backlog created during this phase. It is called the Sprint backlog. It describes in detail in what way the team should complete the determined workload, from design, integration to testing. Sprints are always timeboxed, ranging from one week to months, but they should have fixed duration. Many teams break down the selected features to smaller task to ensure they will complete all they committed to. This is followed by an estimate of how long these tasks will take, mostly in hours [1].

The result of one Sprint is typically a potentially shippable product. Even though the result of one Sprint is referred to as a potentially shippable product, it is not typically shipped at that time. It rather means that the completed work is of good quality. Definition of done in development of software is mostly a bare-minimum of a completed part of product functionality that was designed, including integration, testing process and documentation. At the end of the Sprint, there should not be any left unfinished work on high-priority tasks that the development team committed to complete, as these tasks should always be finished first. [1].

To improve the processes in development, team then finished the Sprint with review and retrospective. The meaning of the Sprint review lies in inspection and adaptation of the product that is being built. The communication takes place between the team members and stakeholders. All the participants should be able to get the overall idea of what has been developed and help guide the upcoming development. The people outside the Scrum team have a great opportunity to sync up with the project. Scrum team, on the other hand, obtains feedback from others. Sprint retrospective is an activity that helps the team improve their Scrum practices. This is an opportunity for the whole development team,

Scrum Master and Product Owner to meet and discuss all the applied technical practices of Scrum. Team identifies and commits to a number of actions that help improve the team in the upcoming Sprint [1].

Throughout the execution of one Sprint, the development team holds a short daily Scrum, or simply a daily standup. During this brief time, team inspects and adapts to current situation. Typically, every team member stands up and upon being called on, explains what he or she is currently working on, points out to any blocking issues preventing from carrying on, and briefly explains what he or she is planning to work on after the daily Scrum is over. However, the daily Scrum is not a problem-solving activity, so any raised issues should be addressed after the daily Scrum is over with the group of interested team members. The main purpose of the daily Scrum meeting is for team to get the idea of the current state of the Scrum backlog items and communicate it. It is one of the self-organizing tools of the development team. It should not serve as a current project state update for stakeholders, but rather a planning activity for the team. The team members should be the ones talking during the daily Scrum meeting and all the other interested people should be only observers to the meeting [1].

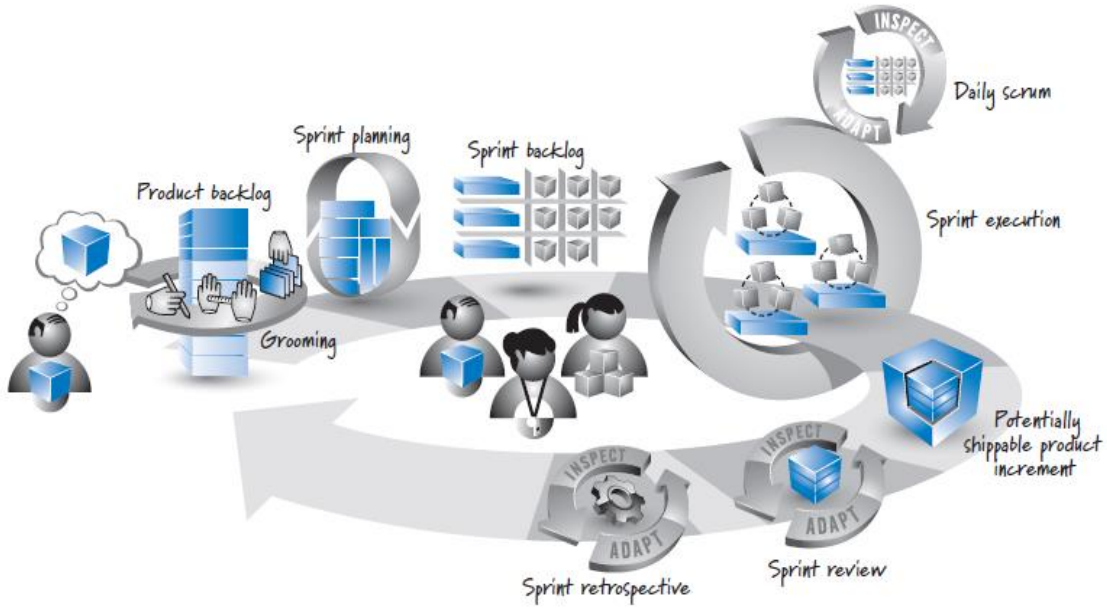


Figure 12: Scrum Framework Example

(Source: [1])

1.7 Comparison of Different Approaches

Dividing the software development community into traditionalists and agilists originates from the growing popularity of innovative approaches to software development. Each group proclaims that their methodology is the most suitable, however, these statements are often just subjective opinions and have no value while deciding what approach to adopt. A more valid point of view can be accepting that there is no such a thing as a perfect method and it is clearly very important to choose, adapt and alter available approaches to our own environment and project. This way management ensures, that the best viable option was selected for the product [14].

1.7.1 Traditional and Agile Approaches

Each model has its pros and cons, that should be evaluated while picking approach that is the most suitable. The comparison of traditional and agile attributes is described in the following figure (Table 1).

Table 1: Traditional and Agile Approaches Comparison

(Source: [14])

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

1.7.2 Agile Methods Comparison

Comparison of various agile methodologies and their typical characteristics is stated in the following figure (Table 2).

Table 2: Agile Methods Comparison

(Source: Previous Chapters)

Agile Method	Description
Crystal	A group of methods for teams with different sizes, e.g. Clear, Yellow, Orange, Red, Blue. Crystal Clear focuses on the communication between small teams that develop non-critical SW. Development has these characteristics: frequent delivery, reflexive improvement, osmotic communication, personal safety, concentration, easy access to expert users and requirements for technical environment.
Dynamic SW Development Method (DSDM)	Divides the project into three stages: pre-project, project life cycle and post-project. Principles are: user involvement, empowering the project team, frequent delivery, approaching current needs of the business, iterative and incremental development, allows reversing the changes, high-end goal is created before the start of the project, testing during the life cycle, efficient communication.
Feature-driven Development (FDD)	This is a combination of the model-driven and agile development. It emphasizes the initial object model, work division into features and iterative design of each feature. Claims to be the best suited for critical system development. An iteration of a feature has two stages: design and development.
Scrum	Focuses on project management, for situations where initial planning is difficult, with mechanisms for empiric process control. The main element are feedback-loops. SW is developed by self-organizing teams. Iterations start with planning and end with assessment. One member is responsible with solving issues that prevent the team working efficiency.
XP	Concentrates on the best development practices. Consists of twelve stages and elements: planning game, small launches, metaphor, simple planning, testing, refactoring, peer programming, collective ownership, continuous integration, 40-hour week, on-site clients and coding standards.

2 ANALYSIS OF CONTEMPORARY SITUATION

In the following part, the current state of the project is described, as well as the company introduction. This part also includes several analyses that are ought to lead to better understanding of the current situation and discover all the opportunities for improvement.

2.1 SDE Software Solutions s.r.o

The company SDE Software Solutions s.r.o was founded in 1995 in Brno by Jeffrey Kent Smith and originally consisted of a few engineers skilled in the software industry. It has currently two labs, in Brno and Ostrava, with about 100 employees. Brno lab is located at Dornych 678/90, Komárov, 617 00 Brno [7].

The name SDE comes from the original name of the company Software Development Europe, which was later modified to current name. Apart from that, letters from SDE are supposed to bring attention to security, dependability and excellence, that are the key attributes of the company's efforts [9].

To ensure security of delivered products, there is need to focus on procedures such as a multi-factor authentication, fire detection and prevention systems, power backups, proactive intrusion detections, hardware health, network integrity scan etc. Dependability is demonstrated by developers who have been working with the customers for many years, successfully delivering needed quality. Engineers are often hired from local universities and partner companies. SDE Software Solutions s.r.o always retains extra developers to help grow existing projects [9].

Excellence is present in creating high quality software solutions for customers, having strong project management teams with international experience and having a local presence at both EU and US campuses [9].

2.1.1 Services

SDE Software Solutions s.r.o focuses on development of web and mobile applications, lab and product virtualization, database design and testing activities including manual testing, automation, and mocking environments. Selected technologies are listed in the following table (Table 3).

Table 3: Selected Technologies Used in SDE Software Solutions s.r.o.

(Source: [9])

Product Domain	Technologies
Web Applications	Java, .NET, Angular, React, Bootstrap, Ruby, PHP
Mobile Applications	Native, Cordova (Phonegap), Ionic
Lab and Product Virtualization	VMware, KVM, Oracle VM
High Speed, Linux/Unix Processes	C/C++, RTOS
Database Design	MongoDB, Oracle, PostgreSQL, MS SQL
Test Automations	Mock Frameworks, Selenium, Scripting Languages

2.1.2 Company Mission and Vision

Company's mission is to be an innovative service-oriented company, that exists to develop and deliver software solutions that contribute substantially to their customers' success [9].

Vision of the company is to make developing software products for their customers a pleasure with highly intelligent, quality focused and English-speaking software engineers [9].

2.1.3 Company policy

Despite the company's sustainable growth, the local management team in Brno focuses on creating a small company atmosphere by valuing the individuals on the team [8].

Company is using a co-sourcing model when it comes to collaboration with other companies. It is a type of strategic partnership where internal and external resources are combined in order to achieve the shared long-term goals. Opposed to traditional outsourcing, co-sourcing relies on increased transparency, clarity, better control over processes and focuses on partnering rather than vending [9].

SDE Software Solutions s.r.o puts effort into broadening current software resources and know-how of specific technologies, tools and processes, so that product delivery remains timely and affordable [9].

Teams in this company are empowered to build trust with costumers at every step. This requires close work with the clients to ensure that all business communications and business requirements are clearly met and addressed. One of the main missions is to provide consultative technical expertise and software development to companies at a fair and reasonable rate. At the same time, there is an importance of doing this without sacrificing quality. Delivering products and results is done through a short-release development cycle. Customers of SDE Software Solutions s.r.o are Medfusion, FoodLogiQ, Oracle, ARCA etc. [9].

2.2 Project ABC

There are many active projects in SDE Software Solutions s.r.o, both internal and co-sourced. Project ABC is one of the co-sourced projects, with one team consisting of team members from both Brno and USA. The company from USA has established business over the globe and has been working with SDE Software Solutions s.r.o for over a year. Project ABC is focused on delivering and maintaining several products, since the company has many different customers with diverse needs. Hence, there are not only many products being developed, but as well as that many versions of one product are created in order to satisfy the demand. The main purpose of the collaboration with SDE Software Solutions s.r.o is to ensure the quality and cost-effectiveness of delivery, based on trust and the past experience of other customers of SDE Software Solutions s.r.o.

2.2.1 Team Structure

Team Force was formed in 2017 as a result of forming a collaboration between the USA company and SDE Software Solutions s.r.o. It has 13 members, as we can see in the Figure 13. For putting an emphasis on the importance of the team being a unified group of members with their unique and equal contributions, the current occupation (USA or Brno) of team members will not be disclosed in this thesis.

Currently, the team Force has 4 developers. Their main roles are writing the source code, maintaining it, doing code reviews for other developers, and solving system

integrity problems. They have their own Development (DEV) Lead, who ensures that all their work is compliant with current plans and requirements.

Testing of products, maintaining, and creating test plans and estimates are everyday activities of the Quality Assurance (QA) part of the team, made of 4 testers. QA has also its own supervisor, the QA Lead, who makes sure that testers deliver results on time and according to compliance with the demanded workload.

Team Force possesses of one Architecture Engineer and one System Integration Engineer, who provide the necessary expertise in many crucial domains of software development, such as architecture of the system and integration of the existing system with other parts or creating builds of the software.

Project ABC has its own Product Owner, whose main objective is to communicate with the customers, and interpret their needs to the team.

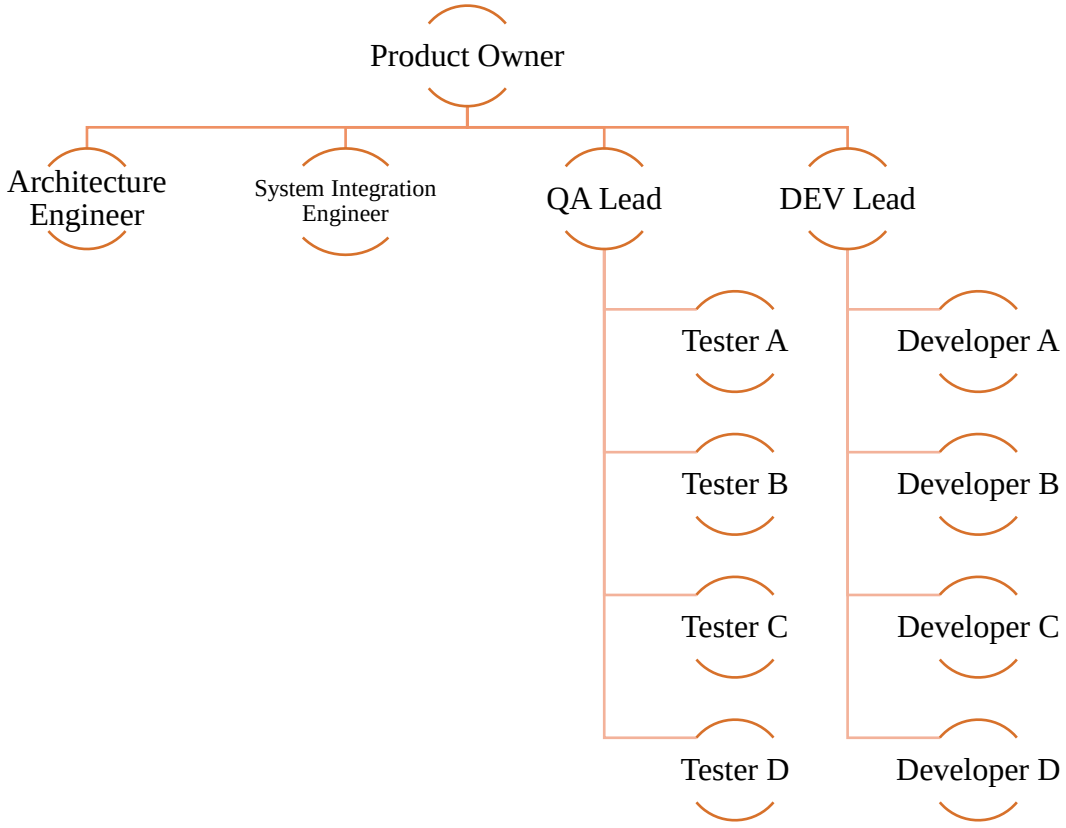


Figure 13: Team Force Structure

(Source: Own Creation)

2.2.2 Team Culture and Shared Values

When it comes to the strategy applied in the Project ABC, the most important thing is the customer satisfaction. Team Force was formed with the intention of selecting the most suitable engineers with different skills and knowledge and delivering the best possible solution to the customers of the company from the USA. The strategy is to be the number one when it comes to providing customers with high-quality and niche product, always adapting to customers' needs and guaranteeing 24/7 customer service. The company from USA has developed a considerable number of loyal and renown customers and it is understandable, that they always place the customer at the first place when it comes to prioritizing work. To show the customers, that they care, they dwell on professionalism when it comes to communication, on-time delivery, quality of the product, expertise in the customer service, innovation, continuous improvement and using the latest and most secure technologies on the market.

These aspects are strongly present in the way that team Force operates. While developing the product according to customer's requirements, management and team members dwell on meeting the desired specification of the product, while maintaining the integrity and security of the system. To go the extra mile, team Force also focuses on development of brand new features and functionality that customers might benefit from. Hence, team always thrives to be one step ahead from the competition. The core values in the project ABC are related to the strategy of the importance of customers' satisfaction. Team Force is built on mutual trust and honesty, as well as on effective communication and planning. Even though the team members are located both in Brno and the USA, coming from both a medium-sized company and a large company, the team culture is unified for all team members. Team culture revolves around being helpful and respective to other team members, and all team members are confident to ask questions, raise doubts or advise others. Generally, team members rate the atmosphere in team as positive and friendly.

2.2.3 Roles and Core Responsibilities

All managers and engineers in team Force are experienced specialists in their own domain. The strongest skill of developers is the ability to effectively use the vast knowledge of products, back-end and front-end development skills, many years of

experience in IT project development and processes, system architecture and system integration, programming skills in various programming languages, knowledge of many frameworks and interfaces, including tools and versioning systems.

Testers possess skills in quality assurance, testing environments setup and maintenance, testing tools and processes, creation and execution of test plans and test cases, functional testing, penetration testing, stress testing, regression testing, exploratory testing, documentation creation. Leads have vast experience in leadership and project management, as well as development and quality assurance, from their past positions. Product owner possesses strong skills in customer relationship area, detailed product knowledge, project management, etc.

All team members have more than 2 years of experience in software development, hence, they all understand the processes present in the product development, including all development phases, customer relationships, supply chain management, release process, customer support and sustaining partnership. Team members have great communication skills, they all enjoy working in this sphere and have friendly and positive attitude while dealing with hardships.

Specific activities and responsibilities of the team roles are described in detail in this part of thesis. It is important that team members always focus on the part of work that they have responsibility of, unless there is need for them to pick up somewhere else, e. g. in case of emergency.

Developers

The key role of developers lies in delivering working and reliable high-quality source code that makes up most of the product. They refer to the information system (IS) of the project to stay up-to-date with current state of the product. They are notified about new assigned work via the IS, a discussion on the daily meeting or other form of communication.

The assigned piece of work may be in a form of new feature. In this case, developer designs the solution and works on the implementation himself. However, if this new functionality is supposed to be a major change in the product and would affect the whole architecture of the system, he asks for assistance from the architecture engineer and the system integration engineer. Another type of item that developers are assigned to is a bug.

A bug can be found by QA, Product Owner, developers, and, in the worst-case scenario, by the client. An assigned developer must begin the debugging process by going through all the available information about the found issue, including logs, screenshots and descriptions provided by the person who had found it.

To ensure the quality of the source code they write, developers participate in an activity called a code review. When a developer finishes writing the code for a feature, or a fix, he then asks for a review from his fellow developers. At least two other developers inspect the written code and give a feedback to the original developer, based on the quality, reliability, security and many different attributes of the code. They may suggest enhancements of the solution or discuss creating totally different solution, that would be more suitable in that particular case, and the product as a whole would benefit from it.

Another activity they are responsible of is letting the QA know when the feature is finished, so that they can start focusing on testing this activity. Developers must keep track of all their items to ensure these items' state is always correctly shown to others, and to avoid misunderstandings in team.

Testers

The quality assurance is a crucial process in the SDLC. Testers in the team Force are split into two main categories – manual and automated testing. Manual testers manually simulate the behavior of the user interacting with the product, whereas automation testers focus on automation of the testing process, hence, writing tests in programming languages. Automation saves a lot of time and is very cost-effective, however, manual testing is required in some of the test cases.

Among all the testing activities, the most important are:

- creating test plans, test cases and scenarios,
- maintenance and revision of the testing environment,
- functional manual and automated testing,
- regression and integration testing,
- reporting issues,
- assistance to developers during the debugging process.

Testers provide necessary quality assurance to avoid shipping a faulty product to customer. That is why testers are simulating new features and their integration into the system on their own environments, using useful testing tools and looking for any odd behavior. It is their duty to keep all the testing environments up-to-date. In case any issue or bug appears, they immediately report all the collected information to the IS, providing the steps to reproduce, environments versions, screenshots or any other useful files. There are 2 manual and 2 automation testers in the team. They have a slightly different expertise when it comes to developed products and a different level of experience in testing, but all of them are able to test anything they are assigned to. When they run into something new, they consult within the QA part of the team. If even this is insufficient, and they still lack the information necessary for their work, they ask for advice from other engineers, or consult the requirements with the Product Owner.

Product Owner

The responsibility of the Product Owner lies in establishing trust-based communication between the team and the customer. Even there are many products developed in this project, it has only one single Product Owner. It is his main duty to ensure that all requirements from customers are met within deadlines and in outstanding quality.

His daily workload consists of the following activities:

- communication with team members and customers,
- reviewing product backlog and current Sprint,
- prioritizing items,
- managing deadlines.

Product owner is always present at the daily meeting in case that team needs to discuss any items from backlog or Sprint. Team informs him about the current state of work and refers to him when there is not enough information about functionality or priority. It is very important to agree on exact attributes of a work item in advance. Sometimes, developers misinterpret requirements and create a functionality that is useless or lacks attributes that customer expected. This is clearly a result of poor communication between team and Product Owner. Reviewing items in Sprint and backlog is a daily activity that the Product Owner should be able to execute before the daily meeting so that

team can decide what is next based on up-to-date information in the IS. He also makes decisions based on the state of items in backlog and Sprint, so developers and testers must update their work items as well.

QA Lead and DEV Lead

Leads are supposed to establish communication across the whole team and are directly referring to the Product Owner. They manage their group of developers or testers and serve the purpose of tutoring, supervising and supporting. Both of them are senior developers or testers, so they are able to share expertise in their domain. If testers or developers lack time or resources, their Leads must make sure this is addressed as soon as possible. They also help with estimates and coordination of activities in the project. Leads do not implement or test features, since these are activities that their subordinates carry out. Their presence at every meeting is required.

Architecture Engineer

Architecture engineer is actively participating in the product development. Before implementation, he often designs core parts of the feature or helps developers who are designing smaller features. He has a lot of experience with all the products and is able to share knowledge on all key points of the products architecture. Architecture engineer and system integration engineer are the first people that other engineers turn to in case of need of advice or any as soon as any problem arises.

System Integration Engineer

This engineer is crucial for development of working and stable products. Not only she possesses expertise about every single product developed including all the different versions, but also understands how every part of the software operates and relates to other parts. Since there were many products and versions created during the existence of this company, further development without this role would be near impossible. She revises all changes done to the core structure of every application, mainly, looking for any interferences or malpractices occurring in the solutions created by other engineers. If she happens to find any of these faults, she then guides engineers to better solutions. System integration engineer is always present at the daily meeting, and apart from that, always present during discussions with customers to evaluate all risks connected to desired functionality or refactoring process.

2.3 Information Technologies Analysis

To analyze the project environment, only systems significant for the team Force were chosen among all systems used in companies. Main systems that serve as foundations and support elements in the project are Zen IS, development environment, test environment, test management system, communication tools, and the documentation portal. To meet company's requirements for exposure level, exact names of tools and versions are not disclosed in this thesis. Human Resources (HR) and financial systems are not described in this thesis, since this is handled between the USA and Brno upper management.

2.3.1 Zen IS

Zen IS is an internal information system developed by engineers from the US company. Its main purpose is to serve as unified project management web application.

Home page contains links to all products and their versions, so that all employees can browse the latest information about project from one place. Every product has its own Sprint and product backlogs, that are typically accessed from the home page as well. The reason for developing this custom internal application above the development and versioning systems, was to cut costs on subscriptions to commercial products, such as Jira from Atlassian, and using the expertise that engineers already possess in order to create unique and well-fitted solution. When users drill down to a particular product, they can view and modify content related to backlogs, such as items, their descriptions, item states, priorities, and product information including deadlines, latest changes and versions etc. Zen IS product page structure is outlined in the following figure (Figure 14).

2.3.2 Development Environment

Development environment consists of Integrated Development Environments (IDE), versioning systems, and servers:

- development server,
- staging server,
- production server.

IDE is used for writing the code, building, testing and debugging it. It is up to engineers to choose preferred tools, even though after many years of development, engineers tend to mostly use the same IDEs and tools. Versioning system allows

engineers to keep track of software versions and they also use is as a change management and revision tool.

Figure 14: Zen IS Structure

(Source: Zen IS)

2.3.3 Testing Environment

The environment dedicated to testing activities consists of these parts:

- test management system,
- test hardware and virtual machines,
- testing tools and add-ons.

With a creation of a new product version, QA part of the team creates a test plan with test cases and scenarios that are related to the software functionality. These plans and

scenarios are stored in the test management system. Other important components of the testing environment are various testing tools and applications, that are widely used for functional testing not only by testers, but also by developers in the implementation phase or debugging process.

2.3.4 Documentation Portal

This is a website that belongs under the Zen IS. It is the first place where engineers go while looking for information related to any product, version of product, customer requirements, and any specific features and technologies used previously in the project ABC. It is supposed to be a collective storage for all knowledge, know-hows, tutorials, product documentation and requirements, as well as significant modifications in the project.

Even though it is a helpful website used mainly by new engineers, after a certain period of time it becomes an insufficient source of information when it comes to quality and extensiveness of its content. This matter is often brought up in a daily meeting by senior engineers who demand creating thorough documentation, but due to lack of interest of other team members and QA and DEV Leads, this discussion always ends by mutual agreement that never comes to action. A lot of the information is missing, mixed up, incomplete or only partly valid.

2.3.5 Communication Tools

Team Force is a co-sourced group of engineers, so it is crucial to maintain regular communication between Brno and the USA, as well as across engineer groups and management levels. Mostly, team manages to solve matters via email, Skype and during daily meetings that occur via GoToMeeting application. Other communication channels are Zen IS, mainly Sprint items descriptions and details, and communication using comments under items.

2.4 Problematics Breakdown

There are two management layers in team Force. One is the local management of the part of the team according to occupation. The second one is narrowed down to management of the project ABC. For example, Brno engineers plan their holiday and inform both Brno manager and the QA or DEV Lead, because it affects the project. On the other hand, all upper-management decisions and communication between Brno and the USA are handled mostly without the participation of the team members and they are accordingly informed about final decisions or topics for discussion. DEV and QA Leads are supervisors and managers of developers and testers, each tutoring and managing their group of engineers and reporting directly to the Product Owner. Leads manage deadlines and requirements, but don't apply restricting rules on team members, leaving them the freedom of choosing their own way of performing tasks.

However, there are clashes in the communication between Leads sometimes, due to placing the priorities of development and testing against each other, forcing the team members and Leads to lengthy discussions. The tendency of putting emphasis on only one stage of development process is present mostly at the end of the Sprint and towards any deadlines. This impacts the communication in team and tends to create hostile environment, which leads to delays in the product delivery. These delays are negatively impacting the company from USA in terms of their customer relationships.

Currently, these issues are attempted to be solved by creating temporary changes in the workflow, such as implementing agile methodologies to project, but are never enforced by management layer and hence, are discarded in the end. With the raising number of new customers, company is struggling to deliver on-time and high-quality solutions.

2.4.1 Management Styles Analysis

There is a strong presence of Waterfall model in the project ABC, since this is the model typical for the USA company management. One task begins only after the previous one has been completed, leaving, for example, testers without any tasks assigned for the large part of the Sprint. Even though testers have other matters to attend, the testing phase of the particular product is pushed away during the Sprint, creating large pressure on team

towards the end of Sprint. When QA part of the team discovers more faults and bugs than expected, delays in the product development cycle are very long and costly.

The second model of software project management, present in this project, is Scrum. Scrum was chosen as a support for the current project management model and was supposed to help managing several projects at once, with sustaining delivery and high-quality solutions. However, only a few elements are drawn into the SDLC and application of these elements is not with accordance to the core Scrum principles. These elements and artifacts borrowed from Scrum and used in the project ABC are:

- planning meeting at the beginning of a new Sprint,
- Sprint (one iteration of software development),
- daily meeting (daily Scrum),
- putting emphasis on scarce documentation,
- welcoming attitude to changing requirements,
- the role of Product Owner,
- using product and Sprint backlogs.

There are several differences in usage of Scrum artifacts and elements used in the project ABC and original Scrum principles.

The planning meeting does not include all team members; hence, planning is not managed properly and additional communication via emails is required in order to spread all the information. Team members do not participate in planning activities, like estimating workload for items with story points. This is the domain of Product Owner, who prioritizes items alone according to his information from customer. Absence of developers and other engineers in this activity makes the development slow down every time there is an issue with architecture and system integration of a planned feature. For example, there are other parts of the application that need to be implemented in advance and there is not enough time to finish the planned feature because of the decision of the Product Owner. After the matter is brought up by team, Product Owner reprioritizes Sprint items once again or moves a part of the workload back to the backlog.

Duration of one Sprint in the project ABC is not fixed. However, this is because customers prefer waiting longer for desired functionality from receiving unfinished part

of software on time. Even though team bears in mind that delivering past the deadline is not the best practice, they partly count on having this possibility.

Project ABC daily meetings have a different purpose than the described and used in typical Scrum environment. Detailed information and main drawbacks are described in the chapter Meetings and Calls.

Scarce documentation in the project ABC is not an advantage, as described in Scrum principles, but rather a disadvantage. Even though it is not healthy to have vast documentation without actually focusing on the process, many engineers in the project lack the time to write down all necessary guides and knowledge and this results in missing information later on. All team members and managers know they would benefit from this activity, however, no one participates in it. A lot of time is consumed later, when team members are forced to go through processes that had already been executed in the past but without creating documentation. This problem is caused mainly by the massiveness of the whole project, developing several applications and its versions at once.

Welcoming attitude to changing requirements is a must in software development projects. and project ABC is no different. This attitude is a strong advantage of the project ABC, and its customers strongly rely on that.

The role of Product Owner in project ABC is very similar to the one described in Scrum methodology. He understands the needs of his customers and interprets these as requirements to the rest of the team. On the other hand, team members would like to be given more chances to participate in estimating duration and complexity of tasks in the Sprint as well as during the planning phase. Also, due to the number of products and versions he must manage, he does not always get the prioritizing task done on time.

Product backlog in project ABC contains all previously created items that are supposed to be implemented in the future. Product owner orders and manages items, selecting the ones that are currently needed to be implemented and moving them to the current Sprint backlog. However, some items, like bugs are reported to the product backlog and some are reported straight to current Sprint backlog. This sometimes creates confusion and misunderstandings between team members. The rules for creating items and placing them in the correct place are not enforced by Leads nor the Product Owner,

nor they are strictly written down in the Documentation Portal, so no one focuses on abiding them.

Sprint backlog contains all items that team members work on at the moment and they are the most pressing ones according to Product Owner. Many times, team does not focus on execution of tasks in one Sprint backlog but in many Sprint and product backlogs for different products and versions of the software at the same time. This is very difficult for team members and Leads, as well as for the Product Owner to keep track of.

2.4.2 Sprint workflow

In this project, all the products developed have their own current and past Sprint backlogs and backlogs. Backlog contains all the features, bugs, improvements planned to be done during the existence of the product. Current Sprint backlog contains all the work chosen to be worked on in the following period and past Sprints are sets of finished items.

Sprint is one iteration of the development of the product. However, Sprints in this project do not have a fixed length and can range from 1 week to 2 months. Sprint does not have strict rules, and anything can be added or removed in the meantime.

New Sprint is announced when the Product Owner decides that team needs to move from the current Sprint. Testers are often stuck in a seemingly finished Sprint while developers are already in a different Sprint. This leads to confusion about the current state of the work and complicates the process of planning. It is very common to have unfinished and active items in a Sprint that is supposed to be closed. Often, team members participate on development of more than one product, hence, must manage work between several Sprints and even backlogs. Backlogs, despite the recommendation, often contain active items. This is because of last-minute changes of requirements by customers or misunderstanding of planned features.

The position of one item throughout the iteration of one Sprint, is described in the EPC diagram (Figure 15).

RACI Matrix

RACI matrix helps to understand core responsibilities on the team members, accountability, information flow and consulting flow (Table 4).

Table 4: RACI Matrix

(Source: Own Creation)

Activity	Team	Engineer A	Engineer B	Tester	Product Owner
Assigning the item	I	R			
Requirements analysis and revision		R			A
Implementation	I	R			
Code review		I	R		
Item in QA				R	I
Revision of requirements					R

Planning Phase

Before the planning meeting takes place, the Product Owner must sit down with a customer with the vision forming a final set of requirements for the upcoming Sprint. However, these requirements often change during the Sprint, so team needs at least a rough version of requirements to begin the planning process.

After this meeting, team can arrange a planning meeting. Planning meeting involves the USA part of the team, as stated earlier. Team members review the requirements and create a set of new items or choose existing items from the backlog and discuss them.

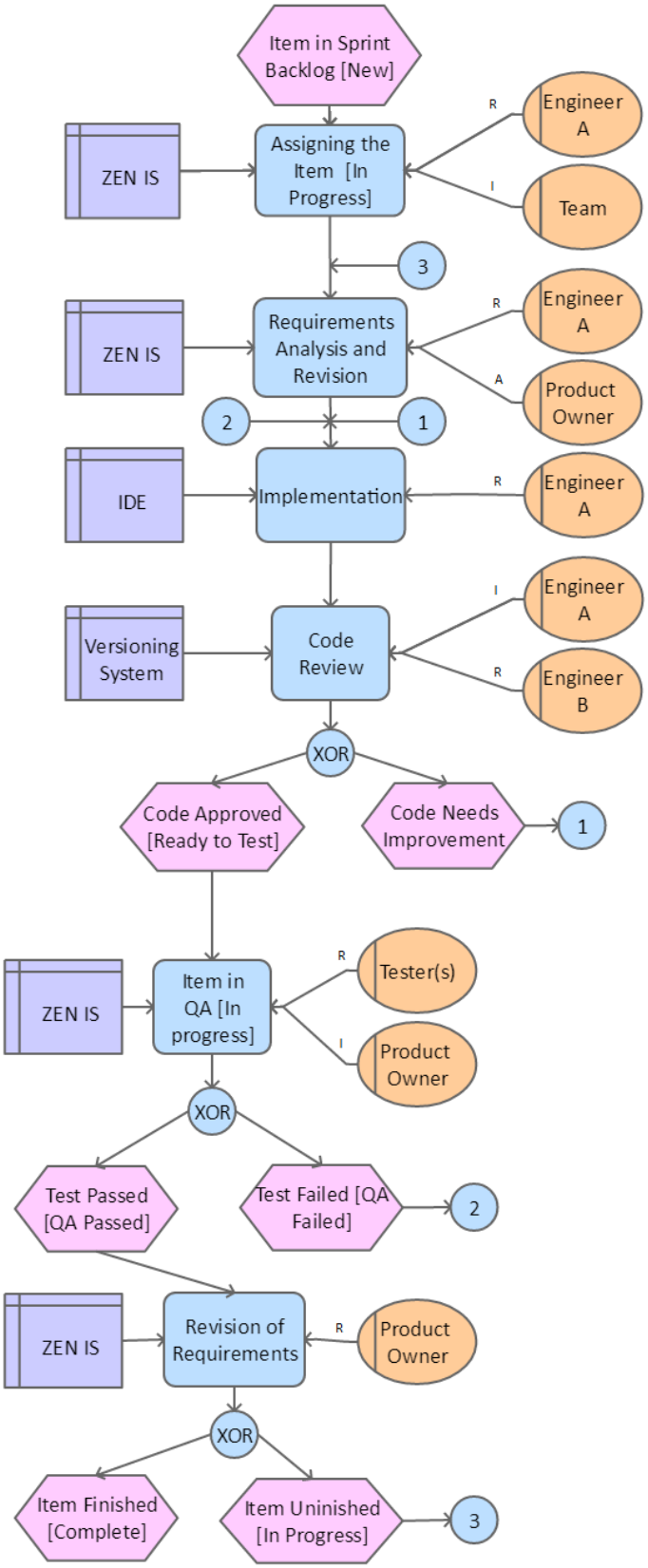


Figure 15: Sprint Item Workflow EPC Diagram

(Source: Own Creation)

Team does not use any official method to rate the difficulty or estimate the process of development for items, rather they verbally describe every item, discussing the workload needed. These mutual estimates are not described in the IS, so it is difficult for e. g. testers, to remember the extensiveness of the feature when they get down to the testing process. This leads to misunderstandings in terms of what is the part of the testing process and what testing activity is already over the top or meaningless.

Implementation Phase

After the official Sprint backlog is formed, it is up to developers to choose what they prefer to start working on. The only exception is toward the end of the Sprint, when the order of performed tasks is often set by the Product Owner, to ensure that customers will get the necessary minimum of their requirements implemented on time. Items in the Sprint backlog can be in one of many states. The first state of an item is New. Any item that was added by a team member is in this state. When developers or engineers start working in this item, they change the state to In Progress. This is because the whole team, mainly the Product Owner, needs to keep track of the current state of the whole Sprint backlog. Firstly, developers need to go through all the requirements for desired functionality and in case there is any missing piece of information or unclear conditions, they need to meet with the Product Owner to discuss the matter. After the implementation of several items is finished, and other engineers positively rated the completed code, owner of this item changes the state to Ready to Test. Right before handing it over to testers like this, one of the engineers produces a working build to be tested on. He or she selects all items set to the Ready for Build state and produces the build. If items are not correctly labeled, they are omitted from the build and this causes several arguments between the team members and Leads.

Testing Phase

Testing activities related to the active Sprint begin as soon as there are any Ready to Test items in the Sprint backlog. Testers can assign these items to themselves and begin testing the functionality on their testing environments. These environments need to be prepared in advance to ensure that everything is complete on time. To keep the environments up-to-date, testers need to possess all parts of the system in the newest

version. However, this part of the process is often slowed down due to volatile state of requirements. System integration engineer often gets caught up in the process of producing many versions of the same part of the application, before the final one is agreed on. Hence, preparing environments for testing purposes is often delayed, which prolongs the whole Sprint duration.

With a working and updated testing environment, QA part of the team tests all the Ready to Test items and sets their state accordingly. If the completed item passed all the automated and manual testing, they set this item to be QA Passed. On the other hand, if there is any problem with the way that new feature behaves or integrates with other parts of the software, QA sets it to QA Failed and assigns the item to the developer who worked on the implementation phase. QA Passed items are then changed to Complete by the Product Owner.

Revision and control activities

These activities are executed throughout the whole duration of the current Sprint. These are mainly:

- daily meetings,
- reprioritizing of the Sprint backlog by the Product Owner,
- restructuring items requirements,
- code reviews,
- retesting by QA,
- functional regression of the product,
- controlling activities performed by DEV and QA Leads,
- occasional creating of documentation.

Release Phase

Release phase is a term used to describe the phase near the end of the Sprint, when a final build, called also Alpha build, is produced and passed the testing. Team members have successfully created a shippable product that can be tested on beta customers. After the build passes even this phase, it can be deployed at the customer. There is sometimes a Demo meeting occurring in this period of time, where several teams and managers gather to observe the demonstration of brand new software features and give feedback to

team Force. This is the favorite part of the Sprint for many team members, because they can enjoy the results of their work and feedback from their colleagues. Feedback from customers goes back to Product Owner, who then shares the information with the rest of the team, mostly during daily meetings. There is currently no meeting dedicated solely to feedback analysis and future improvements discussions. Most of these discussions take place during daily meeting, affecting negatively the time management of the team members, since these discussions tend to go over the estimated meeting duration.

Meetings and calls

There are currently recurring daily meetings held for the whole team and a few other types of meetings with different purposes. Namely, team members and other interested parties meet at:

- daily meetings,
- planning meetings,
- emergency calls,
- troubleshooting meetings with customers,
- regular calls with customers.

The main purpose of the **daily meeting** is for team Force to group up at one place physically or at least via virtual meeting, using application GoToMeeting. Attendees are team members and other employees who are needed for discussion. Daily meeting begins with one of the team members who asks everyone in the room to say a few words about what is blocking their work and what they work or plan to work at.

This project has no Scrum Master, so the person coordinating the meeting is sometimes a developer, sometimes the Product Owner or lead. This is a little confusing when at the beginning of the meeting everyone is quiet and waits for others to start. Also, without a dedicated person to lead the meeting, team often becomes passionate about one single problem or a blocker and tends to completely change the purpose of the meeting. This takes time of all the people not directly connected to the discussed issue. Daily meeting is also used for sharing various news about customers or other employees of the company. This leads to another time consumption. Daily meeting is often mistaken for knowledge transfer meeting and team members once again get caught in a lengthy conversation about implementation details or other technology related topic. To get the

idea of the current state of the work, Product Owner often shares an on-going Sprint or backlog on screen for all team members to see it. This leads to discussion about various work items and the state they are in. Since testers and developers often forget to keep up-to-date state on their assigned items, not everyone is able to get the idea of current progress. Fairly often, test items that have already been implemented and are included in the latest build, waiting to get tested, are stuck in the Sprint in Ready to Build state. Developers often forget about this and do not change the state to Ready to Test. This means that testers do not get to the testing process and are blamed for not doing their part of work at the meeting. On the other hand, testers tend to forget to change the state of tested items to either QA Passed or QA Failed, once again leading to prolonging the work in the current Sprint and misunderstandings at the meeting. Another subject of discussion is the version of the product team speaks about. It is common that after 20 minutes of conversation, someone finds out that team is speaking about totally different version or even a different product. What is supposed to be a daily meeting, in its Scrum form, often turns into half an hour or longer discussion about several topics. After everyone in team shares their progress and blockers, the currently leading person asks if that is all to be discussed and dismisses team. This meeting occurs in the morning for the USA part of the team and in the afternoon for the Brno part.

Planning meeting is exclusive for the USA part of the team and no one from Brno is invited. Since this practice is currently blocking the Brno part from actually participating in planning the future work, a lot of seemingly obvious decisions appear confusing to engineers from Brno.

The only planning activity that Brno engineers participate in, is occasional emails about latest decisions. As they are the last part of the team to find out about them, they often have to stop their WIP and start working at different tasks.

Emergency meeting or call is not a regular event, but rather a sporadic meetup within a small group of engineers and managers with the main goal of solving immediate problems and critical issues. These meetings occur anytime there is a huge probability of failure when it comes to product delivery, either on time or in required quality and extent. Another reason to hold this meeting is any problem that appears at the customer and is not able to be solved easily with a help of customer service of simple fix by developers.

Emergency calls help in prevention of losing valuable customers or losing their trust when it comes to delivery. Anyone who can give useful insights about the discussed topic is required to attend this meeting.

Attendees of the **troubleshooting meeting with a customer** are mostly developers and other engineers who must break down the problem that occurs at customer. This is a direct communication between the customer and engineers with the goal of solving critical issues and preventing their future occurrence.

Regular call with a customer is held only between a Product Owner and a customer. No one else is required to attend, since the conversation is about customers satisfaction with the current progress and the latest version of the product. They can also express their concerns and ideas for future development. After this meeting, Product Owner is able to inform all team members about news and update the IS with accordance to customer's plans and needs.

Items in Sprint

Apart from the work in all phases of the software development, engineers and other team members create items anytime they need to file in a new feature, bug, architecture change, functionality, refactoring or cosmetic change, etc. (Figure 16). Mostly, these items do not have any descriptions, as they are created during daily standup meeting and no one gets back to them afterwards. When this item comes to an attention during Sprint, it is difficult to track down the trail of thoughts that lead to this item. With the item name solely, engineers are not able to determine all requirements and key properties of the functionality, and testers are not able to test the item thoroughly. When the Product Owner goes through the Sprint, he is not able to prioritize an item that lacks a description, hence, he must discuss this matter with team members once again.

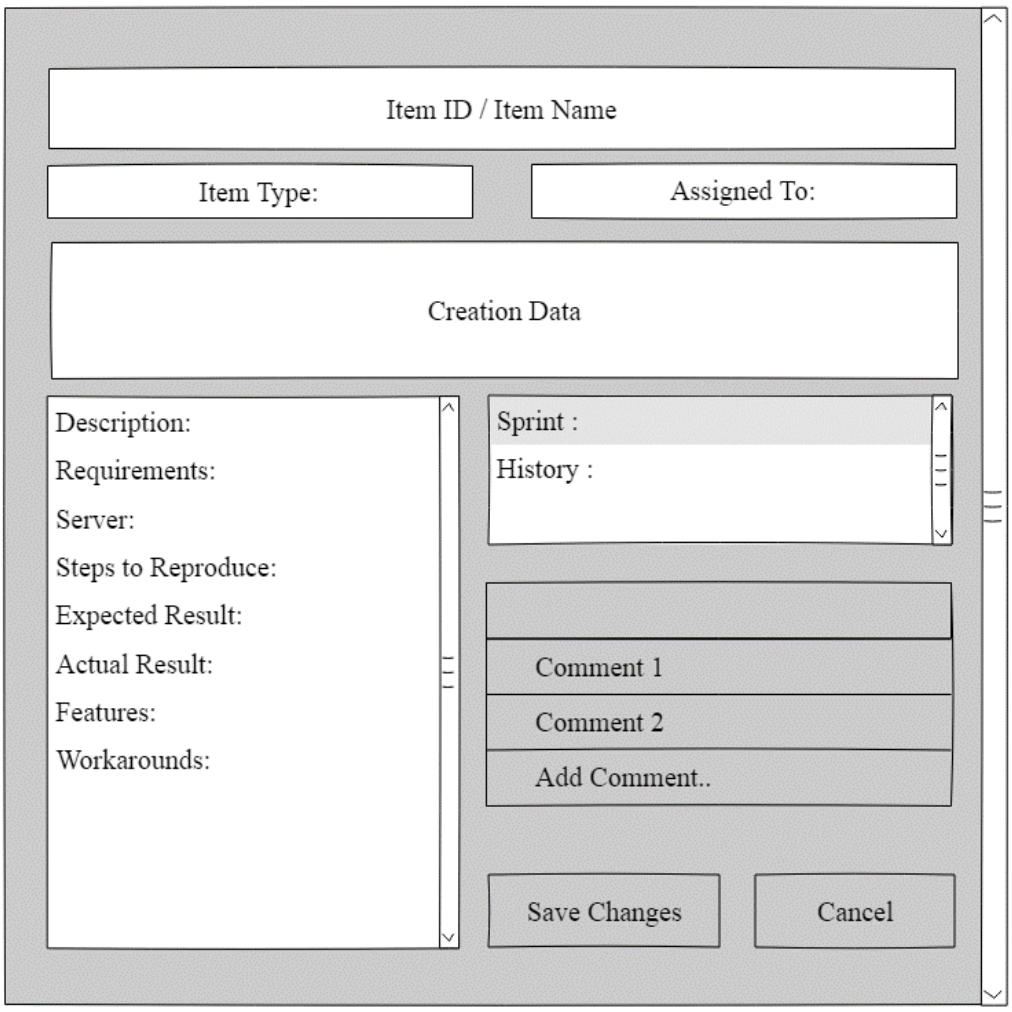


Figure 16: Item Creation Example

(Source: Zen IS)

2.4.3 Cynefin Framework

Among all five domains, project ABC stands in the complex domain. The development environment is rather unpredictable. This is the domain of emergence, and in a lot of cases, the solution is not apparent at the first glance. Team members usually need to investigate thoroughly and inspect the environment. After that, team decides what is the best solution in this situation and adapts. Since efficient interaction and communication are essential for this domain, there is definitely an opportunity to enhance these skills at the moment. This project can be described as enhancing existing products with new innovative features. As mentioned in the theoretical background of this thesis, Scrum is well suited for the complex domain, since is need for innovative and creative approach to matters. The project is not in the disorder domain, which is an advantage.

2.4.4 SWOT Analysis

After collecting all necessary analytical data, the current state of the project ABC can be analyzed with SWOT analysis (Table). This will help in conclusion to all strong and weak spots of the project and pinpoint all opportunities and threats. This summary is based on all preceding analysis.

Table 5: SWOT Analysis of Project ABC

(Source: Own Creation)

Strengths	Weaknesses
Expert knowledge of specific technologies Team members' effort to go the extra-mile to fulfil all customers' needs Critical thinking and problem solving Innovative and niche products development	The lack of the Scrum Master role Vague definition of team members responsibilities and process flow Over-communication instead of emphasizing the necessary level of documentation Poor documentation Unclear purpose of the daily meeting Vague deadlines, progress of work and definition of done Excessive workload towards the deadline instead of sustaining pace throughout the Sprint Omitting the use of helpful tools
Opportunities	**Threats**
Expanding to other continents and markets that would benefit from the technology Referrals from loyal and satisfied customers and partners to improve the position in industry Increasing interest in the industry Implementations of cloud solutions	Increasing competition in the industry Economic crisis Inflation Ending the partnership between Brno and USA due to currency fluctuations Legislative restrictions in terms of security (GDPR)

2.5 Analysis Summary

Even though project ABC is a successful cooperation between the engineers from Brno and USA, there is a significant gap between the ideal and the current state of the project. With increasing number of new customers and delivered products, project struggles to keep up with requirements of products and sustain effective management of software development. Among the negative impacts, delayed delivery, losing customers due to delays and misinterpreted requirements, and costly development due to ineffective management are the most crucial ones.

Leads and managers fully understand that there is plenty space for improvement and refinery of the software development process and they also attempted to fix it by implementing Scrum techniques to the workflow. However, this decision has not been completely brought to action and its poor execution lead to more confusion than solutions. Agile elements and artifacts, as well as core principles are not being used to the fullest potential nor with the accordance to the recommended usage, or best practices. On the other hand, Cynefin framework clearly states that the project, being in the complex domain, could benefit from adopting Scrum techniques. Improvement in this sphere is unfortunately always put at the second place, after the product development and delivery in fear of losing time and profit.

All preceding analysis show the lack of quality in usage or execution of the following fields:

- team communication,
- usage of meetings,
- process workflow,
- definition of progress and done,
- planning and execution of the plan,
- deadlines and priorities,
- level of documentation,
- supporting tools and applications,
- handling massive workloads.

3 PROPOSAL OF SOLUTION

The following part of the thesis focuses on a solution created in order to improve the software development management of the project ABC. Proposal of changes is built on preceding analysis of the current state of the project.

3.1 Implementation of Agile Methodologies in the Project ABC

After evaluation of the current state, proposed solution is to adapt several core principles and artifacts of the agile methodology Scrum while focusing on unique needs of this particular project. Implementing Scrum is a result of Cynefin framework recommendation about projects that are in the complex domain. Apart from this reason, everchanging requirements from customers are impossible to fulfil using solely Waterfall approach, because it is not built to handle significant changes in requirements. Customers using products developed in the project ABC are often changing requirements in the middle of the development process and want to know what the state of their product currently is. This iterative way of delivery and the strong presence of customers are typical traits of agile methodologies; hence, the project ABC would benefit from their usage.

For clarification, this adoption is not about recreating the whole project to fully Scrum-using project, rather it is a proposal to select the most suitable parts of Scrum that project ABC would benefit from. Project ABC is a complex set of several products and versions, being delivered to different customers with unique needs. Managing this project solely with the usage of Scrum would not be suitable in this case, since this methodology tends to neglect documentation, and delivery planning tends to be blurred in terms of concrete dates. There are many advantages of using Waterfall model in a project of this complexity and implementing other methodologies elements, such as from Scrum or Kanban are supposed to support existing components and relations, not completely replace them.

Proposed solution is a combination of the currently working and effective parts of the software development process and new or modified principles and elements from agile methodologies that are supposed to help in terms of effective delivery and flexibility of the project.

Implementation of the Agile methodology Scrum into the project ABC is split into the following activities:

- integration of Scrum Master into the project ABC,
- Scrum training for all team members,
- redefinition of roles and responsibilities,
- Sprint rework,
- continuous measurement and improvement.

3.2 Integration of Scrum Master into the Project ABC

Based on results of the analysis, implementation of Scrum elements into project in the past has not been executed to its fullest potential, and one of the key elements omitted from the process has been the role of Scrum Master. The first step in the process of change is the integration of the Scrum Master role into the team. Proposal of creating a Scrum Master role in the team Force is a result of current hardships in effective communication of team members throughout Sprints. Even though there are roles of QA Lead and DEV Lead in team Force, they do not fulfil the purpose of Scrum Master in team.

Scrum Master will help team members embrace Scrum values and use them accordingly in their unique project environment. Shaping of usage of Scrum principles and core ideas is very important in a project that embraces several software development management models and is necessary for creating the most suitable environment for the project. Scrum Master as a role in team Force helps to understand the importance of Scrum artifacts, team roles, communication flow and agile principles and teaches how to use them to their advantage.

Another area in the project that will benefit from the presence of the Scrum Master role is meetings organization and execution. Daily meeting, as the analysis shows, is an ineffectively used meeting with many subjects discussed, fulfilling many purposes at the same time, but not the original purpose. That is, having a meeting once a day to catch up with team and inform others about the progress of the current Sprint and current blockers. Moving the responsibility of meetings organization to Scrum Master will help all the other team members focus on their workload and the confusion in terms of leading meetings will be removed.

Scrum Master is usually a part-time job and appears to be the most suitable solution even in this situation. Even though he or she will manage one team that develops several products and several versions of the products at the same time, all Scrum activities will intersect all the development processes. This means mainly having more meetings and more time required for Scrum activities than a usual Scrum project has, but not to the extent that would require a full-time position.

Recommended location of the Scrum Master is the USA, even though this role will be present for both Brno and USA part of the team. Scrum Master will have to adapt to the time-zone difference on both parts of the team and adjust meetings and communication accordingly. The reason for choosing the Scrum Master in USA is because the majority of the team is located there.

3.2.1 The Role of the Scrum Master in Treatment of Problematic Situations

With the addition of a new role in team, the Scrum Master, team members will have to learn about his or her responsibilities and daily activities, so that they can ask Scrum Master for help in suitable situations and use his or her presence in team effectively. In case of missing information or uncertainty about next steps, team members will inform their Lead, Scrum Master and altogether with Product Owner they will work on solving these problems.

The role of QA and DEV Lead in team will not change, except for the communication aspect. Any issue will be solved with the help of Scrum Master and with participation of all team members in a meeting, where they will discuss these matters and come to a suitable solution. Whereas Leads will represent leaders in development and quality assurance, Scrum Master will be considered a leader as well, but mainly in communication, problem-solving support, guidance in using Scrum artifacts and core principles. Team members will seek advice from Scrum Master anytime they run into communication clash or anytime they are unsure about their next progress in the workflow. Instead of solving the issues for them, he or she will guide them while they solve it on their own. Typical example of this form of help is when QA Lead or DEV Lead are putting development and testing against each other in terms of priority, creating pressure on team members and causing misunderstandings across the team. Scrum Master can help solve this by helping team members discuss the matter and come to solution.

3.2.2 Core Responsibilities of the Scrum Master

Among other spheres of work, or other areas where Scrum Master might be asked to help, he or she will be mainly responsible for the following activities:

- pursuing the understanding of Scrum principles and core values,
- helping other team members use the Scrum principles and core values effectively and with the accordance to the project's specific environment,
- organization and running of team meetings, enforcing focusing on the purpose of the meeting,
- communication and problem-solving guidance for team members,
- interpreting team members' ideas or needs to Leads, Product Owner and managers,
- helping Product Owner manage the product and Sprint backlogs,
- enforcing the continuous improvement in the software development process,
- helping team members embrace and adapt to changes or improvements,
- removal of impediments and barriers that block team members from work.

3.2.3 Scrum Master Adaptation

As a Scrum Master will be a new addition to the team, he or she will have to observe the current project environment at first, to grasp all the project specific aspects and problems. This observation will be held for 2 weeks, when a team will continue with the previous course of work with the passive attendance from Scrum Master. After mapping of the environment, Scrum Master will become an official part of the team Force.

3.3 Scrum Training

To ensure that all team members have the basic knowledge of Scrum core principles and artifacts, all team members will attend a Scrum training. Brno part of the team has already undergone a Scrum training, as all employees of the SDE Software Solutions s.r.o are encouraged to attend this training as one of the benefits provided by the company.

8 members of the USA part of the team will attend a training under the guidance of the new Scrum Master. Training will be held for two days. During the first day, team members will learn about the basics of Scrum and the second day will follow with a practical workshop Lego for Scrum. This activity is a lightweight way to adopt Scrum

practices and learn about artifacts, as well as the way that a Scrum team operates. The whole training will take 8 hours, but as already mentioned, will be split up to 2 days. The cost of holding this session is included in the overall cost analysis at the end of this part of the master's thesis.

3.4 Responsibilities and Roles Redefinition

In order to incorporate the changes into the project ABC, several changes in responsibilities of the existing roles are required.

3.4.1 Product Owner

Lack of time puts pressure on Product Owner and leaves less time for breakdown of requirements for a product. Poor requirements management is currently reflected in backlog items not being described and organized completely and broadly. Other team members struggle with incomplete requirements and are unable to fully implement and test required features.

Responsibility of Product Owner in terms of requirements management will be divided between both Product Owner and Scrum Master, who will help with entering and refining these requirements in the Zen IS. This form of cooperation will be present daily between both Product Owner and Scrum Master, who will meet before the Daily Meeting to write down, edit and prioritize all necessary changes in items from Backlogs. This way, Product Owner will be focused on communication and relations with customers, while Scrum Master will help him with the task of transforming these requirements into items descriptions in the Zen IS for other team members to work with.

New Core Responsibilities of the Product Owner

The following activities are to be performed by Product Owner:

- communication with customers,
- collecting requirements, ideas from customers,
- collaboration with Scrum Master in prioritizing, editing and keeping requirements up-to-date,
- interpreting requirements to team members and technical aspects to customers,
- definition of acceptance criteria and their revision,

- presentation of new products and versions to customers,
- collaboration with stakeholders,
- economics management.

3.4.2 Team Members

Other team members, such as testers, developers, and other engineers are not going to go through many changes in their core responsibilities and tasks, however, there is need to refine a few of present rules and principles used in the project ABC.

Items Creation

The first modification is in the way they report new items to the system. It will be obligatory to fill in some of the information while reporting a bug, new feature, modification, refactoring proposal and other items. For team to effectively use the Zen IS to track progress, team members will be obliged to always fill in these information:

- scenario steps to reproduce,
- expected and actual result of these steps,
- all environment versions,
- requirements for a new functionality,
- relevant files, such as mockups, screenshots, log files, etc.

Any other significant information necessary for other team members to fully understand or debug the item, will be included as well. Product Owner and Scrum Master will be responsible for keeping the track of these items and the state they are in. If any necessary information is omitted, they will ask the creator of this item to add more details. In case this item is still too vague, or someone does not understand e. g. the description of the item, additional modifications and explanations will occur during the Grooming meeting. The purpose of the Grooming meeting is outlined in the following chapters.

Documentation

One of the biggest opportunities for improvement in the project ABC is the way that team Force handles the documentation creation and usage. With no enforcement of creating documentation, team members currently always put away this task, until the topic is no longer relevant, and it becomes forgotten.

To improve this common issue, team members will be encouraged to create documentation using the Documentation Portal. Since this system already exists, and contains outdated or incomplete data, there will be a new branch of Documentation Portal created and used for the purpose of documentation creation, sharing and revision. Team members responsible of reminding and occasional review of the documentation state and progress will be the DEV and QA Leads.

3.4.3 Work Progress Definition

Throughout the development process, many team members experienced problems with the precise state of an item they worked on, leaving them clueless about the next progress. This lead to lengthy discussions during the Daily Meeting with no clear responsibility for these issues. Proposed change lies in precise definition and dedication of the state of one item to ensure every single team member knows exactly in what state the item is a and what that means in terms of next steps in the process. The rules of setting the state of an item and updating it accordingly will be enforced by DEV and QA Leads. All these definitions and rules will be present on the Documentation Portal for all team members to have a look in case they need it.

Workload Amount

Proposed changes in workload extent are related to the way team currently handles the complexity and amount of work in one Sprint. In the past, vast number of tasks included in one Sprint caused ineffective performance of team members, misunderstandings and delays in delivery of the product. Hence, proposed changes in this sphere are mainly:

- a precise definition of a terminable Sprint,
- using meetings for items estimates and management of both Product and Sprint Backlogs,
- focus on the planning phase to pinpoint an executable amount of workload,
- continuous improvement of the estimated workload that team can handle in one Sprint using burndown charts.

WIP

Only an item that is in the current Sprint Backlog and is at the same time assigned to an engineer who is working on it can be set to the *In Progress* state. On the contrary to previous situations, items from the Product Backlog cannot be in the *In Progress* state, as they were not chosen to be executed in the current Sprint are waiting in the state *New* or *Ready* in the Product Backlog.

Definition of Done

Definition of Done helps to estimate the state of items in Sprint in terms of developed features or a product being shippable to customer. Set of items from the Sprint Backlog is *Done* after all these criteria were met:

- items' compliancy has been reviewed in terms of design requirements,
- documentation related to the functionality, user cases and test cases has been reported to the Documentation Portal,
- system integration of new features has been executed,
- items passed the testing phase performed by QA part of team,
- developers created release notes for a customer,
- there are currently no issues related to the product,
- items as a whole passed the acceptance criteria on a testing environment.

Deadlines Handling

With the experience with previous massive releases, there is a proposal for a change in the way that the Product Owner and Leads manage deadlines. This issue should be partly solved with the help of planning meetings, grooming activity and daily meetings. However, there is also need for more detailed planning further ahead. This is the domain of the Product Owner who should allocate his time accordingly and focus on planning the necessary minimum of project activities in advance for several products and releases.

3.5 Sprint Rework

Sprint as a Scrum element will be defined as one iteration of software development in project ABC with the goal of delivery of a product, its version, or a part of the solution. With more than one product being developed at the same time in this project, there is need to divide the work of team members to logic groups of tasks related to a product, or

product version. Normally, work in project ABC is planned with accordance to upcoming product delivery for particular customers. Therefore, team Force is often split up to two units, each developing a different product at the same time. In case of a massive release for one of the customers, team unites and works together in one Sprint again. In the past, this approach has been established as the best possible solution for development of various products at the same time. To execute several different Sprints in one single team has been possible thanks to the similarity of products and their versions.

Having more than one Sprint when needed will remain, but with much greater emphasis on planning in advance, than before. This planning phase will be mainly about assignment of team members to a particular Sprint and definition of the intended workload and deadlines. The purpose of focusing on planning in advance is a result of past hardships connected to working on several products at once with vaguely defined priorities and deadlines, as well as assigned engineers. Product owner will be held responsible for informing all team members about planned product-delivery, including deadlines and the most pressing matters. All this essential information will be dispensed at the planning meeting before the beginning of the Sprint(s). Planning meeting will also be the right occasion for addressing all raised questions, worries or suggestions.

3.5.1 Sprint Duration

Duration of one Sprint will no longer remain without a defined duration. It will be planned in advance based on the requirements and plans that Product Owner possesses as a result of his communication with customers. Typically, one Sprint will take 3 weeks in case of a common workload in the upcoming release. In case of a massive release consisting of complex changes in the product, such as architecture changes, refactoring or a completely new set of features, Sprint will be planned to last 3 weeks as well. One than Sprint can precede the final release. In the past, massive releases required one long Sprint without a defined deadline and that caused frustration of team members without a clear vision of the goal. Having several specified iterations with the same durations instead of one non-defined iteration will help in continuous improvement of the development process. Strict definition of the workload in advance will allow finishing set tasks on-time and in required quality. Changing workload in the middle of Sprint due to

customers' change in requirements will not be necessary, because of the suitable duration of Sprints.

3.5.2 Sprint Course

The following timeline shows the planned course of 3 Sprints flow in the project ABC, based on all proposed changes stated in this part of the master's thesis (Figure 17). After Sprint 1 (S1) starts, assigned team members execute work with the help of Zen IS Scrum Board. Throughout Sprints' course, developers create builds, chunks of features to be tested by QA part of the team. Preparation of several small builds in one Sprint will allow testers to prepare their environments in advance, without delaying the test process. Every single workday, team members meet at the Daily Meeting to discuss briefly their progress on items from the Sprint Backlog. In order to plan in advance, there is a meeting held every Sprint to catch up with the latest requirements from customers and adapt to changes. Between the end of S1 and the start of the Sprint 2 (S2), team members gather during Grooming and Planning Meetings to prepare for the next Sprint. After they estimate and select their workload for the next Sprint, they check whether all work designated for the current Sprint has been finished. With all work complete or moved to Backlog, team members end the S1 and hold a Retrospective Meeting to review their past work and effort. After that, S2 can begin and meanwhile, team prepares for another Sprint, and so on.

3.5.3 Product Backlog

Product Backlog is the origin of the items that are then selected and included in the Sprint Backlog. The quality of Backlog management will be ensured by several changes in this project. These changes will affect the way that items in the Backlog are created, described, maintained, prioritized and ranked. The responsibility of maintenance in the Product Backlog will be divided between the Scrum Master and Product Owner who will collaborate on this task. Keeping track of existing items in the Product Backlog can save time during meetings and allow team members to focus on important tasks and discussions about the product.

3.5.4 Sprint Backlog

This Backlog consists of the items selected from the Product Backlog by the Product Owner with the assistance of the Scrum Master and team members. It is a list of items chosen to be executed in the current Sprint. Assurance of the quality of the Sprint Backlog will be performed by team members in terms of items creation and modifications according to the latest requirements and investigation results.

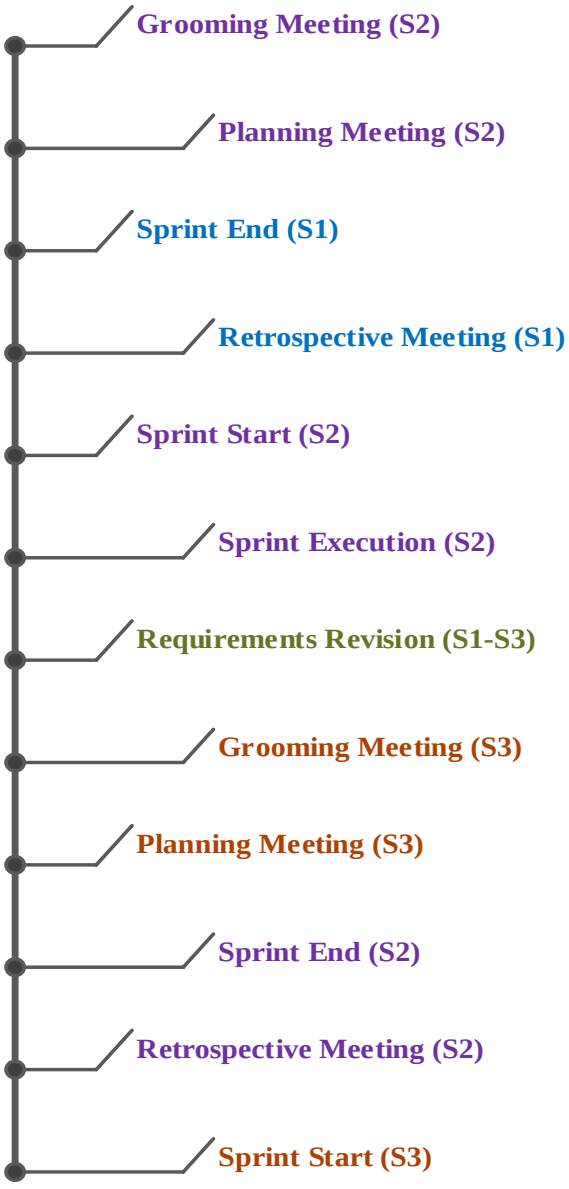


Figure 17: Section of Several Sprints Flow

(Source: Own Creation)

Another part of the team that will focus on sustainability of the Sprint Backlog will be the Product Owner and Scrum Master in terms of accordance to the needs and of customers and suggestions from engineers. Daily meetings will serve the purpose of quick progress-check in the current Sprint with the use of a Scrum Board. A Scrum Board is described in detail in a dedicated chapter.

3.5.5 Grooming Meeting

Grooming meeting will be an addition to the planning and daily meeting already used by the team Force. In the development process, a grooming meeting will take place once in the Sprint and must take place right before the Planning meeting. Backlog grooming is an activity performed by all team members. The main purpose lies in writing down, editing, refining and estimating the Product Backlog items. Adding the Grooming Meeting to each iteration of the development is a result of previous communication flows caused by incomplete, non-estimated, and poorly described items in the Backlog. What has previously been an occasional or emergency meeting with the same purpose will now become a regular activity performed by the whole team each Sprint. The duration of this new activity is yet to be decided by Scrum Master after a few Sprints will have been executed in the project.

Items Estimates Using Story Points

Estimating items is different from prioritizing them (critical, high, normal and low priority). Ranking items in terms of difficulty will allow team members better understand the amount of work connected to them and estimate the overall workload they will be able to perform in the upcoming Sprint. Among many methods used to estimate the workload of one item in the Product Backlog, team Force will adopt the method using Story Points. This method allows team members to define a workload of one typical task from the perspective of both developer engineer and QA engineer and then use this unified estimate as a referral used for ranking all the other items.

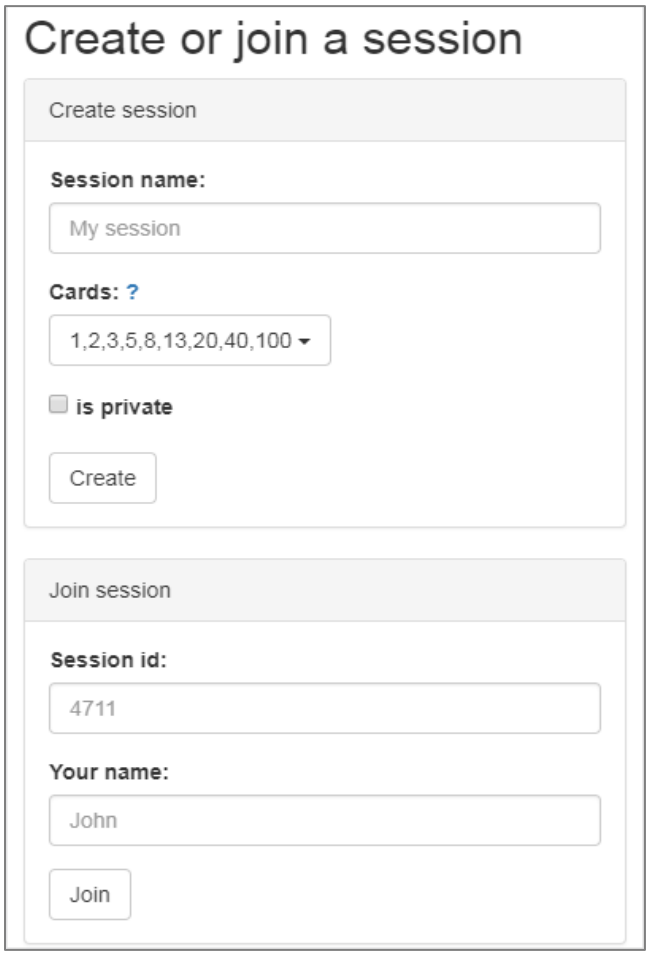


Figure 18: Scrum Poker

(Source: [16])

From the experience from other projects in SDE Software Solutions s.r.o using Scrum, one Story Point will represent a development of one simple form in the application with the implementation of the back-end, front-end and creating a unit test for this form.

From the QA point of view, one Story Point will represent a functional test of several simple forms including pre-defined scenarios and user cases. The definition of one Story Point will be confirmed at the first Grooming Meeting and entered to the Documentation Portal for every team member to access the definition whenever they need it.

Assigning Story Points is an activity performed by team members, namely developers, testers, system integration engineer, architecture engineer, DEV and QA Leads. Since team members are located in both Brno and USA, there is a need for using online version of Scrum Poker, a Scrum activity used for ranking items in the Backlog. Team members will join a session created by Scrum Master and select their best estimate

of the difficulty of the item given in the poll. An example is shown in the following figure (Figure 18). Scrum Poker is an online open source application. The session is typically initialized and shared by a Scrum Master on his or her screen and other team members access the session via their mobile devices using QR Code. Thanks to this system, team members can vote anonymously and without the impact of others' opinions [16].

Definition of Ready Checklist

Grooming requires a clear Definition of Ready for items in the Product backlog. In terms of the result of a grooming activity, an item is in the *Ready* state under these circumstances:

- all the requirements from customer related to the item are clear,
- engineers fully understand all the details about the item, such as technical criteria, user cases and processes related to the item,
- all these requirements and criteria are defined in the item description,
- all relations and dependencies to other items are clearly determined for the item,
- team members possess the abilities and resources execute the workload to complete the item,
- team members understand the purpose of the item and can demonstrate in a demo meeting.

Items in the Product Backlog with the Ready state are ready to be included in the Sprint Backlog and they are all sorted by priority. If they do not meet the criteria stated in the checklist, team members need to go through all the information again and include more details or explanations and repeat the process until every item from the Product Backlog is in the *Ready* state. Definition of Ready will help team members commit to performing workload effectively, with a clear vision of item definition, its purpose and problematic spots.

3.5.6 Planning Meeting

The planning meeting has already been adopted by team Force, but with only the USA part of the team participating in it. With the knowledge of problematic dealing with deadlines and plans for next steps throughout the duration of the Sprint, team Force should unite during the Planning Meeting with the participation of all team members. This simple

change will remove the information barrier between Brno and USA part of the team and help in understanding the course of development processes.

The goal of the Planning Meeting is to determine a suitable portion of workload for team members to execute during one Sprint. Preceding activity required for an effective Planning Meeting will be a quick revision of the items in the Product Backlog that are in the *Ready* state. Planning Meeting will always take place right after the Grooming Meeting and the main activities performed with the guidance of the Scrum Master will be:

- moving unfinished items from the previous Sprint to the current Sprint Backlog,
- asserting a new value of story points to items if needed,
- addition of Ready items from the Product Backlog to the Sprint Backlog one after another,
- quick revision of all items to ensure every team member knows what the purpose of every item is,
- continuous assignment of items to team members based on their choice,
- assignment of items with accordance to everyone's ability to undergo a specific amount of workload,
- repeating assignment of items with the highest priority from the Product Backlog until every team member is content with the amount of workload,
- revision of selected items and reassurance that all team members are committed to handle the workload,
- reminder of the next Sprint's start and end date.

As these Planning Meetings will be sometimes organized for more than one Sprint, there is need to allocate a suitable amount of time between the planning activities of each Sprint. Either Scrum Master choses two different days for Planning Meetings or gives team members sufficient break in between planning activities for two Sprints during one meeting session. Planning activity will be refined as the team will gain experience with estimates of their workload execution ability. To help team members will more accurate estimates, team Force will be using Burndown Charts to track the amount of workload successfully executed in each Sprint.

3.5.7 Daily Meeting

In contrary with the previous usage of the Daily Meeting, the main purpose of the daily meeting will be to catch up with team members, define current blockers and questions and review the overall progress in current Sprint. Participants of the Daily Meeting will remain the same, i. e. engineers, Leads, Product Owner and the addition to the team Scrum Master, as well as any other interested party.

Daily Meeting will be held once a day and will be moderated by the Scrum Master who will ask every team member to inform others about their current and planned activities, and if necessary, their blockers and questions. In case there are any pressing matters blocking a team member from working, these issues are to be discussed in a separate session with the participation of interested team members.

The presence of the Scrum Master during this meeting will prevent team members from discussions about matters that are not related to Daily Meeting. If any unrelated topic is brought up, he or she should remind team members the purpose of the Daily Meeting and help them move the discussion to a suitable meeting, call or simply an email conversation.

3.5.8 Build Delivery and Releases

Build delivery will be defined as a separate item, since creation and testing of one build is done by several team members. It will no longer have a vague deadline, but a defined date, or date range so that team members can plan their next activities accordingly. Build delivery does not necessary mean the end of the Sprint but can be done in any part of the ongoing Sprint. Build is a set of features, bugs, or changes implemented or fixed in current Sprint and are ready for the testing. It is necessary to plan a portion of items that will be implemented by developers and the portion of items to be tested by testers in the Sprint, so that all team members have their work distributed throughout the whole Sprint duration.

Release, on the other hand, will be defined as a set of builds chosen for delivery to the customer. Each release has its release notes with the list of the newest and modified features or fixes and also their descriptions. Release notes will be created by the

developers with the supervision of the Product Owner and all of them will be stored in the Documentation Portal to ensure all team members can access them in one place.

3.5.9 Continuous Improvement

Continuous improvement is one the most typical traits of agile form of delivery and it is one of the key aspects missing in the project ABC. With a denial of poorly defined processes and with the aversion towards any change, project ABC cannot benefit from a well-established and continuously improvement. One of the proposals of this thesis is to accept the necessary level of changes required to head towards improvement and establish the process of improvement as an advantage and not a nuisance. The sphere of software development known for ever-changing requirements and new ideas, hence, project ABC should be able to adapt as well. In addition to Sprints and artifacts, following elements from Scrum are supposed to help team evolve proactively and not only when it the situation is critical.

Retrospective

With the same situation as in the Grooming Meetings, distributed team Force will benefit from plenty online solutions to Retrospective, such as IdeaBoardz [17].

Retrospective is a special meeting dedicated to learning from the past decisions. Once a Sprint is at the end, team members gather to ask themselves these questions:

- What went wrong?
- What went well?
- What action items can solve the issues?

Team members fill in the information anonymously and all stated items are then discussed with others, with the guidance from Scrum Master. Team rates the quality of teamwork, pinpoints all improvement opportunities and creates the list of actions that should be established in order to solve the issues or improve processes.

Demo Meeting

This is the only meeting where team as a whole is appraised or criticized in terms of its performance, reached goals and expertise. Team gathers to demonstrate its newest product, implemented features or innovations to other colleagues or a customer. Gathered

opinions and feedback are great for team's motivation to succeed, do things better, or just keep up with at least the same quality of work.

3.5.10 Usage of Support Elements

Team Force will now focus on using more support elements provided by the Zen IS. These are mainly a Scrum Board that will replace the old list of items used previously. Secondly, team Force will use a burndown chart to measure the workload executed in the Sprint. The main advantage of this change is that team will have better control of the workload progress and will keep track of the state of the tasks.

Scrum Board

Scrum Board is a tool that will serve the purpose of previously-used Zen IS list of items forming a Sprint Backlog. Having Sprint items divided into several categories with the accordance to the state they are in will eliminate the issue of progress flow in the Sprint. The currently used list will be changed to a different view, hence, it will require one developer to make changes in the Zen IS view. The estimate of the workload is 16 hours. Scrum Board layout in Zen IS is outlined in the following image (Figure 19).

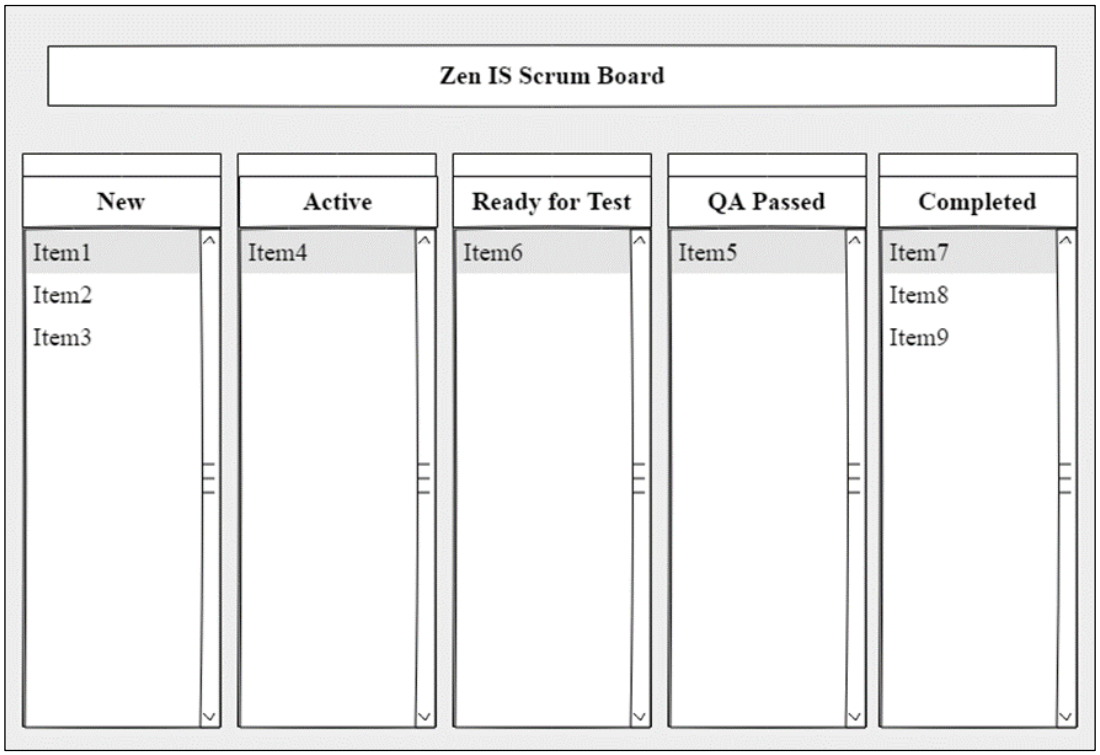


Figure 19: Scrum Board Layout in Zen IS

(Source: Own Creation)

Burndown Chart

The purpose of the burndown chart is to help team evaluate the workload they are able to execute in the course of Sprint. As team Force will progress in using Scrum, planning the workload and estimating the team's ability to execute committed work will get better. To improve the estimates over the time as well as keep track of planned workload state, team will use the burndown chart. It is a plugin already present in the Zen IS, however, team has never used it before. Example of a burndown chart created for the first Sprint is included in the following figure (Figure 20).

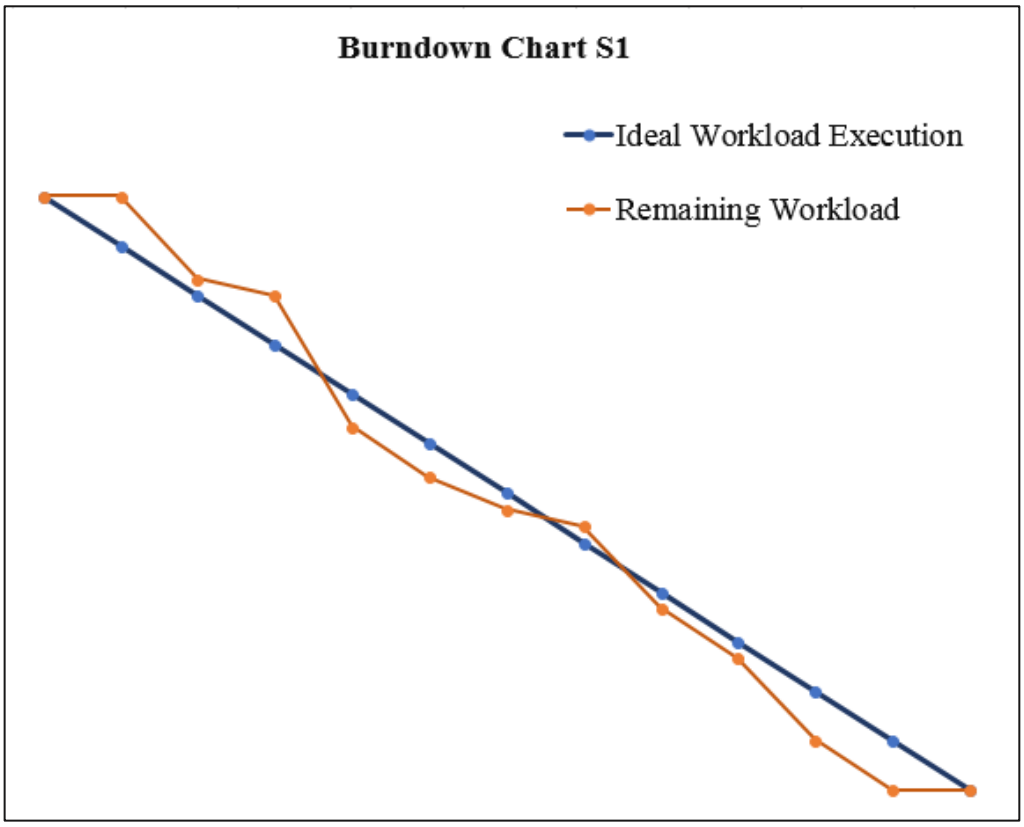


Figure 20: Sprint 1 Burndown Chart

(Source: Zen IS Plugins)

3.6 Costs Analysis

There will not be any costs for the company SDE Software Solutions s.r.o while implementing this solution. Following analysis is a summary of costs required to fulfil the proposed solution in team Force in the first month. It consists of all costs related to changes made in the project ABC (Table 6). The most expensive change will be hiring a Scrum Master. The planned salary of the Scrum Master in the USA who has at least 3 years of experience is 6500$ per month. It is assumed, that the Scrum Master will not need any soft skills trainings and already be experienced in this area. The salary of a senior developer who will implement the Scrum Board in Zen IS is calculated to 630$ per two man-days. The information has been provided by the USA company. Training of the team will be done by newly-hired Scrum Master, so there will be costs for the training of each team member in the USA. This cost has been calculated to 2600$ for 8 hours of training for 8 engineers. Overall costs for the first month of implementation of changes is 7630$. Integration of Scrum Master role to the project will be present in the following months of development, however, cost-wise only as his or her salary. Implementation and introduction of new responsibilities and artifacts cannot be precisely calculated, since these changes will be gradual. As well as that, most of these changes are aimed at the communication skills and team-related aspects of the project. Revision of the situation of the project by an external Scrum specialist that will take place 6 months after the implementation is not included in the table but is calculated to approximately 975$ based on the salaries of Scrum Masters on the market. The duration of the revision should be at least 3 days.

Table 6: Costs Analysis in Team Force

(Source: Own Creation)

Activity	**Costs for the 1st Month**
Integration of Scrum Master into the project ABC	4400$
Scrum training for all team members	2600$
Scrum Board Implementation	630$
Sum	7630$

3.7 Summary of Proposed Changes

This chapter contains a summary of specific changes proposed in the project and their unique way of implementation. The changes proposed for the project ABC based on **best practices from Scrum** are:

- new role of Scrum Master,
- Scrum training for American team members in order to teach them the artifacts and principles of Scrum,
- Scrum Master's focus on meeting organization,
- setting a strict duration of one Sprint and focus on a leaner structure of workload in one Sprint,
- creation of a Product and Sprint Backlog,
- introduction of the Grooming Meeting and estimation of every work item based on a mutual version of one Story Point,
- Product Owner's cooperation with Scrum Master while managing requirements,
- turning a list of Sprint items into a Scrum Board,
- implementation of Burndown Charts into every Sprint,
- Daily Meeting in a form of a daily catch-up with team,
- introduction of a Retrospective Meeting to execute a revision of every Sprint,
- definition of Done, WIP and Ready,
- planning meeting for all team members and planning activity organized in advance.

The changes proposed **based on the specifics of the project** are:

- several ongoing Sprints allowed in the team Force at the same time,
- greater emphasis on the documentation,
- several builds created in one Sprint,
- thorough planning process on contrary to typical Scrum's scarce planning (such as for the upcoming 3 months, etc.),
- strict team roles with defined responsibilities, because of very specific technologies used in product development.

3.8 Summary of Proposed Solution Benefits

The following benefits are the results of the successful implementation of changes in the team Force:

- Communication - Implementation of proposed changes will result in improvement of effective communication in team, thanks to redefined meetings, the presence of Scrum Master and workflow redefinition. New and redefined meetings will result in better process control and execution. Grooming and Planning Meeting will remove the misunderstandings and arguments in the team Force. Priorities will be clear, and tasks will be planned in advance.
- Process Workflow - Testers in team Force will have regular releases throughout Sprints, so they will not be left without any task. Sprints will have a set duration, and all activities will be planned advance. Clear definition of Sprint activities order will remove the confusion about the next planned steps.
- Definition of Progress - Redefinition of items state, as well as regular Daily Meetings will result in effective work execution and estimate of current progress. Definition of Done and WIP will remove delays caused by uncertainty about the current progress.
- Planning and Execution of the Plan - Planning will be handled by Scrum Master and Product Owner, making it easier to be executed. Planning meeting will be held for all members, so information will be spread across all team. Incorporating the Grooming Meeting into the Sprint will result in more efficient planning of the workload.
- Deadlines and Priorities - Deadlines set by Product Owner and enforced by Leads will be regularly controlled and will remove delays in the product delivery.
- Documentation – Huge delays in work execution will be removed by enforcing documentation creation by Leads in the team Force.
- Supporting Tools and Applications – The introduction of 2 new open source applications for the grooming activity and retrospective, will support the flow of Sprint course, with a greater emphasis on the cooperation of both parts of

the team. Transformation of the current list of items into the Scrum Board will bring better vision of the progress of workload and help team adapt to changes in requirements. Burndown chart will help team set the estimated workload correctly.

- Handling Massive Workloads - The workload will be chosen by team members and evenly distributed during the Sprint, so the pressure towards the end of the Sprint will be removed. QA and DEV parts of the team will not have to fight for their place in the team.
- Costs Reduction– Prolonged Sprints, late delivery and unfinished products will be removed from the project ABC thanks to planned and gradual iterative development.

CONCLUSION

Effective communication of team members is a fundamental element in every project. Software development project is no different from that, and with increasing number of customers, every company should focus on continuous improvement of processes, communication flow and techniques they use to deliver the product. Proactive improvement is a must in the software development sphere and every project that discovers a weak spot in the development process should react and find an effective way to prevent it from appearing again. Whether a project discovers a weakness in the communication or a delivery process, team members and management should create a solution that takes into consideration all the specific characteristics of the environment they operate in. Forcing a project to adopt a framework without bearing this in mind might result in more costs than improvements in the future.

This is the case in the project ABC, as they attempted to bend the project unique characteristics to fit into a methodology for the sake of using it. As a result, project not only did not improve the workflow, it continued to deliver products past deadlines and without required features. This knowledge has been obtained by analyzing the concurrent situation of the project, with the aim of locating the main source of all weaknesses found in the project.

The proposed solution of this thesis was created to remove the inefficient way of using the Scrum methodology and instead, adopt and modify the core elements of the Scrum to fit the uniqueness of this project, as it is not a typical software development project. A lot of changes that have been proposed in this thesis are a combination of several models of the software development management with the goal of eliminating all issues connected to the planning, processes execution and the communication workflow.

REFERENCES

[1] RUBIN, Kenneth, S. Essential Scrum: *A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional, 2012. ISBN 9780137043293.

[2] AMBLER Scott, W. and Matthew HOLITZA. *Compliments of IBM: Agile For Dummies.* John Wiley & Sons, Inc., 2012. ISBN 9781118305065.

[3] THE AGILE ALLIANCE. *Manifesto for Agile Software Development [online]*. ©2001. [Accessed 2018-05-01]. Available from: http://agilemanifesto.org

[4] APELLO, Jurgen. *Management 3.0: leading Agile developers, developing Agile leaders.* 1st edition. Upper Saddle River, NJ: Addison-Wesley, 2011. ISBN 0-321-71247-1.

[5] SHORE, James and Shane WARDEN. *The Art of Agile Development*. 1st edition. Sebastopol, CA: O'Reilly Media, 2007. ISBN 978-0-596-52767-9.

[6] RASMUSSON, Jonathan. *The agile samurai: how agile masters deliver great software.* Raleigh, North Carolina: The Pragmatic Bookshelf, 2010. Pragmatic programmers. ISBN 978-1-934356-58-6.

[7] *Kurzy.cz* [online]. ©2018 [Accessed 2018-05-13]. Available from: https://rejstrik-firem.kurzy.cz/25309978/sde-software-solutions-sro/

[8] *Sde.cz* [online]. ©2018 [Accessed 2018-05-13]. Available from: sde.cz

[9] *Sdeusa.com* [online]. ©2018 [Accessed 2018-05-13]. Available from: sdeusa.com

[10] LEOPOLD, Klaus and Siegfried KALTENECKER. *Kanban change leadership: creating a culture of continuous improvement.* Hoboken, New Jersey: John Wiley & Sons, 2015. ISBN 9781119019701.

[11] RUCKER, Rudy. *Software engineering and computer games.* Harlow, Angleterre: Addison-Wesley, 2003. ISBN 9780201767919.

[12] MURCH, Richard. *Project management: best practices for IT professionals.* Upper Saddle River: Prentice Hall PTR, ©2001. ISBN 0-13-021914-2.

[13] STOICA, Marian, Marinela MIRCEA and Bogdan GHILIC-MICU. *Software Development: Agile vs. Traditional.* Informatica Economica [online]. ©2013, October 1, 2013, 17(04), 64-76 [Accessed 2018-05-13]. DOI: 10.12948. Available from: http://www.revistaie.ase.ro/content/68/06%20-%20Stoica,%20Mircea,%20Ghilic.pdf

[14] NERUR, Sridhar, RadhaKanta MAHAPATRA and George MANGALARAJ. *Challenges of migrating to agile methodologies. Communications of the ACM* [online]. ©2005, 48(5), 72-78 [Accessed 2018-05-13]. ISSN 00010782. Available from: http://portal.acm.org/citation.cfm?doid=1060710.1060712

[15] HIGHSMITH, Jim and Alistair COCKBURN. *Agile software development: the business of innovation.* Computer [online]. ©2001. 34(9), 120-127 [Accessed 2018-05-13]. ISSN 00189162. Available from: http://ieeexplore.ieee.org/document/947100/

[16] *Scrumpoker* [online]. ©2018 [Accessed 2018-05-13]. Available from: https://Scrumpoker.online/

[17] *IdeaBoardz* [online]. ©2018 [Accessed 2018-05-13]. Available from: http://www.ideaboardz.com/

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DDS (Design Document Specification)

DEV (Development)

DSDM (Dynamic SW Development Method)

FDD (Feature-Driven Development)

GDPR (General Data Protection Regulation)

IS (Information System)

JIT (Just-in-Time)

QA (Quality Assurance)

SDLC (Software Development Life-Cycle)

SRS (Software Requirement Specification)

SW (Software)

SWOT (Strengths Weaknesses Opportunities Threats)

TDD (Test-Driven Development)

UX (User Experience)

WIP (Work in Progress)

XP (Extreme Programming)