Veronika Vallen

Customization and application of a software project management process in a small and medium-sized enterprise

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

3 April 2019

Author Title Number of Pages Date	Veronika Vallen Customization and application of a soft-ware project management process in a small and medium-sized enterprise. 45 pages + 2 appendices 3 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Media Engineering
Instructors	Ari Peltoniemi, IT director Kari Salo, Principal Lecturer

Nowadays customer demand leads to the digitalization of services. Functions that have been manual are being automatized, paper products are transitioned to the digital forms. To create and maintain digital products to the level of customer expectation, many service providers prefer to organize their own development teams or combine the internal teams with external subcontractors.

New products, functionalities and services are delivered through project work and therefore require project management framework to support its implementation within organization. In the race for winning the competition for the customer businesses must learn to deliver its project as fast as possible. Agile software development frameworks such as Scrum became a clear winner when it comes to customer satisfaction and ability to respond to change. Many companies that recognized the benefits of applying such frameworks to their projects.

Scrum works well for the companies that can support designated resources for the project, and the team preferably should share the location since it requires great flexibility especially when it comes to planning of the workload. When it comes to the SME (including the company in scope of this thesis work), there are always more projects (needs) than people to work on them. This thesis is dedicated to the work of defining the standard process for developing new projects and functionalities through creating customized project lifecycle and applying agile thinking and framework to the phases of the lifecycle where it is appropriate. The goal is to improve the current process to speed up deliveries, share limited resources among projects in an efficient way, remove wasteful processes, and as a result of all the improvement activities, improve the quality of the final product.

Keywords	project, process, Agile, software, project life-cycle, methodology

Contents

List of Abbreviations

E2E	End to end. including everything that is necessary for all the parts of a computer network to be connected and work together: including all the stages of a process:
KPI	Key performance indicator. It is a value that is being followed in order to measure how effectively an organization is achieving key business objectives.
PO	Product owner. Product Owner is a role in Scrum development framework, whose main activity is to manage product backlog.
RAD	Rapid Application Development. Agile action-oriented software development framework that uses heavily prototyping with the user and focuses on delivering a working software.
SAFe	Scaled agile framework. Development framework built upon agile principles that allows to scale the development methodology across enterprise.
UI	User interface. The medium through which the user interacts with computer system.
UX	User experience. The overall experience of an end-user interacting with a computer system.
XP	Extreme Programming. Agile software development framework with the main objective to improve software quality.

1 Introduction

This thesis is dedicated to the work of defining the standard process for developing new projects and functionalities through creating customized project lifecycle and applying agile thinking and framework to the phases of the lifecycle where it is appropriate. The goal was to improve the current process to speed up deliveries, share limited resources among projects in efficient way, remove wasteful processes, and as result of all the improvement activities, enhance the quality of the final product.

It was identified that to succeed and be able to satisfy consistent demand on new products, functionalities and services team needs to agree on a standard approach to project work. First objective was to outline the process as a lifecycle through which the interesting idea of new functionality evolves into actual function that add value to the customer experience with the business solution. Second objective was to define a process and output for each step of a lifecycle to ensure that certain subjects are considered in the right moment and order as well as to provide each role in the project with required input to enable team members to carry out personal tasks.

The company in scope is a SME company working in financial industry with about 60-70 employees. Some of the services company provides are delivered to the user through sophisticated digital solutions built by internal development team (10 team members) in collaboration with external IT development partners. Also, development team oversees improvement of many internal back-office systems that support everyday work and processes of the company.

1.1 Current state analysis

Since this company is not a software development firm that can solely focuse on producing digital solutions, we must consider certain specifics of working on the development project.

- Size of development team is considerably small. Team members are working on different projects simultaneously, meaning that the same people cannot dedicate fulltime to one project.
- Part of the development work is outsourced to external consultants, most of whom are working from abroad.
- Building a project requires participation of the people whose main responsibilities are outside of the project work, for example, marketing, sales, customer service.
- Whereas part of development work can be outsourced, the task of understanding customer needs relies on the internal customer connection and knowledge.
- In majority of the cases new functions are built upon or depend on the legacy systems, which significantly complicates the development task, especially considering that the knowledge of these systems is distributed among different specialists.
- Budget dedicated to development work is limited, so it needs to be carefully considered how the money is invested.

1.2 Project management challenges

Considering the role of the company in the development process and the existing set-up there are several challenging questions that can be identified:

- How to design, specify and develop valuable solutions for the customer?
- How to develop solutions fast, efficient, within the budget and available resources?
- What shall be an approach to design that allows to consider properly complex system set up?
- How to ensure that the team can support different projects simultaneously?
- How to efficiently use the time and knowledge of the employees outside of the development team?

2 Theoretical background

2.1 Agile methodology: transition from traditional project management

Even though the Agile as a concept was summarized in 2001 when the Manifesto for Agile Software Development has been formulated (Beck et al., 2019), the Agile frameworks and ideas about "agility" in product creation process have been evolving for a very long time.

It all started in 1948 when Taiichi Ohno, Shigeo Shingo and Eiji Toyoda have developed revolutionary ideas of "Toyota way" where they defined fundamental principles of lean production and lean managements such as continues improvement and respect to people. Since then many key frameworks have been created and taken into use, each new idea has been bringing an important input based on its success or failure, some examples of which would be "Rapid Application Development" (RAD) and Extreme Programming (XP) that brought the concepts of frequent releases and short development cycles as well as a large amount of technical programming aspects of engineering a software. (Cobb, 2013; Measey, Levy and Short, 2015) Therefore, the 4 agile principles and 12 values (Appendix 1-2) are based on best practices of the ideas that have been tested for over 50 years.

In short, the goal of the true agile approach is to create a working software that the customer will find valuable because it reflects the customer's real needs. The real needs can be collected only through interaction with the individuals that are going to use the software, and it is completely normal that understanding of the needs can develop and change while working on the project; therefore, it requires constant collaboration with the end user. (Cobb, 2013)

And, of course, Agile methodology puts great emphasis on the value of the people involved in the process of creating the product. Agile is the methodology that brought the people working on the project or consuming the results of the project above anything else. Project work becomes more human oriented, more adjusted to the human psychology, and that is one of the main key success factors of such approach. (Cobb, 2013)

A more general connotation of the word “agility” is defined by Dr. David Rico as follows:

> The ability to create and respond to change to profit in a turbulent global business environment.
>
> The ability to quickly reprioritize use of resources when requirements, technology, and knowledge shift.
>
> A very fast response to sudden market changes and emerging threats, by intensive customer interaction.
>
> Use of evolutionary, incremental, and iterative delivery to converge on an optimal customer solution.
>
> Maximizing the business value with right-sized, just enough, and just-in-time processes and documentation. (Dr. Rico, 2019)

Agile manifesto is a set of principles and values that are theoretical whereas the agile frameworks are aiming to put these concepts into practice. In other words, they define certain practices that are incorporate values into steps that can be applied to the work of the team.

For example, we take the value from the top of the list:

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software (Beck et al., 2019).

Early and continuous delivery is managed through the concept of increments and iterations that is key concept of many agile methodologies such as Scrum or SAFe (Cobb, 2013). At the same time some questions remain unanswered. For example, how to define value or what is the working software or how to keep the pace of the development constant?

There could be many reasons why moving from waterfall project management to Agile can be challenging. One, of course, is the difference in the mindset that shall be adopted throughout the business. But also there has been a common misconception, especially at the early days, that being Agile means completely give up control over the costs and schedules of the projects. (Cobb, 2013) It is understandable why many companies will not agree to have a project with no deadline and no idea of how much it will cost in the end. In most cases there is a limit of how much resources in terms of time and money a business can dedicate to a certain project.

Waterfall methodology, which has been widely used before the Agile time, focuses on the upfront planning and control over cost and schedule of a project. As a result, it is heavily loaded with cumbersome documentation. It is a very straight forward approach that consists of sequential phases, but it has proved ineffective, extremely bureaucratic and of the false thinking that the schedules and costs defined at the very beginning of the project are dependable. (Cobb, 2013)

The cost and schedule estimates cannot be any more exact and precise than the requirements are, and in most of the cases the requirements at the beginning of the project are too uncertain. What is even more inconvenient is the fact that they might be a subject to change due to some new realizations or a rapidly changing business environment. Users are not capable of defining their needs explicitly without relying on visual examples or experience of working with similar systems. Even then, it can be difficult, and some requirements can surface only later in the project when the team stumbles upon new information. This means that there two likely scenarios: the project may either meet its cost and schedule goals but miss the target in satisfying the business needs or earlier estimates of costs and schedules will become worthless. (Cobb, 2013)

Another risky assumption of the waterfall model is that when developers read the documented requirements they will understand it perfectly, and nothing will be lost in translation.

Agile thinking brought the understanding that the business environment is most certainly is unstable, and the requirements of the project will be subject to change throughout the project. And since the cost and schedule estimates may not be as exact and precise, it is much better to just accept the fact that the user requirements are uncertain and choose a model that is designed to be more flexible and adaptive to uncertainty. And if there is a way to accurately manage costs and schedules, it is by having a good approach to control and manage changes in requirements and scope. (Cobb, 2013)

Since the Agile frameworks has become the way of achieving agile values, it has created the impression that the only way to be agile is to practice a defined framework such as Scrum, Kanban or XP. But even though there are several methodologies that are widely used, there is a whole spectrum to being agile. (Cobb, 2013)

If we take waterfall methodology and Scrum methodology as two extremes, the former being all about upfront planning and predictability and the latter being about adaptivity to the current requirements and user needs, there is a whole spectrum of ways to mix and match different levels of agility and control to fit a given business environment. (Cobb, 2013)

Being agile is not about practicing a specific methodology; in fact, in many cases it is impossible to apply the methodology by the book, and a smarter approach is to blend together different methodologies, tools and ideas to develop an effective overall strategy that is aligned with the company’s business culture, risk levels and even complexity of individual projects. In a way if Agile principles are well understood, it does not matter which framework is adopted. (Cobb, 2013)

2.2 Scrum

Scrum has been created by Ken Schwaber and Jeff Sutherland in the early 1990s with the goal to help businesses address complex problems. It is used heavily in the software development, but not limited to it. It has been used to develop hardware and even applied within governmental projects. The founders of Scrum emphasize that it is lightweight and simple to understand, but it is challenging to master. (Schwaber and Sutherland, 2017) It is an Agile framework that defines practices through which to apply Agile principles and values to the product development.

Scrum is a great model to apply to software development projects. It is founded on the empirical process control system, meaning that decisions are based on what is known and a new knowledge is gained through experience. It accepts the fact that in some cases it is not possible to predict or acknowledge all the details of the implementation and applies an iterative and incremental approach with which project team acts upon what is known and adjusts once the new knowledge is obtained. (Schwaber and Sutherland, 2017)

Iterations is the key concept when it comes to Scrum. The goal is to deliver the product iteratively and incrementally so that the team can collect feedback and adjust if necessary. The work is done in Sprints. Sprint length can be up to one month. During this time a usable and potentially releasable increment shall be created, but, of course, that is not always the case. Sometimes it takes more than one increment to make a release. (Schwaber and Sutherland, 2017)

Scrum relies heavily on a well-trained and specialized team capable of self-management, communication and decision making (Cobb, 2013). The Scrum team consists of development team, scrum master and product owner.

Product Owner is a person, whose role is to manage a backlog. Backlog shall consist of the items that are clearly defined, prioritised according to the business value and optimised performance. The person is also responsible for explaining the items to the development team and to keep the work of the team transparent for the rest of the company. It is also important that the Product Owner is sole person and not a committee, and all the questions regarding adding new items or re-prioritizing existing ones shall be addressed with the PO. (Schwaber and Sutherland, 2017)

The development team is a group of professionals who work together on increment of the product. There are no roles or titles for team members, but for the team's work to be successful its members shall have all skills necessary to work on the product. Individuals can have areas of focus, but the whole team is accountable for the result. The size of the team is not defined, but it shall be large enough for the size of the work done during the sprint to be significant. On the other hand, team cannot be too big to make the communication complicated. (Schwaber and Sutherland, 2017)

Scrum Master is a role that supports Product Owner, development team and organization. Scrum Master promotes the methodology and trains people on the theory, practices and values. In a nutshell Scrum Master enables the team to work according to methodology by coaching the team to be self-organized and cross-functional, helps team to understand the goal and value of each item and sprint in general. The role is also about protecting the team's work and removing all the obstacles to team's progress. (Schwaber and Sutherland, 2017)

There are two types of backlogs in Scrum. One is the Product backlog which in practice is a list of all product requirements sorted in prioritised manner. It stores all the features, functions, improvement ideas and fixes that are being constantly added to the list and prioritised among other items. It is a living artefact that shall be always in line with the current business needs, market conditions and technology. Backlog items are being refined to the point they are clear enough how they shall be implemented and can be estimated. Sprint backlog is the set of the items that have been forecasted to be developed during the Sprint. The point of the Sprint backlog is to make it very visible what has been set to be accomplished during the Sprint and have a clear picture of the progress at the specific point in time. (Schwaber and Sutherland, 2017)

To ensure that both backlogs are being constantly maintained and items are clear for the team, Scrum holds regular events that are on one hand designed to minimise the need for meetings and on the other hand ensure the transparency and systematic communication. The work is done in sequential Sprints, where Sprint is an agreed timebox during which the team is focusing to develop a product increment that potentially can be delivered to end user. It is advisable for the Sprint not to be longer than one month to enable predictability and limit the risk of long-term commitments in the ever-changing business environment. (Schwaber and Sutherland, 2017)

To run a successful Sprint, apart from development work itself there shall be other activities organized within the Sprint such as Sprint Planning, Sprint Review, Sprint Retrospective sessions and Daily Scrums. (Schwaber and Sutherland, 2017)

Sprint planning is a collaborative task of the Scrum team to gather the understanding of the work to be done and to evaluate what can be accomplished within next Sprint. The amount of work that can be done within one Sprint is unique for every team but keeping track of the past performance can help to project the capacity to the future Sprints. The workload is taken from the Product backlog, which is essentially a prioritised list of development items. This session is the time to clarify each selected item and define how it is going to be done. (Schwaber and Sutherland, 2017)

Daily Scrums are 15-minute events held every day to briefly discuss the planed work for the next 24 hours, to keep track on the progress and ensure consistent collaboration within the team. In case there has been some new discoveries highlighted during the Daily, more detailed discussions might follow. (Schwaber and Sutherland, 2017)

At the end of each Sprint there must be a Sprint Review to go over accomplishments of the Sprint together with the project stakeholders as well as to go over the high-level plan, as an input to the subsequent Sprint Planning. It is also a place to highlight the changes on the market, new opportunities and review the timeline and budget. (Schwaber and Sutherland, 2017)

The last but an important event is Sprint Retrospective. The main goal of Sprint Retrospective is to integrate the continues improvement into the process. The team reviews the previous sprint on the subject of what went well and what can be done better. The session allows to adopt the best practices and, as a result, to improve the quality of the product. (Schwaber and Sutherland, 2017)

2.3 Applied Methodologies

It is clear that the company in scope is looking for the Agile way to carry out project work since it is the most efficient way to ensure customer satisfaction and ability to respond to change in demand as soon as possible.

The company has already recognised the benefits of practicing the product development work through Scrum methodology, but it was also recognised that some of the duties of the project work require better process definition than Scrum can offer. Also, with the current team set up certain requirements of Scrum cannot be fully supported as defined by the book.

Scrum works well for the companies that can support designated resources for the project, and team preferably should share the location since it requires great flexibility especially when it comes to planning of the workload. In this case part of the team is working remotely. Therefore, the standard Scrum approach requires to be adjusted to the business environment.

Scrum has a continuous improvement approach through the retrospective, but not strong enough process orientation to transfer those lessons learned into a broad-based process definition that is carried forward to the other projects (Cobb, 2013).

The process described in this thesis has been mostly defined retrospectively through experimenting with different approaches, defining major pain points of the previous development experience and putting actions in the logical order.

3 Project management lifecycle and process

The goal of the project is to define a standard process for software development that will allow to make fast deliveries, effectively share limited resources among projects, remove wasteful processes and, as a result, deliver a better-quality final product.

To run a project, two methodologies are needed: project lifecycle and project management process. Project lifecycle consist of sequential steps of various phases of the project and might vary drastically depending on the industry. (Sasal, 2019)

The first objective is to outline the lifecycle through which the customer needs are developed into business solutions. It has been recognised that the product development work goes through a repetitive consequence of steps (lifecycle), and a project management process of each step would require a better definition than what Scrum development framework can offer.

The second objective is to define a process and output for each step of a lifecycle to ensure that each step produces the necessary results, certain topics are considered at the right moment in the process and team members are provided with required input to carry out personal tasks.

Process of each step shall be:

- agile
- standard and reproducible across different projects
- customer oriented
- design oriented
- focusing on planning “just in time”, instead of redundant upfront planning
- documenting just enough
- continuous and scalable

As has been already pointed out, when it comes to the project management process, the team has agreed to use Agile Scrum framework, benefits of the Agile and Scrum has been described in the dedicated chapters. The Scrum framework originally has been applied only to the implementation phase of the project leaving important phases of the project such as business analysis and design undefined.

3.1 Development process lifecycle: from Idea to production

Figure 1 shows phases that are relevant to all project types of the company in scope. It considers all the steps that are important to emphasise and keeps the idea simple even though in practice there is a lot to consider during each phase.

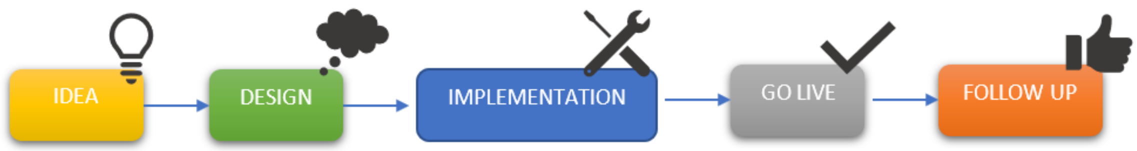


Figure 1. Phase of project lifecycle.

Development lifecycle starts with the selection of the idea worthy to dedicate resources for. Once the idea is selected and understood, it goes through a design phase where it is processed to the condition where development can start. This phase can take reasonably long time because this is the time to experiment, collect feedback and validate assumptions. It is much faster to manipulate, easy to adjust and considerably cheaper than actual implementation. It is also important to note that the design phase assumes a technical design as well.

Once designs are ready, implementation can be started. If all previous steps are done well, the development becomes a very clear process, and that is why it is beneficial to put enough effort into two previous phases. In this case implementation covers more than writing code, but also code-reviewing and quality assurance as well.

Go live phase is about preparing the delivery of the application to the production servers as well as proper communication around the delivery. Follow up phase required to ensure that the delivered product is a success from technical and business point of view.

It has helped to define several issue types to support the process phases, where issue type has a separate workflow and scope. The idea here is that with the progress through the lifecycle the overall task is being broken down into smaller more concise actions (see Figure 2). Not always, but in many cases, business requirement will consist of more than one process that would be convenient to design separately. Also, design story might be too broad to be implemented as a single development task.

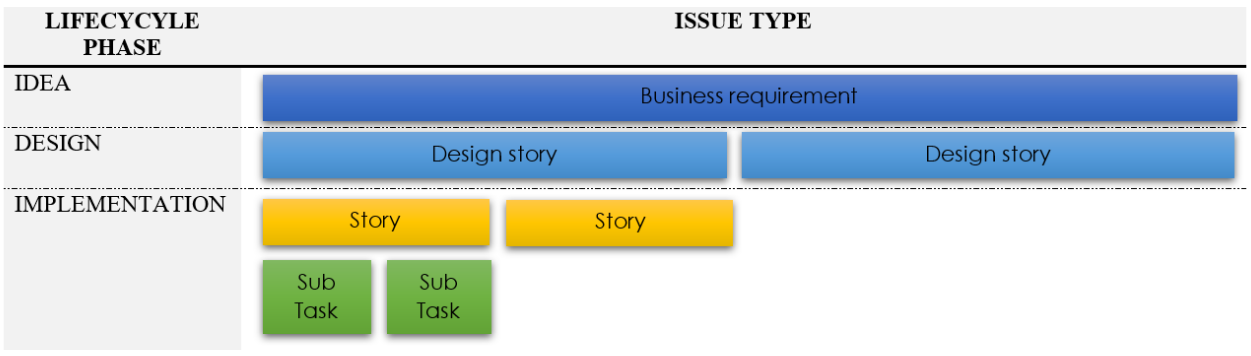


Figure 2. Illustration of the scope of the issue type and its relevance to the lifecycle phase.

The process captured in the Figure 1 looks much like a “waterfall” model, but in fact it is not. Many agile principles have been applied to each project workflow step. Figure 3 shows how the process looks like when agile principles and tools are applied to support the process control.

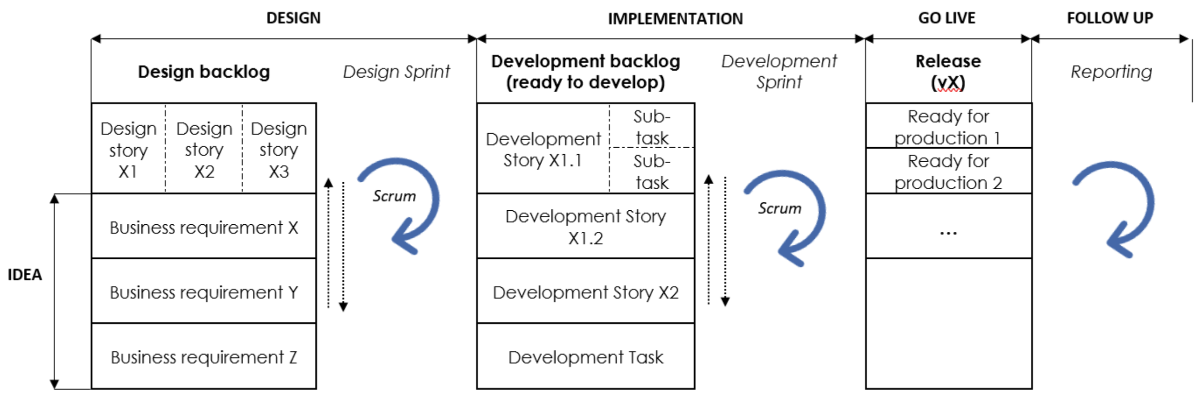


Figure 3. Process control of the project management lifecycle.

Each phase will be described in more details in the next chapters, but in a nutshell, it is designed to have two separate backlogs and two activity sprints. Design backlog is dedicated to collecting and handling business requirements, designs and specifications. Development backlog is fully dedicated to support the implementation phase. Both boards are managed as Scrum methodology dictates, practicing all the planning and communication related activities, as well as applying the principle of iterations and increments. The task that goes under the processing is always sliced into reasonable sizes and prioritised. The good example to illustrate is if we would have a target to build an application, each function of the application would go through the process and not the whole application at once.

The idea is that the design sprint shall produce the necessary input to the implementation phase; thus, the scope of the design sprint is always ahead of development sprint. The two sprints can be run simultaneously, but having dedicated sprints to these phases allows more flexibility to level out the workload. For example, design sprint can take a longer period of time, but at the end produce enough work to be gone through several development sprints.

The approach has a good scalability. If the project has a small team, it is possible to go through lifecycle one function at a time. But if the team is big enough and has all necessary skills available, team can learn how to process several such functions concurrently.

3.2 Planning

Unlike the waterfall model when most of the planning is done upfront, Agile methodology suggests recognising the limits of planning in a turbulent environment (Highsmith, 2019). This means that the planning is distributed and repetitive task throughout the process.

There are several layers to planning an agile project (1):

Vision – is the plan of the goal and objectives of the project. It is a larger picture that supports overall business strategy for the period, for example, for the next calendar year. This is the most stable part of the plan.

Roadmap – is a breakdown of the vision into the actions (features, functions, fixes) or selection of development ideas that are in line with the vision.

Release – is an action of grouping the business requirements/ designs/ development stories in a way they will be delivered to the end user. The scope of release planning is done throughout the progress of the project. The initial planning shall be done as soon as possible, but it is okay to adjust it in the progress.

Iteration – defines the tasks to be done during one iteration.

Daily – is an everyday act of reviewing progress against planned effort, which helps the team communicate and resolve obstacles.

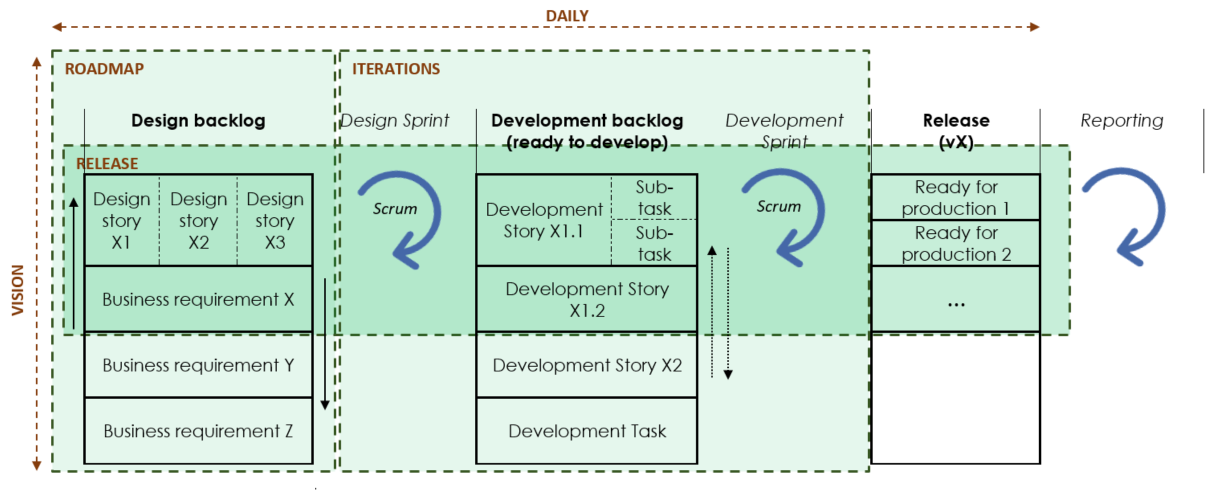


Figure 4. Layers of planning mapped to the project process control.

Figure 4 visualises the way each planning layer can be mapped to the project process. As shown in Figure 4, the design backlog becomes a convenient tool to work on the roadmap prioritization and to ensure that the roadmap is in line with the business vision for the project.

Design and development sprints are organized through the iteration planning, and development backlog support the planning by slicing the functions once again into development-oriented tasks.

Prioritization and estimation are important tasks of planning effort, but the purpose and extent of details to which it goes is slightly different depending on the phase of the project lifecycle. It will be covered more specifically in the dedicated chapters for idea, design and implementation phases.

Release planning can start very early and adjust throughout the progress. It is possible to combine one or many business requirements into the next release version or even take only a part of the requirement (only certain design stories). When application is developed in a team, it is very important to define the scope of the next release in advance. To deliver new version of the application to production, the development must be finalized and the whole version shall go through the set of regression tests to validate that the new development does not create any conflicts with previous version. Having a clear scope of the release in advance draws a very clear line when the development for this release shall stop and the team shall focus more on the quality of the release instead of putting more content to it. Otherwise, the release will never be ready to be delivered due to constant work in progress.

3.3 Tools

Having appropriate tools to manage the tasks can be absolutely necessary to keep up with the fast-paced work (Cobb, 2013).

Now when the project lifecycle has been defined, it is necessary to apply it in practice, and it can be achieved with the help of the well-defined processes, templates and project management tools.

There are many tools that can help to run application development project in an efficient way, but there are three must have tools that are required if the work is done in a team. The tools are:

- version control and collaboration platform
- issue tracking software
- collaboration wiki tool

3.4 Version control and collaboration platform

In this case GitHub is a version control and collaboration platform that developers are using to combine their effort into one functional application. Since the development practices are out of scope, so is the functionality of this tool. But it is important the process flow of using such tool is taken into consideration in the implementation phase.

3.5 Issue tracking software

To track the project properly, the project management tool (or issue tracking tool) is required. For this purpose, JIRA is being utilized. JIRA is an issue tracking tool that allows using Scrum as a framework and Kanban board as a process control tool. Jira helps us to follow almost on every step of the project starting with idea gathering to the point when the new version of the application has been delivered to production. It helps to collect the ideas, keep the priorities straight and oversee the progress of the issues (stories). There is a lot of freedom on how to use the tool, and the goal is to apply the tool in a way that it supports defined project lifecycle and matches the real-world processes.

JIRA is the tool where the project with two boards described above is created, and issues are stored in the backlog and then are put to active sprint once the issues become a focus of the next development. Also, there is a functionality which allows to create issue types that have been defined and assign custom workflows to them.

3.6 Collaboration wiki tool

For creating and sharing the documentation Confluence collaboration wiki tool is used. There are several reasons why it is good to keep documentation in one place, instead of spreading it among the issues in JIRA:

- There is one source of truth. Once the development starts, specifications might slightly change due to some discoveries, and if documentation is scattered among different stories, there is no easy way to keep it up to date and even spot all the dependencies.
- Once the development is done, stories are closed, leaving no easily accessible documentation behind for the further reference, unless somebody collects information retrospectively, which is an additional workload.
- Usually when it comes to the final design, it is not only the development part that is important, but, for example, communication internal or external. Having the document to rely on to produce other type of documentation such as training materials, instructions and marketing material make the job that much easier.
- It shows the whole picture and makes it easier to see how much effort it is going to take, especially if the task involves different teams or different applications.

4 Idea phase

Previous chapter was a general description of the project lifecycle and starting from the current chapter each phase will be described in more details. Figure 5 highlights the idea phase of the project lifecycle. Idea phase is managed in the design backlog and consist of list of high-level tasks which are agreed to be identified as business requirements.

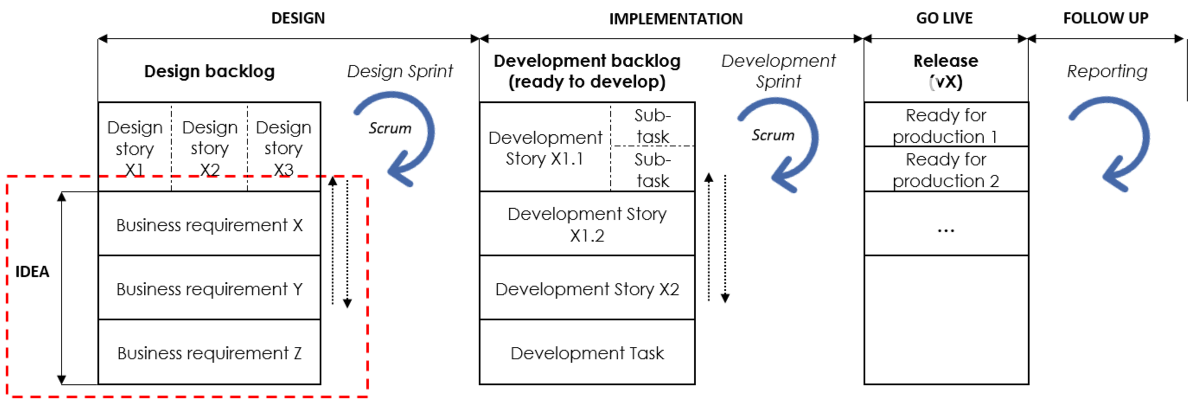


Figure 5. Idea phase of the project lifecycle

4.1 Business requirement

It all starts with the idea. Development ideas come from different business or customer needs. These are the development idea categories that can be identified:

- new application
- new function/ process of existing application(s)
- improvement of existing function

Considering that in Agile methodology the team is not trying to build the whole application at once, but instead views application as a set of functionalities, it is possible to regard each functionality of the application as a business requirement. For the project work, business requirement issue type is used to describe:

- Items of the business road-map to fulfil company vision for the next period.

- Idea of development that has not been evaluated, specified and validated to be developed (for e.g. customer service requests).
- Items that arise during current design/development but are outside of the current scope (must be prioritized against other items in the backlog).

Business requirement is a very high-level definition of the end user, business or system need. At this point it is just an idea and it is uncertain if the feature will be valuable to the end user, but the good news is that this is not the time to make a commitment.

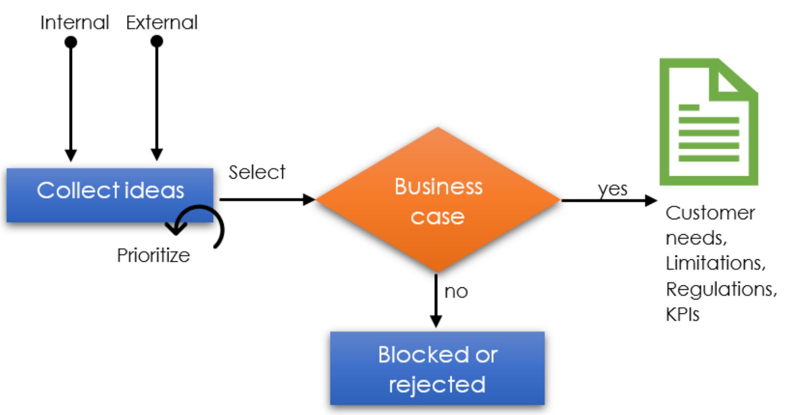


Figure 6. Processing of the collected ideas.

As illustrated with Figure 6 such requirements need to be collected, prioritized and analyzed on the subject of value. If the idea is not worse the effort, then it can be rejected, but if it proves to be beneficial, then the business requirements shall be defined as well as the measure of success. It is also important to get a general understanding how time-consuming development can be, but the estimation at this point can be only high-level because the requirements are not explicitly defined. For example, T-shirt sizes can be used at this point. The sizes that can be applied to requirements are extra-small, small, medium, large, and extra-large, similar to the t-shirt sizes in the physical world. For simplicity abbreviations are being used: XS, S, M, L, XL. The estimation is fairly rough, and the main goal is to understand the relative size of the requirement to understand how development topics compare among each other. This method of estimation is a very fast and easy to apply; therefore, it works perfectly for an initial estimation.

4.2 Source of the development ideas

Ideas for development may come from internal and external sources. Internal ideas come from within the company, for example, brought by sales representatives or customer service. People that are in direct customer communication can often defined customer pain points as well as possible solutions to those. External ideas are the ones that collected from the customer based on their direct feedback through the interviews and questionnaires.

Without a doubt the better approach is to collect the feedback directly form the end users of the application. Having a constant customer feedback is essential to provide the desired value to the user. But the input from the people who support daily operations of the end user is also valuable especially considering that they understand the existing internal processes and their impact on the customer.

The first step is to collect the business requirements. It is a very good sign if company has a customer or business team that are enthusiastic about improving the services and full of the ideas how to do so, but it also means that those might arrive to the table at the most inconvenient times when the team is busy working on the high priority matters. Keeping focus is important for work to progress.

When it comes to customer needs, sometimes it feels like every customer demand is urgent and solutions are needed right now, but for the most companies the reality is that there is a limit on how much money can be spent and how much skilled people can be allocated, so there is no way to tackle everything at the same time.

There will be an appropriate time and place to process new ideas, but to keep track of all the ideas flying around it is very handy to have a backlog where it is possible to collect all ideas no matter how random they are without spending too much time on reacting to every single request coming your way.

Scrum defines only one backlog pull for the project, but experience showed that the development backlog is not the place to collect the random development ideas. Backlog grows too big, messy and too heavy to maintain, especially considering that the business requirement must go a long way before the implementation can even start.

And that is why instead of having one board for the project two boards have been created, each with dedicated backlog and active sprint. The concept is very similar to the approach that is defined in such scalable frameworks as Safe 4.0 where a product and project backlog are used. Suggested approach is not as complex, but it as well separates the work into two interdependent categories and keeps both parts clear and well maintained.

Just like in normal Scrum a board dedicated for the development and quality assurance had been used, but in addition to that a second board was created where the work on business requirements and designs will happen up to the point when the functionality is ready to be implemented. All the ideas will be stored in the design backlog as business requirements.

To progress through the ideas in the organized way and to ensure that team provides the valuable functionalities to the user in the right order, prioritization of business requirements shall be done occasionally. Prioritization helps to validate set priorities against current business landscape.

If this task is done regularly enough, then, as a result, the team gains a clean backlog that is prioritized based on the understanding of the value of each development in a current market situation. Then the backlog becomes a tool that helps to align to a company vision and the roadmap to get there.

The vision is the set of business goals and objectives that the project shall accomplish. It is very high-level, and it gives the direction to the project. The vision is the most stable part of the planning even though it is still important to check occasionally if initial vision is still true to the current reality of things.

The purpose of the roadmap is to break the vision into actions (which is what business requirements are in the current context) and plan in which order the functionality will be incrementally delivered to the customer. This is the initial place to combine the functions into releases and estimate approximately when each release can be delivered. (Cobb, 2013) Defining the scope for the release allows to know when time is to pause the implementation and focus on delivery.

4.3 Business decision

Estimation and prioritization work start already during the idea phase of the project. It is important to understand that at this stage it is almost impossible to come up with very accurate estimations simply because there is not enough information discovered yet. Estimations will get more accurate closer to the implementation phase, but at this point the purpose of estimating is mainly to understand the value of the development and its place among other competitive ideas. The primary goal is to validate that the ideas are worth doing and agree in which order they shall be delivered to the customer and maybe even get a high-level understanding when the delivery to the customer can be done.

One of the main questions here is how to avoid developing functionalities that are unnecessary or unreasonably costly? The goal is not to jump to the solutions too soon, before any ideas shall be understood and analyzed from the business point of view. The need has been identified to define a proper process for going through the ideas and analyzing them from different perspectives, such as business and customer value and estimate of the cost, before putting too much effort into development. Of course, the goal here not to go “waterfall”; thus, the depth to which such analysis goes can vary. The more money, time and risk involved the more it should be considered whether this development shall happen or not. This is the time to define what is the value of such implementation and, if stakes are high, a high-level calculation of cost in terms of time and money is required. But if the risk is low a simple yes or no decision can be enough.

In Agile methodology **value** is the key concept. In Lean and Agile methodologies extra features that no one needs are considered as a waste, they complicate the product and cause unnecessary maintenance, as well as are taking time of the team from focusing on the right features. It is okay to reject the ideas because it is a waste to create something that does not create value relative to the cost incurred. (Cobb, 2013)

Developing always has a cost in terms of money and time, if these two resources are put in a wrong “basket”, it is good to consider not only the financial investments that have been wasted, but also other opportunities that are lost.

In accounting there is a term opportunity cost, and in short it is a benefit that was discarded by not doing certain activity. Even if it is not the cost that is recorded in the project budget, it is still the cost that should be considered when deciding. (Accounting Coach, 2019)

It was not a goal of this thesis to describe how to build a business case, but there are several lessons learned that would be beneficial to highlight:

- When customer need is discussed, what is meant that the customer or business has a problem and the need is the solution of this problem.
- It is not an easy task to properly define what the actual problem is. It important to solve the actual issue instead of focusing on fixing the symptoms.
- A problem can have more than one way to be resolved, and it is a good practice to investigate different options before settling for a certain solution.
- It is important to investigate to whom the solution is valuable. It is very tempting to start working on something per single customer request, but sometimes the need is only relevant for one customer in particular and going with its demand might even have a negative effect on the rest of the client base.

4.4 Idea phase output

As a result, some of the ideas will be rejected, but for the approved idea to be ready for the design there are two beneficial inputs to the next phase that can be defined under the business requirement.

One is the business perspective on customer needs and expectations as well as limitations and possible regulations to follow. Basically, collecting any background information that can assist on making key decision during design phase can be useful.

Another beneficial action that can be initiated on the business requirement level is to define what will be relevant key performance indicator(s) (KPIs). If business requirement is properly analyzed, then there shall not be a problem to answer this question: how to measure a success? If the KPIs are outlined at this point, there are two benefits:

- Having a definition of success will help to make certain design related decisions.
- It will ensure that the data necessary for the measurements and follow up is collected from the start.

The team has now a dedicated board with the backlog collecting the ideas. Ideas are called business requirements and each requirement analyzed deep enough to be able to accept or reject the idea. Through business analyses the team can understand the value of putting a specific idea into further development and prioritizes it among other ideas. The list of the prioritized requirements becomes the roadmap that can be broken down into possible releases. KPIs are defined to be able to measure if expectations were reasonable. At this stage it is a good way to produce just enough planning, and know that things can change, and it is okay.

5 Design Phase

In this context design work covers everything that comes after the conscious decision of developing certain business requirement has been made and before the actual implementation starts. Therefore, as an input to the design phase team usually gets the idea that is somehow formed from the words of the client interpreted by other people. But if the idea phase went well, team also is getting a clearer picture of the benefit of such functionality and what is the measure of success. Figure 7 focuses attention on the design phase of the project lifecycle.

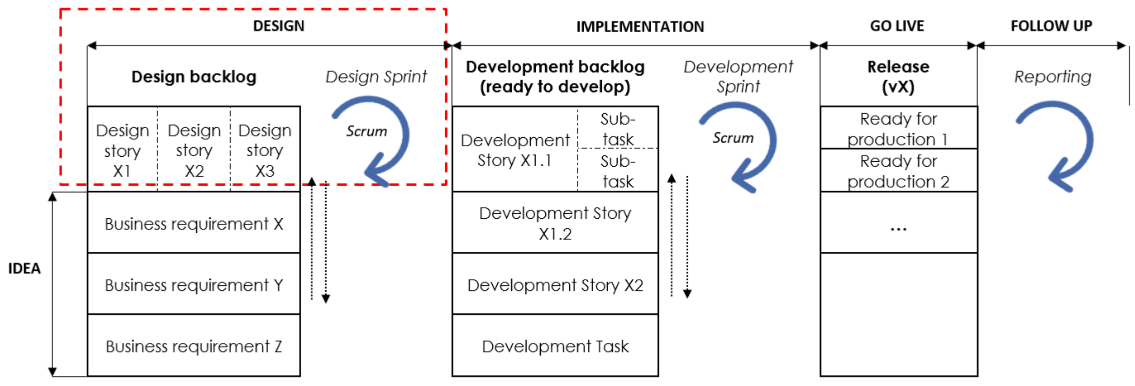


Figure 7. Design phase of the project lifecycle.

There are many techniques to facilitate the design process, and it really depends from case to case which tools to use in particular situation. The larger the tool box the better chance to end up with a perfect solution. But the goal is not to describe the tools, it is to focus more on the consequence of events.

When it comes to design, it was noticed that having a defined process is necessary. It is very hard to keep focus when the challenge is too complex. Process gives a direction to where to start, how to proceed and what shall be the result which will serve as in input to the next phase of the lifecycle. Process also assists to create a team work rhythm that would be easy to follow. It allows to solve the gaps in knowledge and understanding as they arrive instead of worrying about it all at once. In Figure 8 the schema that represents the idea of the design process in general is displayed. Two main components are the service design (including user interface (UI) design and user experience (UX) design) and technical design.

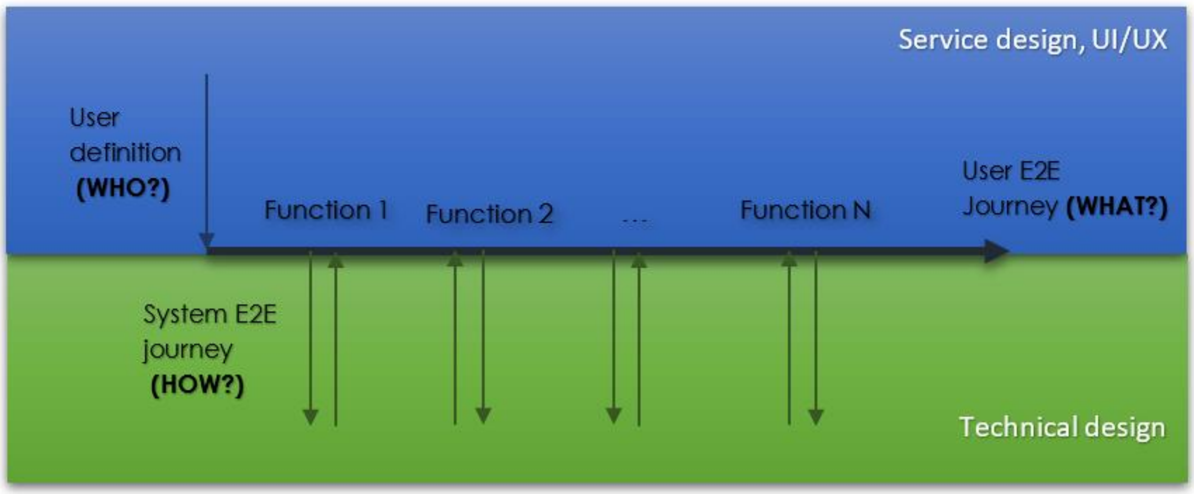


Figure 8. Visualization of connection between user and system end-to-end (E2E) journeys

As shown in Figure 8, user journey is the base on which all subsequent decisions are made. Therefore, first comes service design (or functional design if the work is done on a smaller scale development). The purpose of the service design is to build the unified and efficient user journey through the system in a most convenient and clear way (Talley, 2019). It should be in sync with the business case/model and technical peculiar features, but the user is still the king, and user experience shall not be compromised unless absolutely necessary. This is the process for which service designer and UI/UX designer usually are responsible, but as has been already covered in the Scrum team defining the clear roles is not an objective.

Service design also takes care of the special cases where the user behavior varies from the one that is outlined with the general process. To be able to do so, it is important to know who exactly user(s) of this process/ function are and how each type of the user is going to interact with the system to perform this task. It is good to try and define as many possible user types and scenarios as possible, especially the not so standard cases, because those are the ones that put the strongest pressure on the system. As long as it is defined who is the user and how the user might use the system technical specialists can define what the system should do to support each act, which will lead to understanding what the functional requirements of the system are.

Agile methods shall be applicable to the way solutions are designed and implemented. Idea of increments and iterations works well for this phase and allows solutions to come in reasonably small portions to be validated by the end user as worthy.

To be able to progress fast, the team must keep the scope reasonable and clear and go into details gradually. Quite often the business requirement is very broad and involves many actions. The important step of the process is to plan how to approach the problem. That requires continues breaking down of a complex task into more manageable steps, which in this case can be described in Design Stories.

The target is to keep scope small enough to be able to comprehend all the details, but preferably so that the task can be processed from end to end. When the scope is too big, it is easy to get confused or not to consider important facts. Therefore, sometimes it is okay to focus only on the part of the whole process (sub process) if dependencies are kept in mind, such as inputs and outputs to a different process. If dependencies between two processes are too high, then implementation will not be started unless both are defined well enough. It is also acceptable to go back and adjust the solution.

Qualities of design story:

- Design story usually is a part of (or equal to) the business requirement (see item gradation in Figure 2).
- Design story scope normally shall be a function / end-to-end (E2E) process from user and system point of view.
- *Important.* Design story that comes up during the current design/development but is outside of the current scope must be created and prioritized against other competitive items in the backlog.

Based on the points above, business requirement can consist of one or many user journeys, and user journey might consist of one or many system functions. First step would be to break the whole solution into high-level end-to-end user processes(journeys), then those processes can be broken down into rather independent user tasks (functions) that can be defined E2E from the system point of view.

Once the set of functions has been established, the list of tasks shall be estimated and prioritized once again. But in this case, we are not only appealing to the value of the task, but also are defining the best order considering possible dependencies between these tasks to ensure that the larger picture is kept in mind. It is better to start with either the simplest tasks or the most time-consuming task. Doing simple tasks will allow to progress and collect empirical knowledge about the problem fast, whereas focusing on the most time-consuming tasks will allow to avoid the bottlenecks in the future. Estimations applied to this phase also allow to compute the size of work and evaluate how much work can be done within next iteration.

The result of this planning is a roadmap how to get to the desired result. The roadmap as well as prioritization of it is not set in stone, it is important to adjust as feedback, knowledge and experience are gathered.

UI/UX work can start rather early. As soon as first understanding of how the process shall go from the user point of view has been gathered, it is beneficial to think about user interface and user experience. It is good to start early because viewing the task from this perspective might also add or remove the steps of user interaction with the system and dictate certain logic of the application.

The main goal, of course, the best customer experience possible, but at the same time it is essential to keep the system requirements and limitations in mind. The same works other way around, system requirements shall support defined user experience. That is why UI/UX design shall be in constant sync with the process definitions.

UI design can be iterative and incremental process as well. The designs can start from very simple and get more detailed. Modern prototype tools allow to build very comprehensive designs rather fast, but it is better to start with paper or paper-like prototypes. It will leave discussions of the details such as colours and fonts to a more appropriate time.

5.1 Feedback and interactions

Design process shall always revolve around feedback and design phase is the time to gather that feedback as much as possible (see Figure 9). Today prototypes are made fast, very similar to the actual solutions, mush cheaper than development, and very flexible to change; therefore, this is the best time to collaborate with the external and internal customers of the function in scope. It is possible to view the process of designing a feature as an iterative process around customer feedback, and with each iteration the actual result is improving, and the understanding of the problem starts to grow.

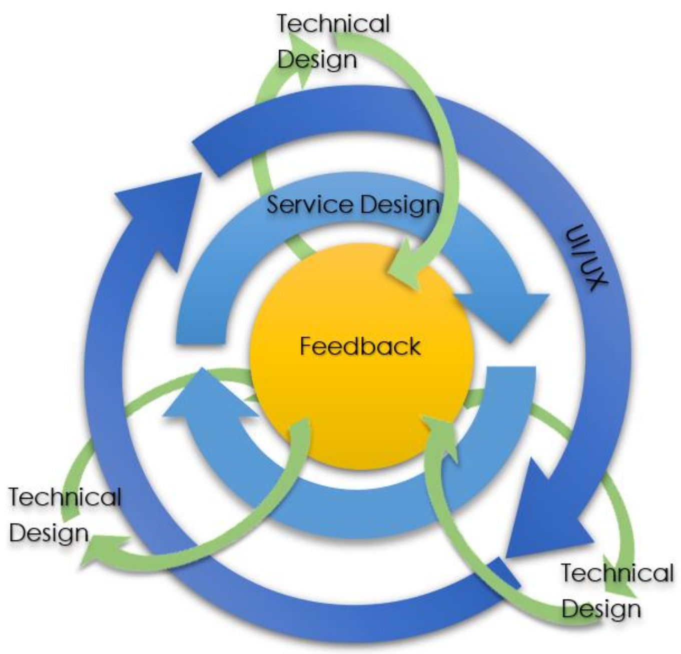


Figure 9. Feedback solar system.

When it comes to design review, feedback is equivalent to a quality check in the production line, and same principles apply. If the quality of the product checked once it is already finished, it might be so that the product is severely defected, and team basically must discard it. Of course, when the intangible product is designed, the effect might not be as dramatic, but it still might end in a lot of time and effort being wasted. Feedback shall be received frequently and guide the decision-making process. In Figure 9 the feedback is the force that in many cases will drag team backwards, and that means the less team consults with the user the further back the misconceptions might take it.

> Lurking beneath every goal are dangerous assumptions. The longer those assumptions remain unexamined, the greater the risk (Knapp, Zeratsky and Kowitz, 2016).

It is best to experiment fast, and make informed decisions, based on the gathered insights.

This is also time to get all stakeholders on the same page about the solution, hence there is no going back and forward once the solution reaches the development phase. It does not mean that once the development starts there is no room for change, but if design phase is taken seriously, the adjustments are minimal.

It is beneficial to partially involve developer(s) and tester(s) into the service/UI designs process. It brings new perspective on the designs, and it will be clearer to the developer how to develop and to tester how to test the functionality if they follow early enough on what is the idea behind the solution and how certain decisions have been made; otherwise, due to standard communication issues, many details can be lost in translation to technical language.

Technical design does not really require work with external customer, but it still important internally to involve right people at the right time, especially when new functionality is built on top of existing systems or as a replacement to the outdated solution. The problem is that usually systems and processes of the company grow too fast too big, and knowledge about them is distributed among different specialists, who know, use and maintain the system. Using the expertise about current processes allows to understand the limitations, dependencies as well as find areas to improve the flaws of the initial implementation.

5.2 System Journey

Even though the service design shall have a head start, it does not mean that the whole service shall be designed before technical design initiated. Good planning and scoping will allow to start and progress with the technical design as well. As soon as a function as a part of a holistic user journey is defined, the journey of the system to support it can be also specified. Figure 10 illustrates the part of the application that can be specified within one design story that describes user function and end-to-end system process to support such function.

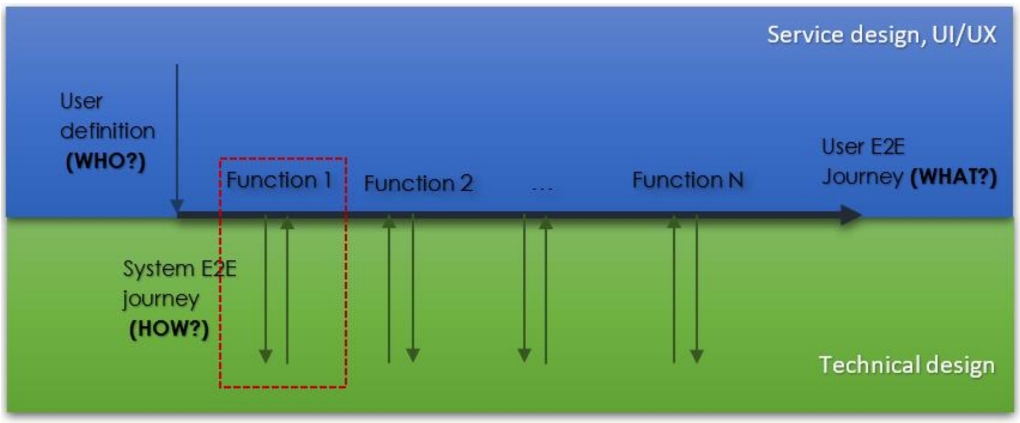


Figure 10. The scope of the design story (user function and system E2E journey).

Functional and non-functional requirements are part of the system requirement type. Non-functional requirements specify how the system should behave. They define system attributes such as usability, reliability, performance, security, maintainability, scalability, and in some cases rules to follow dictated by the business itself. In a way they serve as constraints or restrictions for design, and because of that they shall be defined first. (Miller, 2019)

From the user perspective it is known what triggers the process and what is expected as an output, and the goal is to define what shall happen in between to reach the desired end result as well as how to handle the positive and negative variations in user behavior. For example, if the conditions have not been met, there shall be an error message shown. This approach requires an end-to-end thinking.

The functional requirements specify what the system must do to process the user inputs and provide the user with their desired outputs (Miller, 2019). Sometimes they are documented as textual statements. But this approach is hardly allowing to understand the connection between user actions and system response, and it makes it easy to miss important details. Since the goal is to design the process of how the user and system are going to interact, the great way to document it is to map it through the flowchart. A good flow chart allows to combine the user and system behavior together in one image and even map out the data flows and logic of the application. It also guides the thinking process allowing to see the undefined areas of the process.

Considering that the team is working on a multilayered software architecture, it is neccessary to define the input to each layer (or service on the layer), logic of how the input shall be processed on the layer and what will be an output for each condition met. Figure 11 shows three main layers of the multilayer software arhitecture that shall be covered within the scope of technical specification of the design story. It might not be an easy task, but having a consistent strategic approach to the complex problem makes it easier to define a proper solution.

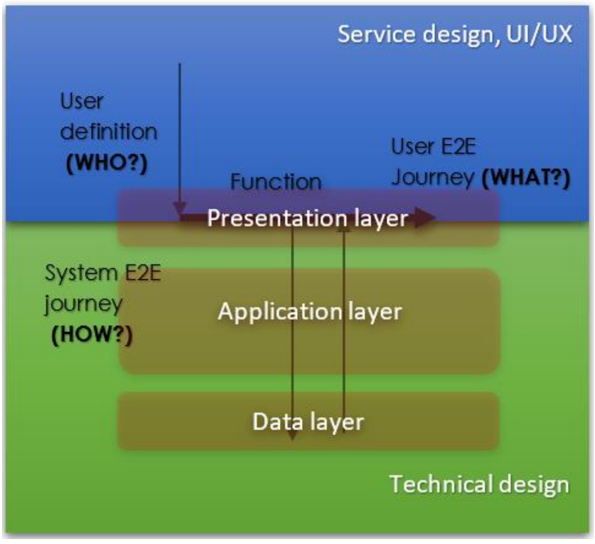


Figure 11. The scope of technical design for multilayer architecture

5.3 Other activities during the design phase

Many other actions can be taken during the design phase; for example, it is possible to evaluate the security of the solution using different threat modeling techniques. Communication design shall be considered as a part of the service design. When team is designing a function, it is time to define how this function must work; therefore, this is the best time to define how it shall be tested and come up with acceptance criteria. Involving the tester in the design process proves to be very valuable. If the tester is not reasonably involved in the design process, then it is a separate task to introduce him how it must work.

6 Implementation Phase

In this thesis the development topic will not cover any technology related aspects. But it will focus on the development task from the project process point of view.

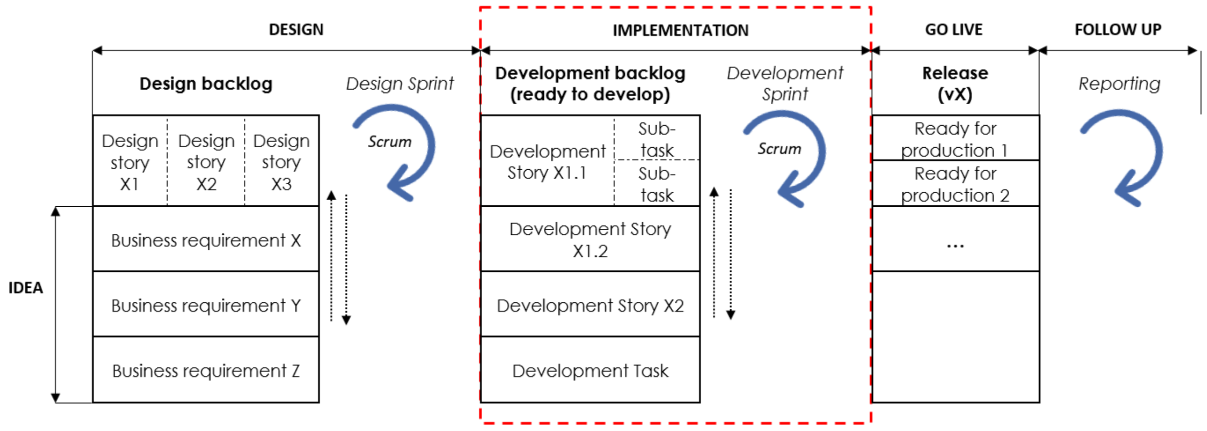


Figure 12. Implementation phase of the project lifecycle.

Implementation phase of the project is highlighted in Figure 12 and dedicated to working on the stories that have been prepared through the design phase to the point they are ready for development.

6.1 Development story planning

Design story sometimes might be grand and too large to be done at once; therefore, the process of the implementation again starts from breaking down the specification into development stories.

As has been defined above, development story represents a function or a part of the function. There are many techniques how to write stories, but focus is how much development one story can cover. The story is the piece of information that both developer and tester will rely on; thus, it must make sense for both. When building stories, it is important to consider that after the story has been implemented it will be tested from the user point of view (system is a user as well in some scenarios). That is why there is a limit to how far it is practical to break down the requirements into development stories for it to still be testable. On the other hand, story shall not be too big. If the story is too long to develop, the risks are that the story will not be finished within one iteration, the issues in implementation will take long to find or story will be too complex to test. In this case the feature might be broken down into set of stories, the amount and effort of which might be larger than what can be developed in one iteration.

6.2 Estimations and prioritization

Estimations are always a challenge, but after all the considerations and grooming the initial business requirement into design stories and later into development stories there should be enough information to make an educated prediction based on the prior experience. Estimates in this case are required to be able to evaluate the speed of the team and see how much work can be done within one Sprint.

The work of software development is rarely routine and repetitive which creates a problem of not knowing how much each task will take. In this particular case team is using story points, which is another unit of measure used to express the relative amount of effort. Story points can be used to guess-estimate each development issue, and through experience and collecting historical data those guestimates become more accurate.

Prioritization of development stories becomes easy too. If development is focused on the tasks relevant to the next release and next priority feature(s), developers can evaluate themselves in which order it is more logical to do the relevant stories.

6.3 Process control

Development work has several important parts to it: coding, writing technical tests, code-reviewing and testing. Implementation process is not linear, and that creates certain challenges for the process control. Figure 13 shows a very basic workflow through which functionality goes during implementation phase.

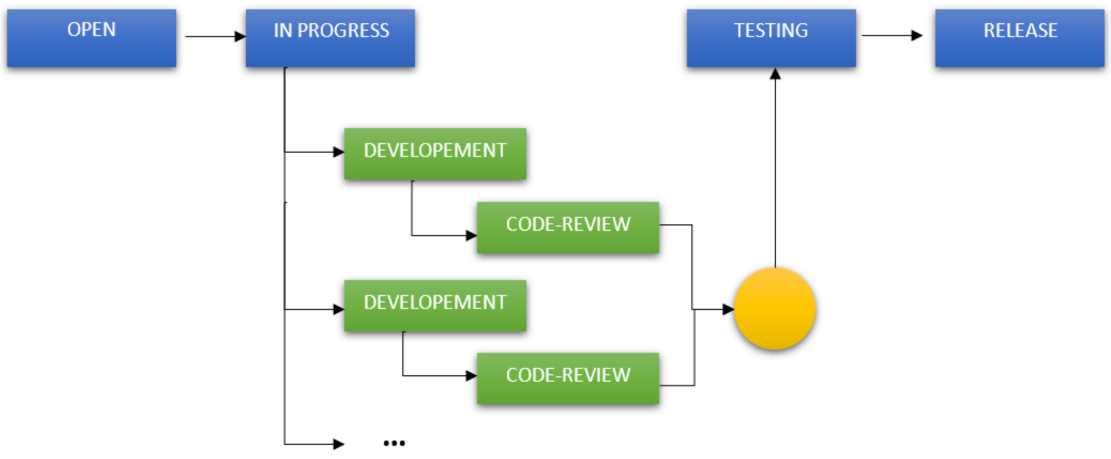


Figure 13. Process workflow of the development.

The reason why the process is not linear is because the development is done upon different layers of the system architecture that in some cases are built into separate development brunches for the easier version control. The tasks of coding and code-reviewing are individual per each development branch and process control model shall support this fact. On the other hand, when it comes to testing, normally it is required to validate the functionality works from end to end.

To address this complexity, development stories are used and are broken into sub-tasks. Story represents a functionality, and subtasks are development tasks that are merged to its own brunches. Figure 14 illustrates separation of the development workflow into story and subtask issue type.

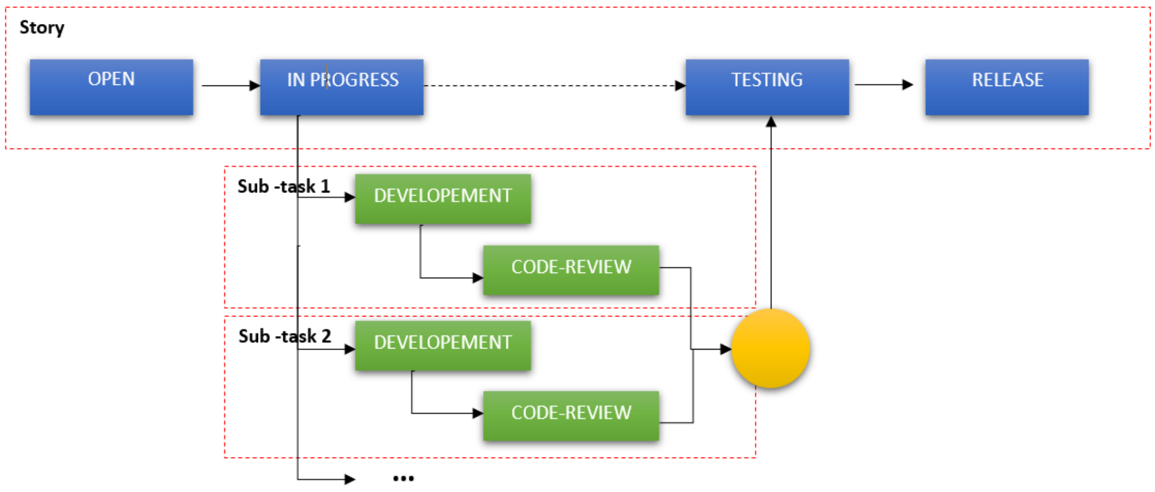


Figure 14. Separation of the development workflow into story and subtask issue type.

When nuances of synchronizing the process between process control tools and reality are considered, the workflows become more complicated and have more outlined statuses and transitions. Applying proper process control workflows allow to ensure that each story goes through the required activities to ensure the quality of the result. Figure 15 outlines the fully defined workflow of the development story, and Figure 14 describes the workflow of the development subtask.

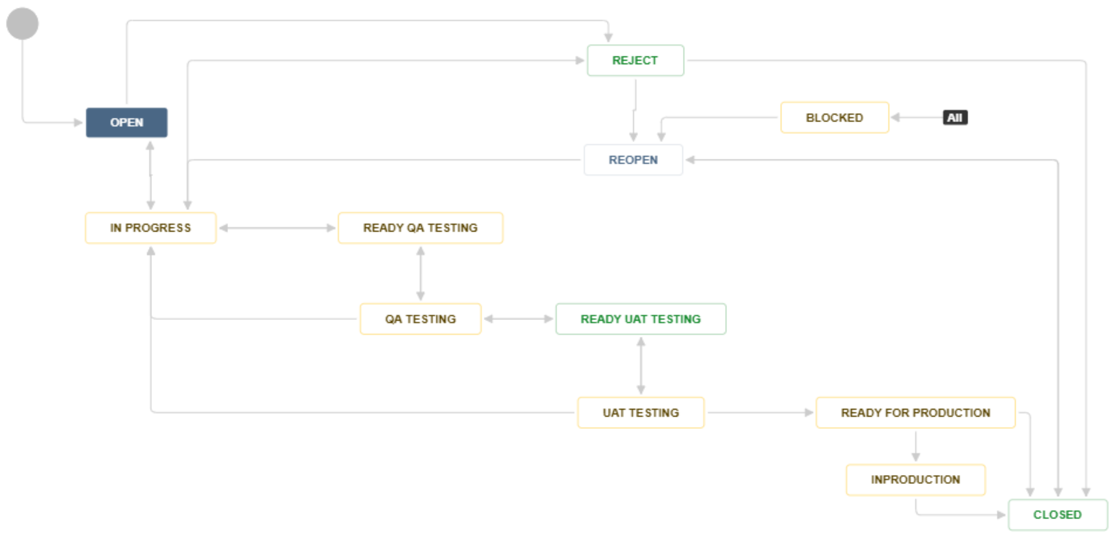


Figure 15. Workflow of the development story.

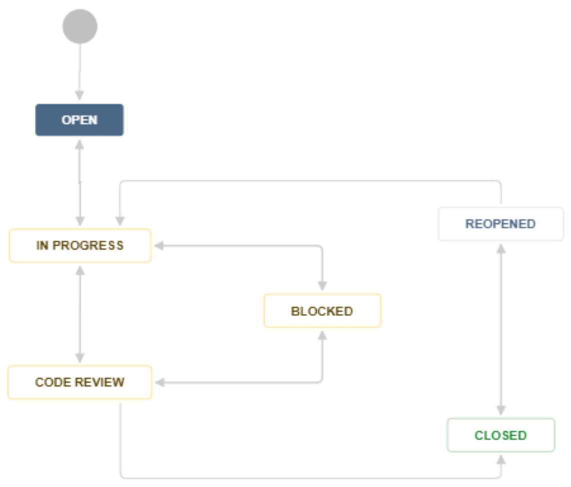


Figure 16. Workflow of the development subtask.

If principles described above have been applied to properly define scope of design and implementation, the result of this phase there will be a batch of stories that can be delivered to production, and thanks to E2E thinking during the design phase the output will be a working software as prescribed by the Scrum methodology.

7 Go Live phase

As the result of implementation phase(s) there will be stories that are *ready for production* or in other words ready to be delivered to the end user (illustrated with Figure 17). Once every story under the scope of the release is implemented, it is time to focus on the Go Live phase of the project. This does not mean that the implementation work must stop completely, but it means that at least part of the team shall allocate their time to focus on the delivery.

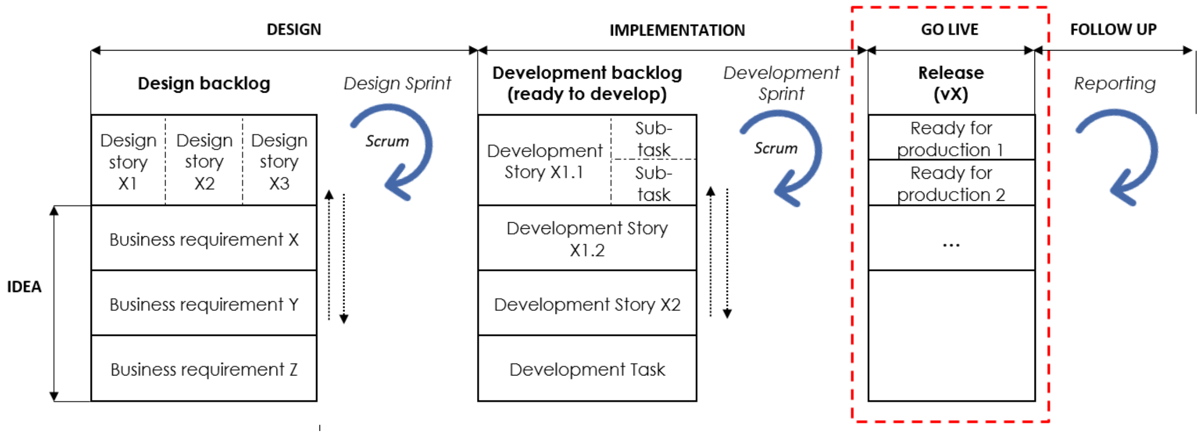


Figure 17. Go live phase of the project lifecycle.

When the release is ready from the development point of view, it is time to plan delivery to production servers. Deployment itself is not a complicated task once it has been practiced or in the best-case scenario autotomised, but there are plenty of other activities before, during and after deployment that shall be done. The best way to describe this process is a schedule of deployment preparation tasks as shown in Figure 18.

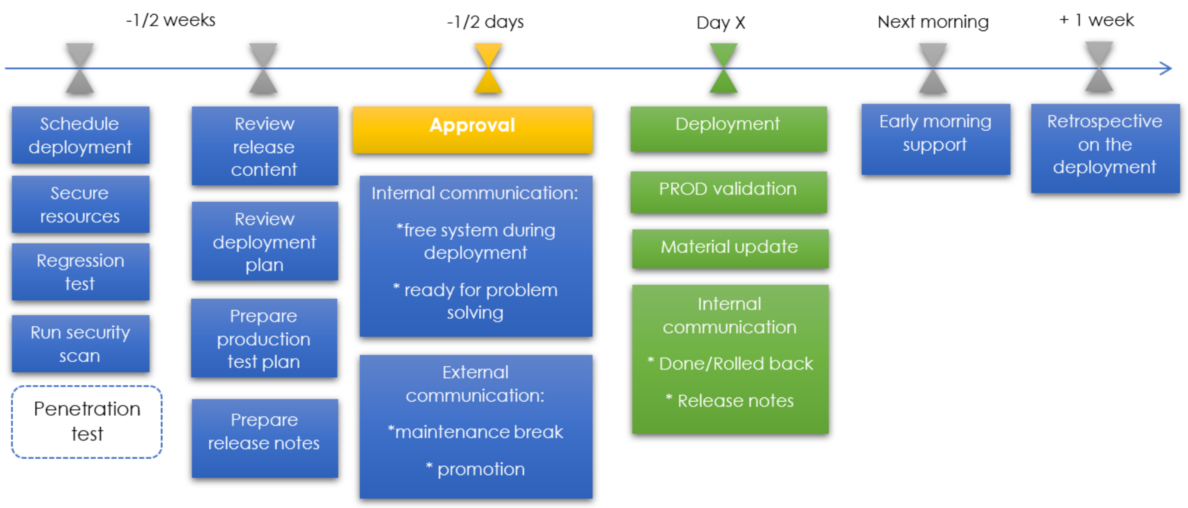


Figure 18. Schedule of deployment preparation work.

7.1 Deployment preparations

It is compulsory to schedule the deployment and secure the resources to perform the delivery. If the delivery requires a maintenance break, it means that it will not be done in the regular working hours or during the time when the service is heavily used. For that reason, the people involved shall agree on the time that meets the requirements of the service as well as personal schedules. It is good to do it as a first step of the process because knowing the target date will set a deadline for all other activities.

Even if the new functions have been thoroughly tested from the quality assurance point of view, it is not enough. Team needs to ensure that all previous development has not been negatively affected. To achieve this, a set of regression tests must be run on the release bundle.

To validate the security of the new release, another compulsory step is to run it through the security scan. If the risk is high, then penetration testing can be ordered from the external consultant.

Next task is to review the release content with the people whom it might impact; for example, training a customer service and an operations team about how new function will work and how to maintain it.

Deployments can normally be autotomised to the point it becomes an easy task, but even then, it is always good to review deployment plan on the subject of any variations from a normal procedure.

Another task is to prepare production test plan which is a combination of tests to validate the main functionalities as well as newly delivered functions. Release notes is a good document that can be used to communicate the content of the release more widely.

Approval or release sign of is the necessary milestone in the delivery process. It is a decision of whether the release can happen on the target date or if it shall be postponed due to certain issues found in the previous steps.

If release is approved, then one or two days prior to the delivery it is time to communicate it to the company and to the client. Internally people need to be informed that the maintenance break is coming and that they shall free certain systems involved from use as well as be ready to address the issues that unfortunately might occur. External communication meant to inform the users about break in service and also promote new features and functionalities of the new version of the application.

7.2 Deployment

Once the deployment itself is done, team needs to validate that version released to production works as it should with the help of the production test plan that has been prepared in advance.

If problems have been found, of course, it is good to first try to resolve them. If that proves to be impossible within a short time limit, then the team might choose to roll back to the previous version. The relevant parties shall be informed about the roll back.

But if everything looks well and functions properly, then it is all right to proceed further. Sometimes it is necessary to update the relevant materials; for example, new instructions, articles on the website or other supporting information outside the application. In this scenario the whole company can be informed about successful release, and release notes as a summary about the version content can be attached to the announcement.

7.3 After deployment

It is required to ensure that there will be specialist in place next morning to support the customer in case of the incidents that have not been detected on the deployment day.

As a part of the Agile practice it is good to review the deployment post factum, especially if something went not according to the plan. Retrospective is a good technique to learn progressively and improve the performance.

8 Follow up phase

The last phase of the cycle is follow up (see Figure 19). Once the functionality has been succesfully delivered to production, it starts to be used. It is important to see whether the customers are happy and comfortable to use it. There are two parts to follow up reporting: business follow up and technical follow up.

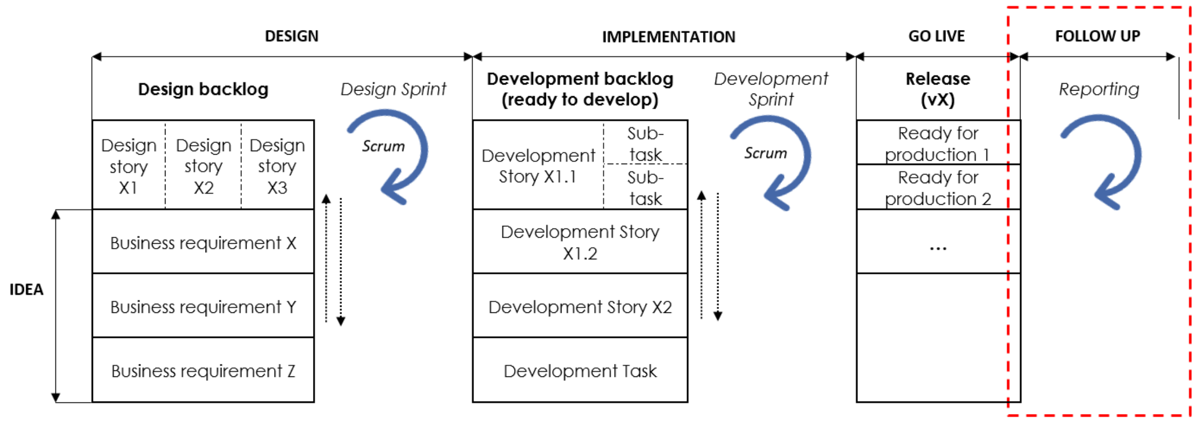


Figure 19. Follow up phase of the project lifecycle.

8.1 Business follow up

During the development of a new functionality many assumptions have been made. Business follow up is a way to confirm or contradict those assumptions.

During the idea phase KPIs that wanted to be followed have been defined, during design and implementation phase it was ensured that the data required for the measurements can be collected. Another way to evaluate the performance of a new functionality is to collect a direct feedback of the user through different channels. Now all that must be done is to gather the data, analyze and adjust the plans if needed.

8.2 Technical follow up

Technical follow up is about observing the performance of the application and predicting the issues before customer starts to find it inconvenient to use the application.

One method is monitoring the logs for the errors and measuring the performance (response times of the user interface, CPU and RAM loads of the servers). Technical follow can also provide valuable insights such as whether functionality is being used, at what time of the day, and on which devices, browsers, or operating systems.

9 Discussion

The process defined in the previous chapters has proven to be very functional. Even though there is always a room for improvement, it is a good base to build upon. One of the main benefits is that it allows an easy way to priorities the work. The top business requirement become a very clear target, and all the work whether it is design or development shall be focusing on reaching the target in scope and then repeating the same actions when moving to the next business requirement. This approach helps the team to stay focused and protects it from being overwhelmed by tall order.

One of surprising advantages gained from specifying issue types is improvement of communication. Each type represents a certain level of communication and has its target group. For example, it is very easy to priorities the roadmap of business requirements with stakeholders, protect the business people from irrelevant technical details by communicating on the level of design stories and create well defined development stories, but leave the specific detail of the implementation to the developer who can document those as sub tasks of the story. Applying the standard process also makes the communication easier in a sense that everyone knows what to expect from each other and what needs to be done to keep the work flowing from phase to phase.

Using Agile principles in the project management process of each phase of a lifecycle has allowed to define a way to flow through each step in a more rapid, but organised manner. Focusing on the user function and end to end system journey allows to get work done, actually develop testable code and deliver working software to production more often. What is as important is that it creates a sense of accomplishment, which is good for the team spirit. It also gives a possibility to re-plan as frequent as necessary based on the new knowledge without unnecessary straggle.

Proper handling and prioritization of the development ideas allows to ensure that the time is spent on the work that will bring truly valuable results for the user. Process is customer oriented and outlines the importance of collecting feedback around service design and graphical user interface, but it also highlights the need to gather feedback retrospectively (once the function is being in use) to ensure that the customer needs have been properly interpreted.

Most of the work in the process concentrates around the design phase. It outlines all the necessary steps to produce good designs (graphical and technical) as well as the importance of user and stakeholder validation to be part of the design process.

The process gives a very good visibility and understanding what it is going to take to implement a chosen functionality, determine the order of development and the content of the release. It allows to plan just in time and just enough without spending too much time on inaccurate predictions.

Having clear stages which functionality shall undergo, allows an easier process control and understanding, which leads to the improving the continuity of the flow through this actions and opportunity for the team to be able to work simultaneously on several goals without being lost and feel out of control. The ambition of doing the right thing at the right time eliminates the repetition and redundancy of the work.

But, certainly, there are difficulties as well. One difficulty is for the team to learn how to apply the process due to the fact that there are many ideas gathered together, but it has proven that it can be applied gradually and can be adopted well after couple of iterations. The other drawback is that the business requirements and corresponding designs can be so different in nature that it is not possible to standardize every possible scenario. The advice here is to stay as close to the process as possible, but always use a common sense and learn to adjust a process to serve the current needs.

10 Conclusion

The goal of this thesis was to improve the current process to accelerate deliveries, share limited resources among projects in efficient way, remove wasteful processes, and as result of all the improvement activities, enhance the quality of the final product.

To reach the goal, there has been two objectives set:

- To define the standard process for developing new projects and functionalities through creating customized project lifecycle.
- To apply agile thinking to the phases of the lifecycle where it is appropriate, to define a process and output for each step of a lifecycle.

The process is a sequence of steps that are logical to do in a certain order to achieve the desired result in the most efficient way. Throughout this project the sequence of logical steps to make a complex task of developing a new function more manageable has been defined. Approach addresses the routine tasks of the development; therefore, it is applicable to all software development projects.

The phases of the idea and design has been defined with the customer in mind. The process is design oriented, continues since it focuses on the full cycle of the development and can be scaled from a small project with limited resources to the large projects, such us developing new applications from scratch. These qualities raise the efficiency of the team work. Improved visibility of the project makes wasteful actions become very observable. There are strong reasons to believe that the quality of the end product will be positively affected once the process is applied to the project work.

There are always a room for improvement, but implementing the principles described in this thesis will allow to get the job done more efficiently and remove unnecessary stress from working on the challenging development tasks.

References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D., 2019. Manifesto for Agile Software Development. [online] Agilemanifesto.org. Available at: https://agilemanifesto.org/ [Accessed 18 Feb. 2019].

Cobb, C., 2013. Making sense of agile project management. Hoboken, N.J.: Wiley.

Dr. Rico, D., 2019. Demystifying Scaled Agile Frameworks.

Highsmith, J., 2019. History: The Agile Manifesto. [online] Agilemanifesto.org. Available at: https://agilemanifesto.org/history.html [Accessed 18 Feb. 2019].

AccountingCoach, LLC, 2019. Accounting coach dictionary. [online] AccountingCoach, LLC (US). Available through: <https://www.accountingcoach.com/terms/O/opportunity-cost> [Accessed 18 Feb. 2019].

Knapp, J., Zeratsky, J. and Kowitz, B., 2016. Sprint. Riverside: Simon & Schuster.

Measey, P., Levy, R. and Short, M., 2015. Agile foundations. Swindon, UK: BCS Learning & Development Ltd, a wholly owned subsidiary of BCS, The Chartered Institute for IT, pp.1-15. Schwaber, K. and Sutherland, J., 2017. The Definitive Guide to Scrum: The Rules of the Game. [ebook] Ken Schwaber and Jeff Sutherland. Available at: https://www.scrum.org/resources/scrum-guide [Accessed 7 Feb. 2019].

Miller, R., 2019. What are requirements types? [online] SearchSoftwareQuality. Available at: https://searchsoftwarequality.techtarget.com/answer/What-are-requirements-types [Accessed 8 Dec. 2019].

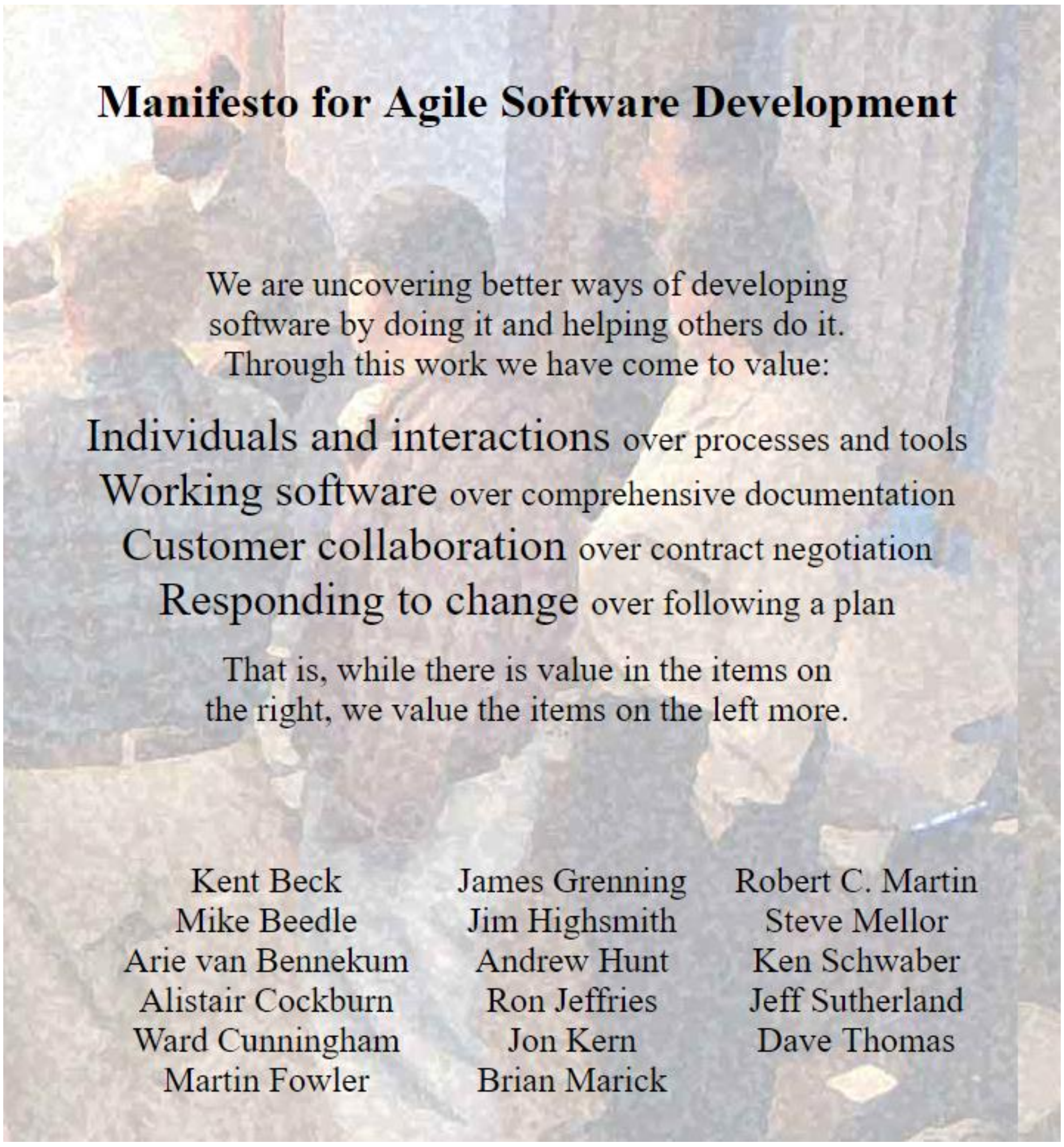
Sasal, D., 2019. Project Management Simplified: Learn The Fundamentals of PMI's Framework. [video] Available at: https://www.youtube.com/watch?v=ZKOL-rZ79gs [Accessed 12 Dec. 2019].

Talley, J., 2019. What Does a UX/UI Designer Do? [online] www.mediabistro.com. Available at: https://www.mediabistro.com/climb-the-ladder/skills-expertise/what-does-a-ux-ui-designer-do/ [Accessed 27 Dec. 2019].

The Standish Group Report., 2019. CHAOS Report. [online] The Standish Group. Available at: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf [Accessed 19 Feb. 2019].

Manifesto for Agile Software Development

Appendix 1. Manifesto for Agile Software Development (Beck et al., 2019).



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:

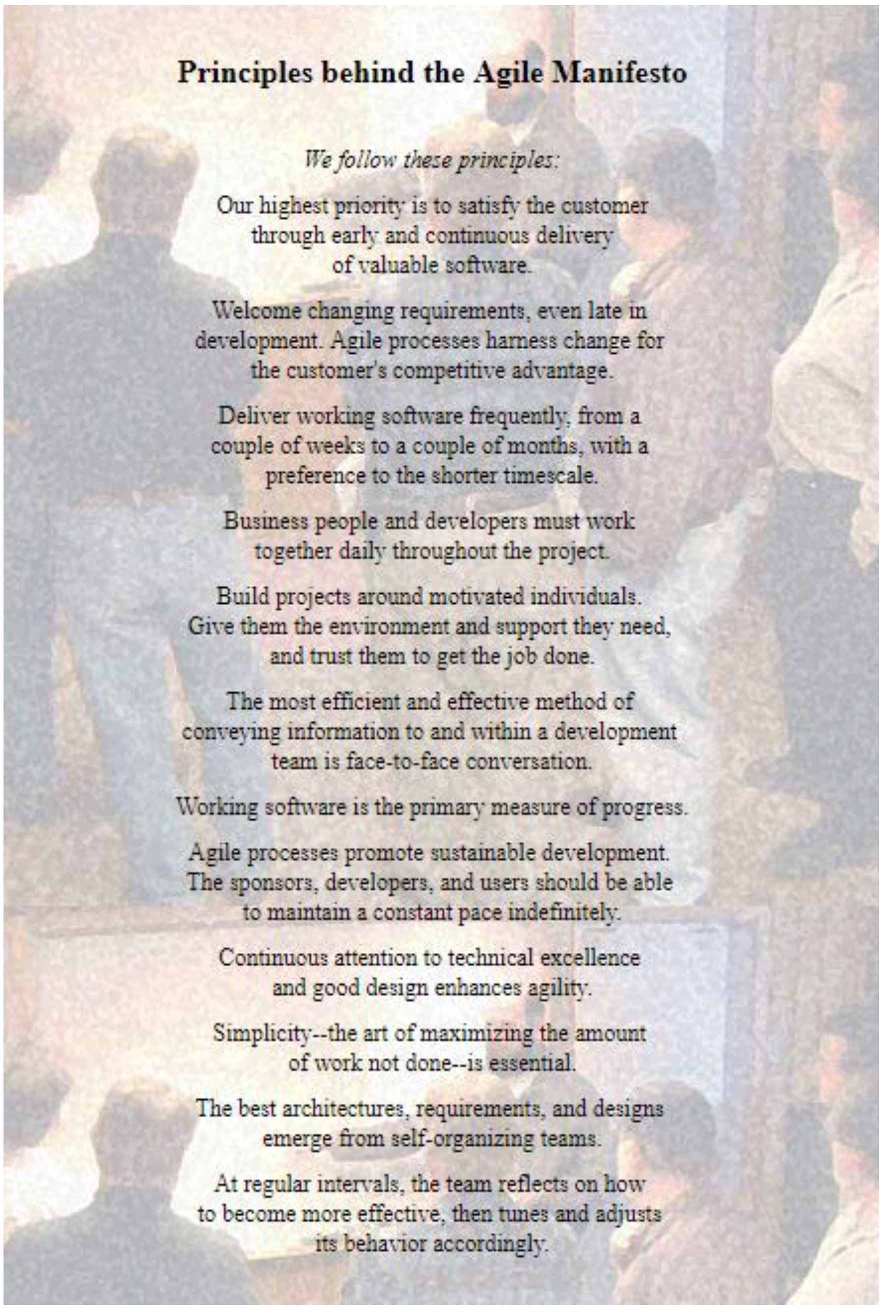
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

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

Principles behind the Agile Manifesto

Appendix 2. List of Principles behind the Agile Manifesto (Beck et al., 2019).



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.