Tallinn University
Institute of Informatics

Applying Agile Methodologies to Design and Programming

Master Thesis

Author: Tatjana Pavlenko

Supervisor: David Lamas

Author	Supervisor	Head of the Institute
(name, date and signature)	*(name, date and signature)*	*(name, date and signature)*

Tallinn 2012

Author's Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any other comparable academic degree. Where other sources of information have been used, they have been acknowledged.

. . . .

(date) *(signature)*

Abstract

This master thesis covers problems of applying Agile methodologies to design and programming. The paper is focused on Company Sigma adopting the most popular Agile framework – Scrum. The research addresses problems of introducing Scrum in a small company which has never used any particular methodology. Additional problems resulting from company specifics are the lack of team self-organization and inability to manage designing process.

The purpose of this work is to design an effective Scrum approach for Company Sigma, which can be easily adapted and passed to future employees. The approach corresponds to company's objective and fits the needs and skills of team members.

The study has been conducted within the frames of design research, where the designed artifact is implementation of Scrum into specific environment. Design research consists of four implementation cycles and is enhanced with ethnographic approach in order to receive reliable feedback from participants. The research is based on theoretical overview of Agile methodologies. The strategy of this study was inspired by literature covering design research.

The results reveal an effective Scrum approach that has a core idea to involve as much team members as possible, keep tracking others and slack their resistance. The list of methods applicable to Company Sigma environment is presented in the end of the paper.

Keywords

Agile, Scrum, design, programming, application, software, iterative, incremental, linear, development, methodologies, management, user, interface, approach, team, implementation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A comprehensive manual for developing iOS applications starts with the promising statement: *„Everybody has an idea for an app"* (Welch, 2011, p.3). On July 7, 2011 Apple Inc announced that over 15 billion applications have been downloaded from App Store by more than 200 million iOS users worldwide. The App Store offers more than 425 000 applications and developers have created over 100 000 native iPad applications (Apple, 2011). Companies that produce software should adjust to these changes smoothly and fast.

This paper studies the case of Company Sigma[1] that has been developing large Windows software for specific target group since 1990s. Recently their product became outdated due to extreme development of other software platforms. Therefore customers requested the same functionality with better interface. An ideal solution was to produce a set of iPad applications.

However, it appeared to be very challenging. The company had an idea but did not have a corresponding strategy. Even though leading developers had twenty years of experience in information technology, they have never done iOS application programming before and have never spent much effort on designing

[1] Company office is located in Tallinn (Estonia), but its business details remain confidental throughout the research and no real names are used in this master thesis.

user interface. To manage the situation, director of Company Sigma decided to increase productivity by trying new sofwtare development methods.

Scrum was the most corresponding framework, which belongs to Agile type of methodologies. According to annual survey *The State of Agile Development* (VersionOne, 2012, p. 7), three top benefits obtained from implementing Agile are: *ability to manage changing priorities, improved project visibility, and increased productivity*. These results motivated Company Sigma to start being Agile too. However, this was not a smooth and simple process: *"transitioning to Scrum and other agile methods is hard – much harder than many companies anticipate"* (Cohn, 2010, p. 3).

1.1 Research Problem

This research is focused on Company Sigma adopting the most popular Agile methodology – Scrum (VersionOne, 2012). As describe above, the company's problem is rooted into several layers: transforming outdated software into iOS application, requiring new technical knowledge and new strategy. Our research is focused *only* on the problems of adapting new strategy.

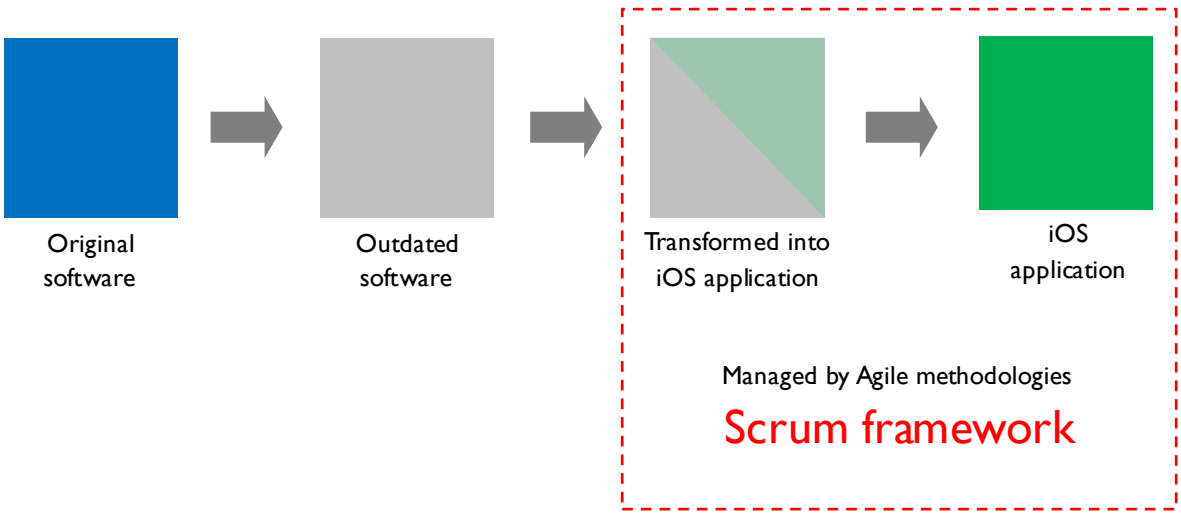


Figure 1.1 – Alterations in Company Sigma development process

As it can be seen from Figure 1.1, we are interested in the part contoured with

dotted red line – process of applying Agile methodologies, Scrum in particular, to design and programming of iOS application.

Not only acquiring technical knowledge of iOS development was problematic for Company Sigma. Completing tasks according to methodology was not easy either. And it is not the unique case of Company Sigma. Cohn (2010) confirms that "*transitioning to Scrum and other agile methods is hard – much harder than many companies anticipate*" (p. 3). According to annual survey *The State of Agile Development* (VersionOne, 2012, p. 5), 8% of companies using Agile said they do not plan to implement these methodologies for future projects, 33% said they do not know. The most common obstacles in adopting were: *inability to change organizational culture, unavailability of personnel with right skills, general resistance to change.* Some of these factors existed in Company Sigma as well.

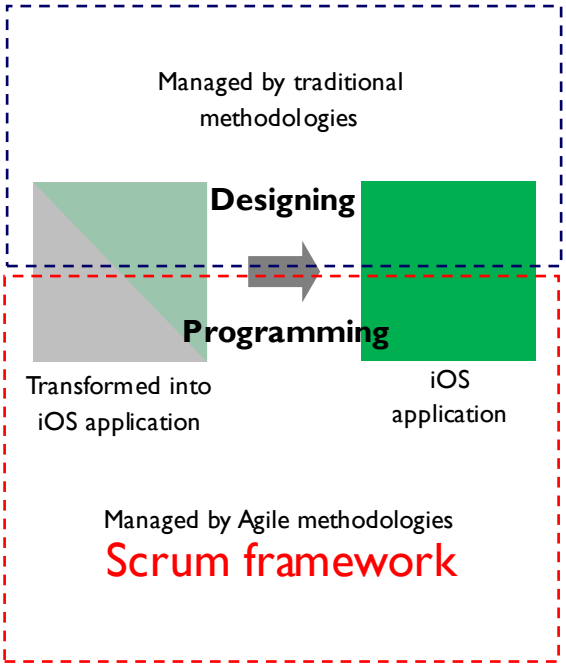


Figure 1.2 – Problem of fitting design into Scrum in Company Sigma

Later there appeared the second problem: fitting design into Scrum, as illustrated in Figure 1.2. Company did not have experience in designing graphical user interface and hired a freelance designer to fill in the gap. However, the designer

expressed resistance to working iteratively. This made Scrum adaptation even more challenging. And again, such problem is not the unique case of Company Sigma. According to Cohn (2010), designers often have a legitimate concern with adopting Scrum. This happens mostly because Scrum framework involves working iteratively, which is not appropriate for designers who prefer working in advance of the rest of the project. Additionally, the problem is that the origin of Scrum lies in software engineering where visual appeal didn't really matter (Arslan, 2012).

Thus Company Sigma has two major problems: (1) adapting Scrum and (1) managing user interface design. However the research problem is slightly different:

How to design an effective Scrum approach for Company Sigma

But in the end, it leads to solving company's problems anyway.

1.2 Research Strategy

Based on the introduced problems, the following research strategy helps to achieve sufficient results and satisfying solutions:

1. Study working environment of Company Sigma
2. Study what is preventing Company Sigma from following Scrum principles
3. Propose an effective Scrum approach
4. Implement the design of effective Scrum approach.

First of all, the designed approach allows passing it to new members of the company, so that Agile methods can be sufficiently used in future. In addition, the collected data may be helpful for other companies facing the same problems.

Additional goals involve company's interests, such us improving the work

process, making the development faster, settling productive working atmosphere, and other small changes, resulting from the conducted research, which may lead to increasing the business value.

1.3 Methodology

The study has been conducted within the frames of design research, where the designed artifact was implementation of Scrum into specific working environment of a Company Sigma. The research consisted of three constantly overlapping phases: 1) highlighting the weak points of adopting Scrum 2) designing new implementation of Scrum, and 3) evaluating the adopted implementation.

To reveal Scrum-related problems, design research was enhanced with ethnographic approach, which allowed observing characteristics of the team in Company Sigma. As far as the researcher is one of the team members, constantly engaged in the observed activities, it was sufficient to use participant type of observation (Cohen, Manion, and Morrison, 2007). The instruments employed to collect data were:

1. open and semi-structured qualitative interviews
2. online conversations
3. card sorting games
4. personal constructs (repertory grid technique)
5. observations
6. photographing

Interviews were conducted with 5 team members selected out of 8 according to their involvement in the current project. The most relevant people were Designer,

Product Owner, Junior Developer[2], Senior Developer, and Software Architect. Interviews were audio-recorded and transcribed. The audio transcriptions can be found on a CD enclosed to this master thesis. Online conversations are also documented and filed there.

Interview types were taken from Patton (1980): an interview guide approach and standardized open-ended interview. Basic open-ended questions were determined in advance, however their wording and sequence was decided during the interview. To define individual identities we asked: "What is your role in the project?", "Describe your job as if you are speaking to a 6 year old", etc. For Scrum evaluation, there were questions like: "To what extent does Scrum work according to 100% scale?", "Do you think there is enough collaborative tools?", "What could you do to keep Scrum going?" etc.

Such approach was suitable for our design research, because it guaranteed conversational and situational style of the interviews, so that respondents have not felt tension while speaking. We gave people freedom to express themselves. Even though received information was massive and covered many aspects, the same topics were easily detected within every interview. Interviews were thematically coded according to typological classification system of Lofland (1950): settings, acts, activities, meanings, participation, relationships. Settings (entire context) remained the same within each interview and coded as following:

- Role in the project
- Attitude to Scrum
- Current situation
- Towards ideal situation

[2] Junior Developer quit his job during the final phase of this research. As a result, interactions slightly changed; the rest two developers (Senior and Chief) shared the tasks between each other. Distantly working Chief Developer became more involved. Nevertheless, we count Junior Developer as a team member, since he has been working most of the time during the research.

The full list of codes is presented in Annex A. In addition, we used Tag Clouds for visual representation of textual data. Those can be found in Annex B.

Traditional problem-solving approach helped to maintain the implementation phases. Constant alterations were made within Company Sigma. During the whole study period, Scrum technique was reshaped several times either by managing director or by author of this research. New alterations had to be implemented, nevertheless previous instruments still stayed unused. The researcher's interest was to separate alterations from each other. Those alterations are objects of our design research. Evaluation was done by analyzing interviews, personal constructs, and team's overall productivity. Plus, several tables and diagrams were used to track the workflow and observe any improvements.

A set of photos was made in the office to illustrate the work process and reveal the attributes (Task Board, Burndown Chart, schedule etc.). The researcher took notes during weekly meetings of the team, where every participant expressed his ideas, problems, and suggestions for further development.

As a whole, 5 face-to-face interviews, 2 online interviews, 5 questionnaires, 9 weekly meetings, 27 pictures, and 31 pages of notes were collected and observed since the research process has started. Three of five interviews were conducted in native language of speakers, which is Russian. The codes for those interviews are available in English. The type of gathered data was nominal. The whole research period took ca 4 months and was split into overlapping stages: general observations (generating concepts, literature review, sorting ideas – ca 2 months), deeper analysis (defining what is working, what is not working and why – ca 2 weeks), constant implementing and evaluating the design (ca 1 month).

Chapter 2

Agile Software Development

This chapter highlights Agile methods of software development and their differences from each other. First, we give an overview of classical methodologies and continue with introducing an Agile approach. After that, paper will focus on Scrum, one of Agile methodologies. The main issues for discussion are: which techniques are covered with Agile framework, is Agile a synonym to Scrum, can Scrum bridge the gap between designers and programmers.

2.1 Classification of Software Development Frameworks

Classically there have been three types of methodological frameworks: linear, iterative, and combination of both. The term 'linear' means *"progressing from one stage to another in a single series of steps"* (The New Oxford American Dictionary, 2010). This exactly describes the methodology. The most common linear framework is Waterfall, proposed by Royce (1970), where projects consist of sequential phases with acceptance of some overlap. Each step in a waterfall process must be completed before moving on to the next (Sims & Johnson, 2011). The customer can see the product as soon as the last stage is over. Additionally, as proposed by Royce (1970), there should be "*quite a lot*" of documentation (p. 332). Such linear methods are also called 'plan-driven' because they need a set of

requirements predefined from the start. The requirements should be precise, clear, and relatively static (Williams, 2007).

Iterative development is quite opposite. Unlike plan-driven linear methods, it excludes initial planning but focuses on constant changes, and stimulates continuous revision and improvement of software. The work is broken up into small pieces that are developed over some period and finally put together when they are ready (Cocburn, 2008). An example of pure iterative framework is Prototyping (Centers for Medicare and Medicaid Services [CMS], 2008). Iterative frameworks can be also used in combination with linear methods, setting up such frameworks as Incremental, Spiral, Rapid application development (RAD), and Extreme Programming.

Researchers Larman and Basili (2003) have studied iterative development together with incremental development and treated them as a whole (IID). Cockburn (2008) supports this idea and believes these two branches *"fit well with each other"* (p. 28). Incremental approach improves development process, iterative approach increases product's quality.

Comparing to linear process, iterative incremental development is far more popular and widely applied in software companies today. Its main advantage is flexibility, which is very important in terms of extremely developing software industry and software technologies (Williams, 2007). Customers' expectations are moving quickly and become unpredictable, that is why sticking to a static plan, as suggested by Waterfall method, may lead to frustrating results.

2.2 Introducing a Buzzword

Until now, we have not yet mentioned Agile. It is important to understand the chronology of developing Agile methods. Before the word 'agile' became so

widely used in software industry, several iterative and incremental methodologies have been already practiced since 1970s. Scrum was launched in 1986, Rapid Application Development (RAD) in 1994, Extreme Programming (XP) in 1996 (Larman & Basili, 2003).

Agile is an umbrella term that covers Scrum, RAD, XP, and other *'lightweight'* methodologies, such as Crystal, Lean, Kanban, Feature Driven Development, etc. The term 'agile' was introduced in 2001 when seventeen enthusiastic software developers, interested in further promotion of quick and easy techniques, created a movement opposed to classic linear Waterfall method. They formed an Agile Alliance and wrote an Agile Manifesto (Sims & Johnson, 2011).

Since then, Agile has been awaking high interest among IT companies. It is indeed a very popular iterative and incremental approach to software development. Some teams came across it accidentally; some were intentionally searching for a new strategy. Both ways, Agile brings success if adapted properly. There are plenty of online groups and communities for practicing Agile methods; special events and presentations are organized in order to meet in person and share the experience (e.g. Agile Saturday in Tallinn and Riga). Teams want to be Agile.

At the same time, it remains unclear what 'agile' is all about and how its methodologies are different from each other. At some point 'agile' sounds like a 'buzzword', something very important and constantly heard, but difficult to understand since there is no common meaning for it (Jensen, 1998). There is no lack of books, manuals, educative videos, slideshows, and training courses regarding Agile, but this diversity makes it harder to find a unique interpretation. However, in terms of Agile, various interpretations may and should coexist. According to Cohn (2010), if someone has read a book about Agile and thinks he found the right approach for his company, he is wrong. In reality, there is a special way for each organization to become Agile.

2.3 Does Agile Equal Scrum?

It is quite common to say 'agile' in reference to Scrum, and vice versa. Agile experts also interchange these terms, for example, Cohn (2010) or Rasmusson (2010). Cohn treats Agile and Scrum as comparable concepts in his book *Succeeding with Agile: Software Development Using Scrum*. Rasmusson's guide *Agile Samurai* includes many Scrum strategies (i.e. compare to Sims & Johnson, 2011). Also, if we type 'scrum' into Google Books, the following titles will be displayed: *Agile Project Management with Scrum, Agile Software Development with Scrum, Agile Game Development with Scrum.* In practice, Agile and Scrum may be confused. During the interviews, enclosed to this paper, one respondent has wondered whether these terms can be used as synonyms. There is a recent online discussion in a blog (Pledgerwood, 2012) where the author claims: *"...it's very annoying when people assume that everything I say or do or put on a profile regarding Agile is actually about Scrum"*.

There is no surprise for such substitution, because Scrum is the most used Agile methodology by 2011. This tendency remains stable during the last six years (see Figure 2.1). Figure 2.1 is based on six annual surveys, conducted by VersionOne, *The State of Agile Development.* It also shows that Scrum has been practiced together with Extreme Programming since 2007, which makes it even more overwhelming.

As discussed before, Agile is a much wider concept than Scrum, it is a set of values and principles, whereas Scrum is one particular methodology based on those values and principles. However, Figure 2.1 illustrates how big the proportion of companies using Scrum is, comparing to other Agile methods. Therefore, if someone says Scrum instead of Agile, or vice versa, the mistake is relatively small, for in most cases these terms are indeed equal.

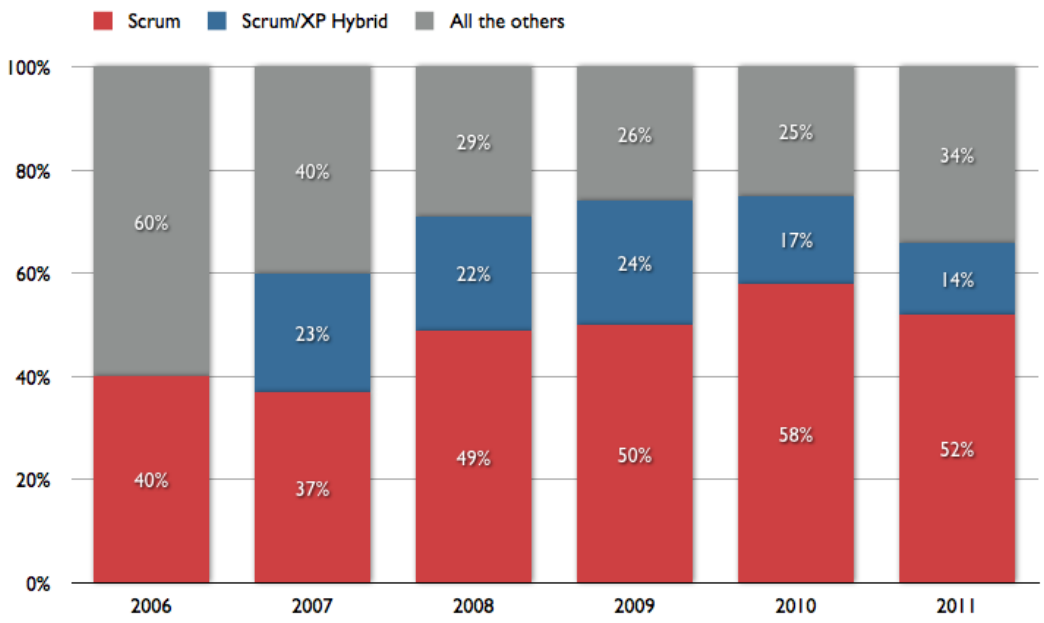


Figure 2.1 – The percentage of implementing Scrum among other Agile methods during the last six years (based on six annual surveys by VersionOne, 2007-2012)

Of course, we cannot literally replace them without knowing the difference. In this paper we refer to Scrum as a sub-term of Agile and do not interchange them. Moreover, there are practices and instruments relevant only to Scrum, which cannot be generalized.

2.4 Core Principles of Scrum

What singles out Scrum among other Agile methodologies is that it is not a strict methodology, not a system of methods, but rather a team-based framework, which relies on self-organizing cross-functional teams (Mountaingoatsoftware, 2012). It is very important to select the right members and maintain teamwork sufficiently. That is why Scrum is concentrated on introducing new roles and shaping the old ones (Cohn, 2010). In order to make team self-organized, the process should be supported by flexible schedule, useful artifacts, and shared terminology. Figure 2.2 shows the development process of Scrum framework, which is usually called *Sprint Cycle*.

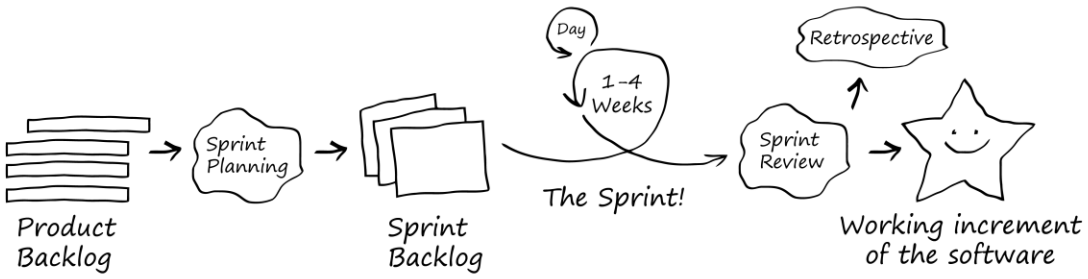


Figure 2.2 – Sprint Cycle

Sprint is an iteration, or development period, that lasts no longer than 1-4 weeks. *Product Backlog* is a set of business and technical functionality that has to be developed or revised during the whole release period. It is constructed by Product Owner and includes features, bugs fixes, documentation changes. Sometimes Product Backlog is called 'backlog items'. *Sprint Backlog* is a set of business and technical functionality selected by Product Owner from the product backlog for the next sprint. *Sprint Review* is a meeting organized at the end of a sprint, when the increment of working and potentially shippable software is presented. *Retrospective* is also a meeting on the same day as sprint review, which involves everyone discussing the strong and the weak points of the previous sprint in order to improve the mistakes within the next sprint.

User Stories are building blocks of the product. They are expressed in a simple language, as if the future users have told them. User stories are written during the specially organized meeting by Product Owner and team members. Stories are split into smaller tasks that should be completed during each sprint. *Sprint Burndown Chart* shows the hours or points remaining to completed tasks for a current sprint. *Task Points* are usually calculated during the Team Estimation Game or Planning Poker. It makes easier to estimate different assignments and prioritize them.

Scrum Master is a team member who also performs as a mentor or coach of the team. His main responsibility is to keep Scrum working: reinforcing product

iteration, goals, values, and practices. He coordinates Daily Scrum Meetings and Sprint Reviews. *Product Owner* represents customers' needs, creates and prioritizes Product Backlog, selects items for a Sprint. Product Owner and Scrum Master should be two different people. All the rest are team members: developers, designers, architects, testers, etc.

Basically, the whole Scrum is built upon two poles: team members and customers. There is a product in between, which should be somehow delivered from one pole to another. Scrum is responsible for that. Cohn (2010) notifies that introduction of Scrum affects not only developing team but everyone involved in the project, even the financial department. Of course, customers are also affected, since they constantly receive an increment of working software, not the completed version right away, as it traditionally was.

2.5. Technical Practices

The crucial difference between Scrum and linear Waterfall method is no consequence in analyzing, designing, coding, and testing – they are applied altogether to each increment of a product, as a set of mixed mini-Waterfalls. When all the increments are complete, customer receives the final version. But before that, he is able to monitor middle stages. Scrum is what makes software production look transparent and visible. As a result, all mistakes and inaccuracies can be noticed at an early stage, rather than in the end, when nothing can be changed.

Scrum suggests using different technical practices for making problems even more predictable and avoidable, these are: release planning, refactoring, project micro-charter, test-driven development, pair programming, collective ownership, continuous integration, and also methods, introduced by Cooper (2007), such as user stories and paper prototypes.

There is quite enough freedom for combining technical practices in Scrum. The only assumption is that teams should definitely use at least some of them (Cohn, 2010). As long as teams are self-organized and well-directed by Scrum Master, they can choose between various techniques according to company's needs, working environment, convenience, and their goals in general.

2.6 Guiding Values

Besides the technical part, there is also a set of values, which Scrum team should appreciate and remember while coding and designing. Such values are taken from *Agile Manifesto,* created by members of Agile Alliance.

1.	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2.	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4.	Business people and developers must work together daily throughout the project.
5.	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7.	Working software is the primary measure of progress.
8.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9.	Continuous attention to technical excellence and good design enhances agility.
10.	Simplicity - the art of maximizing the amount of work not done - is essential.
11.	The best architectures, requirements, and designs emerge from self-organizing teams.
12.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Table 2.1 – Principles of Agile Manifesto

Agile Manifesto includes twelve principles proposed for companies as a starting point to Scrum (Sims & Johnson, 2011). These principles are published on the website Agilemanifesto.org and listed as shown in Table 2.1.

In addition, there are 4 general values that can be illustrated as a concept map (see Figure 2.3). The main idea is to work opposed to classical software development based on planning, documentation, contracts, and tools. For a Scrum team, it is more important to interact with each other and with customers, whereas the 'old' tactics of negotiation should be remembered but not prioritized.

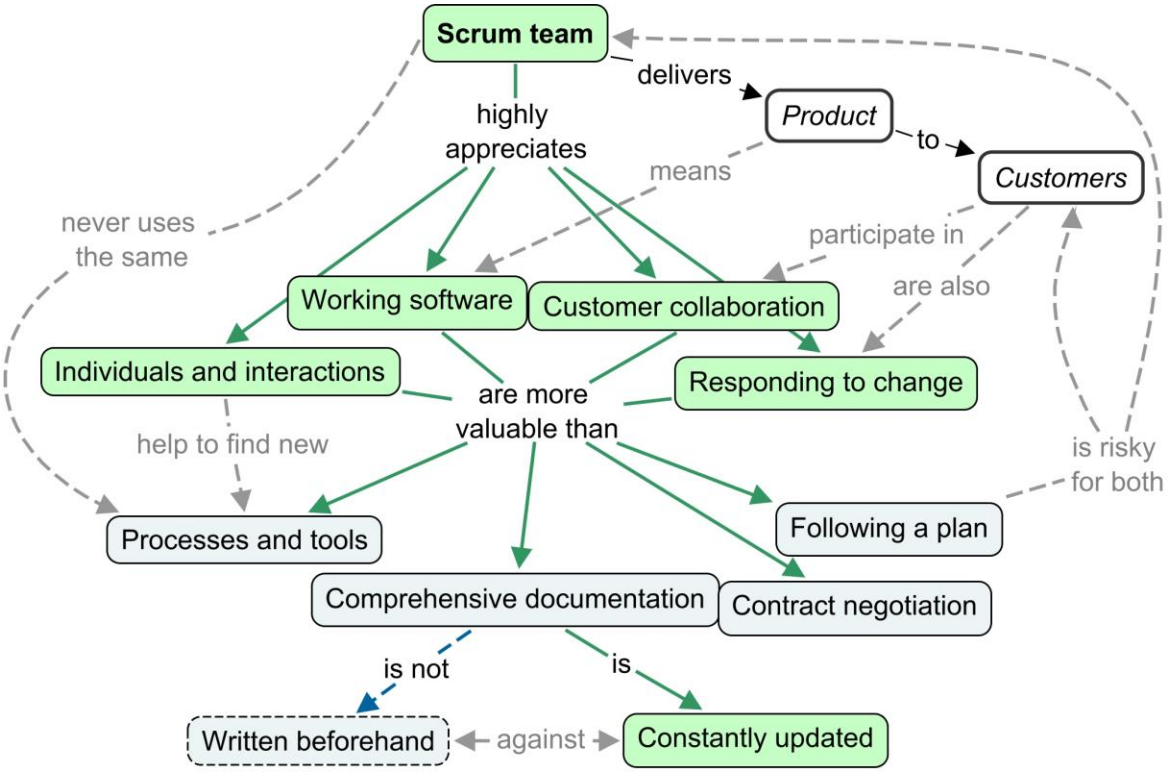


Figure 2.3 - Values of Agile Manifesto

2.7 Scrum in a Small Company

As discovered above, teamwork is very essential for adopting Scrum. However, there are cases of having only one small team, which cannot be changed. As demonstrated by VersionOne in *The State of Agile Development* (2007-2012) surveys,

team size has never been the reason for failed Agile projects, which means that there are barriers harder than having a small team

The case of a small company trying to adapt Scrum is thoroughly discussed in Chapters 4-5. Before moving forward, it is necessary to discover the perspective of adopting Scrum in a small company.

The minimal size of a Scrum team is five, excluding Scrum Master and Product Owner. Different sources suggest nearly the same numbers, for example, Sims & Johnson (2011) say there should be *"seven, plus or minus two"* (p. 71). Cohn (2010) suggests five to nine people as an ideal team, but also proposes an approach used by Amazon.com: "*a team that can be fed with two pizzas*" (p. 177). There is a much wider discussion about geographically distributed teams than just small teams. The only notion is done by Sims and Johnson (2011): "*Fewer team member and the team may not have enough variety of skills to do all of the work needed to complete user stories*" (p. 71). The most essential thing is experience and ability to follow Scrum techniques, whereas quantity is the matter of individual performance.

To sum up, Scrum is suitable also for small companies; hence the challenge of adopting it can be accepted. During the interviews, conducted in Company Sigma, several members claimed that Scrum was impossible to run in their circumstances (small company, distributed team). Such statements are not valid within this research because the team of 6 members plus Scrum Master and Product Owner cannot be treated as small. And having developers working in another country is not a problem for Scrum (Cohn, 2010).

2.8 Design and Programming: Can Scrum Bridge the Gap?

First of all, the definition of design should be clarified. As noticed by experts on

design research Koskinen, Zimmerman, Binder, Redstrom, and Wensveen (2011), 'design' is an ambiguous English term, because it means both 'planning' and 'form giving'. The difference between outlook and functioning is quite pale: "*a plan or drawing produced to show the look and function or workings of a building, garment, or other object before it is built or made*" (The New Oxford American Dictionary, 2010).

However, designing interface and designing functionality is not about the same. As stated by Apple (2011) in their iOS Human Interface Guidelines:

> *A user interface that is unattractive, convoluted, or illogical can make even a great application seem like a chore to use. But a beautiful, intuitive, compelling user interface enhances an application's functionality and inspires a positive emotional attachment in users* (p. 21).

Norman (2002) underlines mutual nature of objects design, which is also applicable to software:

> *If everyday design were ruled by aesthetics, life might be more pleasing to the eye but less comfortable; if ruled by usability, it might be more comfortable but uglier. […] Trouble occurs when one dominates all the others* (p. 153).

To sum up, it is quite important that both sides of the designing process (form giving and planning) could overlap. As a result, a good cooperation between programmers and designers is needed. The question is: how this cooperation is managed in Scrum framework?

According to Arslan (2012), the challenge lies within combining engineering and designing user interface. Designers are under the pressure whether to design an up-front design one sprint in advance or wait for functionality to start building a

visual solution for the interface. This challenge has been mentioned several times within Web discussions about Agile methodologies (Manning, 2008). Developers argue that one of the biggest challenges moving away from an up-front design approach to an Agile (Scrum) approach is figuring out the best way to incorporate the work of visual designers into the collaboration. And finally, Ambler (2010) claims that the Agile approach to design is very different than the traditional approach and apparently more effective too. However, there is no general rule how to make designers work with Agile. Several solutions were introduced by Cohn (2010) and Ambler (2010). The most common advices are that design should be intentional but yet emergent and designers should iterate their work. Nevertheless an extreme changing of IT industry requires new methods and advices. Unfortunately even participants of Agile events pay little attention to this problem. The recent Agile Saturday hold in Riga, Latvia (agilerigaday.lv) included 17 presentations covering various topics except for design. The impact of such an absence of a set of guidelines can be quite noticeable. There are small and inexperienced companies enthusiastic about introducing Agile to their teams. At the same time they are not able to do it successfully because of unexpected problems with designers who join the ongoing development. The research paper is focused on one particular Company Sigma that meets the problem described above.

Chapter 3

Company Sigma Case Study

This chapter accurately describes the environment of Company Sigma, where our design object is implemented. We evaluate what can and what cannot be changed. The variables that *cannot* be modified, but should be observed, are: company background, company's objective, product roadmap, and personnel volatility. We are not allowed to interpose in them, even if we could, since they are predefined by company managing department. What we *can* modify during implementation period is infrastructure and team engagement. However, all the factors discussed in this chapter are interconnected and affect our research in a certain way.

3.1 Company Background

Company has been successfully developing software since 1990s. It produced and maintained a large Windows based Sales Force Automation tool, which ended up with an old-fashioned user interface. In addition, Company initiated a mobile phone version of the same tool, which became outdated due to extreme progress in technology production. As a consequence, Company decided to transform Windows software into seven simple and handy iOS applications – up-to-date products with high speed, user-friendliness and nice-looking interface. These characteristics are beneficial for customers but rather challenging for developing

team. Producing nice-looking but well-working iOS application requires collaboration between designer and programmers. The company has never had a designer for visualizing interface before. This condition has to be taken into account while implementing Scrum, because designer should work incrementally with other team members. It could be really challenging for a company that is inexperienced in both fields: user-interface design and Scrum.

3.2 Company's Objective

Company aims to produce seven user-friendly iOS applications with modern design. With the new application company offers the enhancement of business process and client meeting experience, speed and easiness, lifting customers' reputation, minimizing transition costs from the old software.

Concerning technical part of the process, company aims to provide all the distinctive features of high quality iOS applications. It is important for the company to follow Human Interface Guidelines (HIG, 2011) provided by Apple: *"aesthetic integrity, consistency, direct manipulation, feedback, metaphors, user control"*.

In order to accomplish this quite an extensive plan, company needs to attract investors at an early stage of development, because they already had a negative experience with producing mobile application that is no longer in demand. In this case, Scrum seemed as a good solution to maintain the process since it allowed presenting a potentially shippable product every certain period. In addition, company's managing director (Product Owner, according to Scrum terminology) found Scrum framework well-documented and easy to understand for everybody. The company has not had any formal methodology before, but has been working *"relatively close to what is Scrum"* (Annex A.1). Product Owner was attracted by Scrum's way of defining roles and tasks. As a result, Scrum was applied and supported by Product Owner:

> *I don't think there is a huge enthusiasm but I don't think there is a big resistance either, because the changes are not very big. It's just that we share the information and we are a little bit formal about it* (Interview with Product Owner, p. 6).

However, Scrum was not working to its full extent and, therefore, several modifications were needed. These modifications will be discussed further.

3.3 Product Roadmap

Company's current project is focused on the first application from a set. It was launched in November 2011 and should be released in fall of 2012, which makes the development cycle exactly one year long. Forty percents of the application is ready by April 2012, therefore the first project is running in time. The rest projects will be developed consequently, either by current team or involving more distributed teams from abroad. Figure 3.1 demonstrates the roadmap of releasing all seven applications. There is a gap in the first quarter of 2013, since this time is predefined for developing the corresponding back office. This is quite a long development period, which means that Scrum should be implemented in a way that it would still work far in the future. There also exists an accelerated version of roadmap in case if Company Sigma finds more investors and hires an additional distributed team in order to speed up the development process.

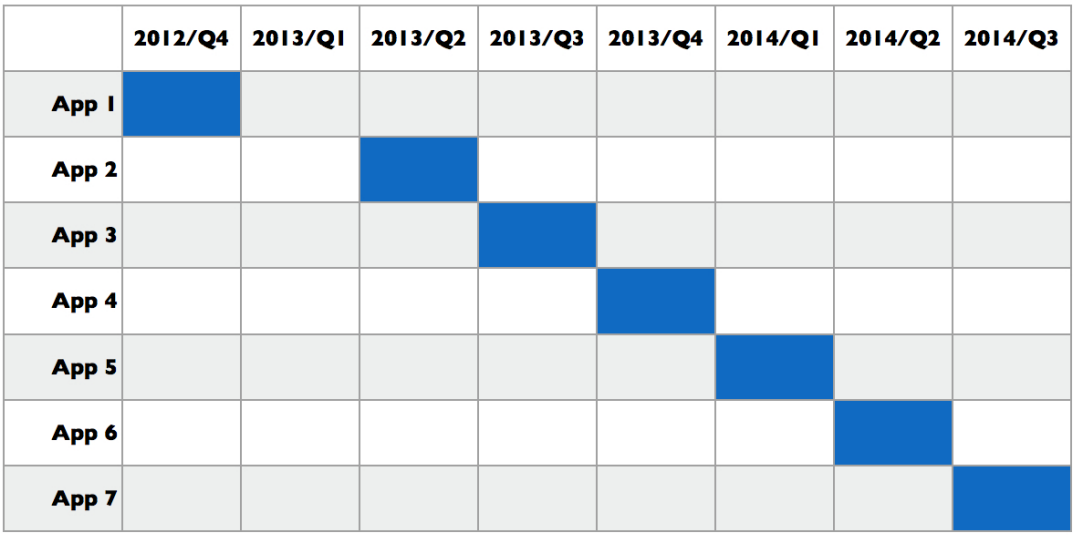


Figure 3.1 – Release chart for seven iOS applications.

3.4 Personnel Volatility

Unfortunately, personnel volatility cannot be predicted. Managing director may decide to break contracts with some employees or hire the new ones. Those changes will certainly affect implementing the design, because new members should be taken into consideration, while the old ones might behave differently.

And that is what happened during this research. One of the team members quitted his job at Company Sigma during the middle phase of the research. However, that person was highly important because of being active and interested in our design object – Scrum. Therefore, after leaving the position, he was still interviewed several times and was treated as part of the team.

3.5 Infrastructure

Infrastructure means *the basic organizational facilities needed for the operation of an enterprise* (The New Oxford American Dictionary, 2010). Scrum itself is a part of infrastructure. It introduces new artifacts and new roles, and reshapes the old ones.

Team works in the office where each member has his own room, MacBook, and iPad for professional purposes. There is also a big meeting room with a rounded table, whiteboard, and bookshelves. Sometimes developers practice Pair Programming there. But mostly this room is used for weekly meetings where people share ideas, discuss further plans, and analyze current problems. Remotely working members can communicate with each other face-to-face during these meetings. Whiteboard is used time to time for noting important issues. Figure 3.2 illustrates the working environment in Company Sigma.

Figure 3.2 – Working environment of Company Sigma

Company uses several online tools for spreading information. Developers use GitHub for sharing the code. TestFlight is used for releasing an improved version of iOS app and sharing it with other team members. Dropbox is used for sharing design elements, such as icons, buttons, backgrounds, etc. In general, there is enough tools for Scrum framework.

3.6 Team Engagement

There are eight people involved in the current project. According to Scrum, all participants, except for Product Owner and Scrum Master should be called 'developers' and treated the same (Cohn, 2010). Therefore, we have six developers in Company Sigma, which might seem quite an ideal '*two-pizzas-team*'. However, in scope of the project, their responsibilities and their input are different. In frames of this research we specify the roles of the participants in this way[3]:

1. Managing Director / Product Owner
2. Chief Developer
3. Senior Developer
4. Junior Developer
5. Software Architect
6. Designer
7. Usability Tester / Scrum Master
8. Customer Support Specialist

The connections between team members, at the moment when Scrum started to be implemented, can be seen from Figure 3.3. Even though, the roles were distributed properly, situation still did not answer all standards of Scrum, since there were few interconnections between team members and low self-organization.

Employees reported directly to Product Owner, although Product Owner should have treated team as a whole system, where nobody is responsible for concrete achievements or mistakes. As suggested by Cohn (2010), *"there is no 'my work' and 'your work' on a Scrum team; there is only 'our work'"* (p. 201). However, Company Sigma worked in a different way when they started adapting Scrum. Figure 3.3 indicates cooperation between Junior, Senior and Chief Developers. Chief

[3] Henceforth we write names of the team member with capital letters, meaning concrete people but not revealing their names.

Developer works in different geographic location, so the conversations were usually held via Skype. Their cooperation was entirely caused by technical issues, with no relation to Scrum activities. Generally, there was low connection between members working outside the office.

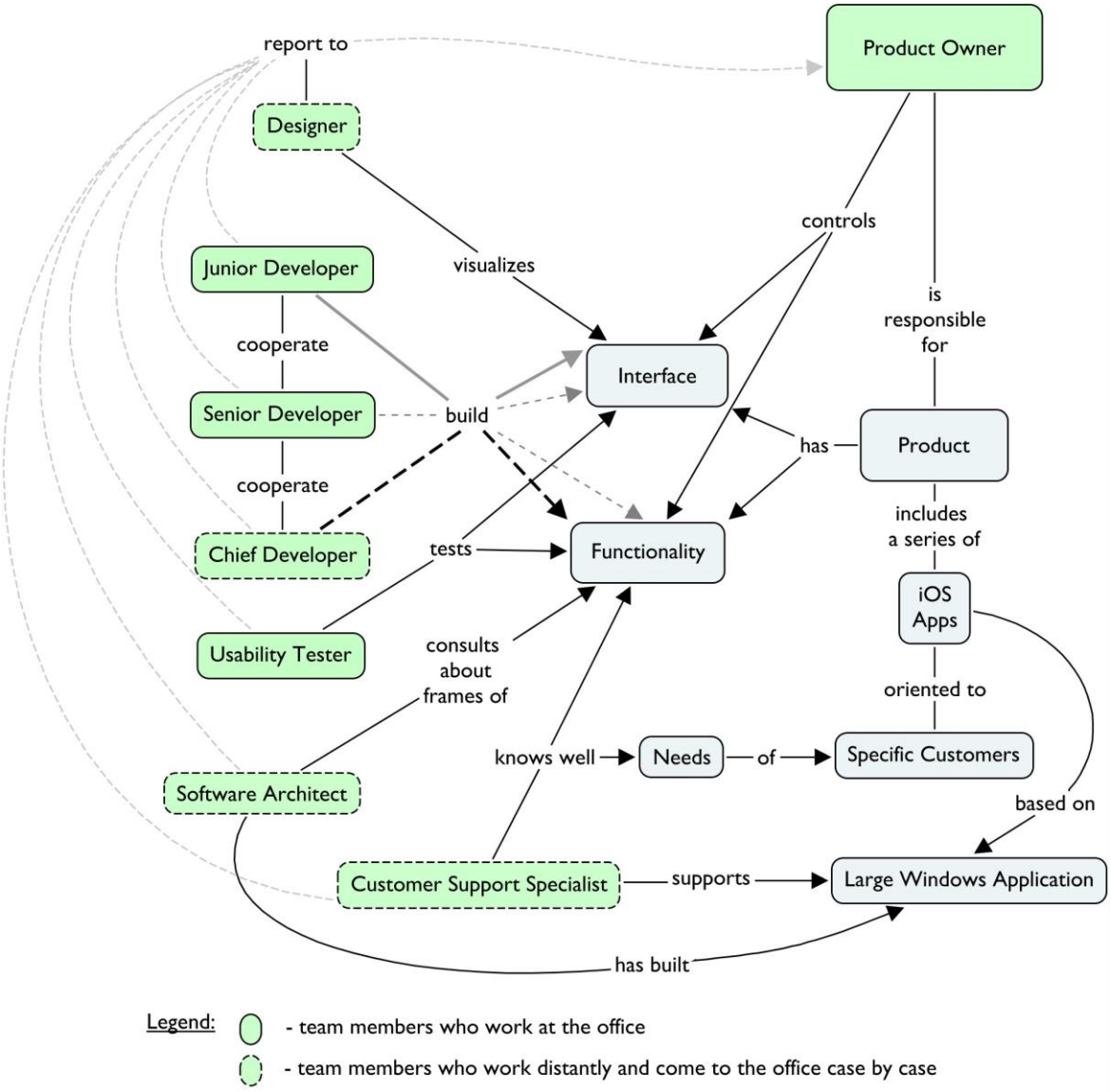


Figure 3.3 – Internal connections and responsibilities of team members in a Company Sigma at the moment of introducing Scrum.

To compare this situation with an 'ideal' one, that is highly recommended by Scrum, we designed another concept map as in Figure 3.4. This shows that team is equally responsible for the product, and that each member does not report to Product Owner separately but does it together on the meetings through Scrum Master who further provides results to Product Owner.

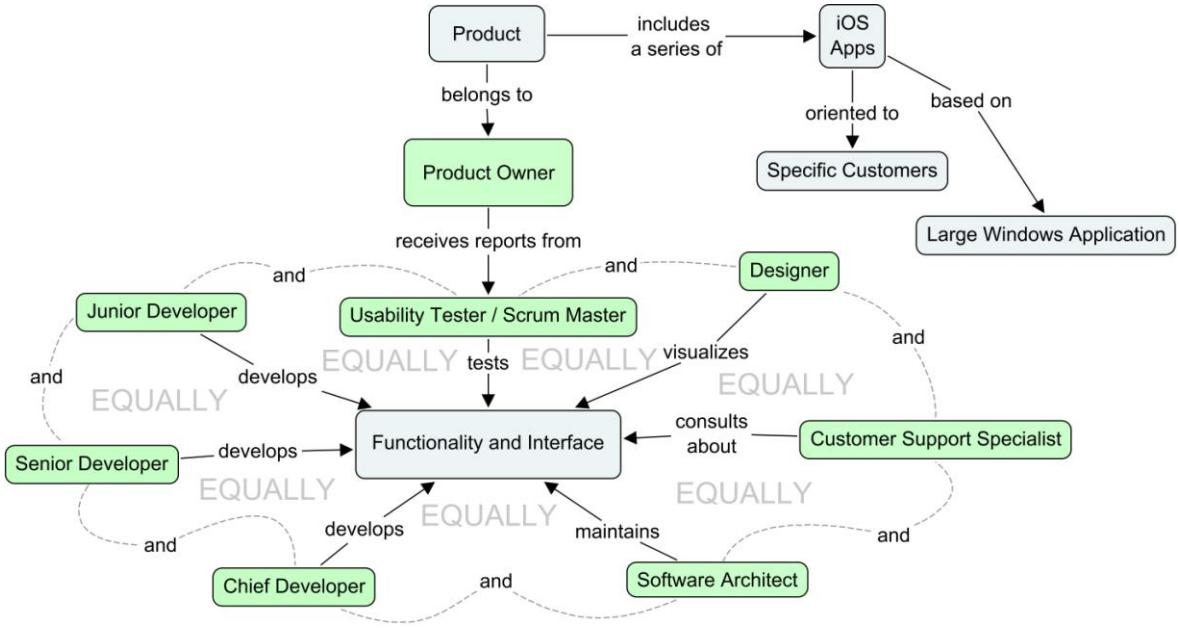


Figure 3.4 – Ideal model of Scrum for Company Sigma

CmapTools application allows analyzing the number of 'links in' and 'links out'. In both concept maps, the greatest amount of 'links in' are towards Functionality and Interface, which is right, because that is what team develops. However, the most 'links out' in the real model are coming from Senior Developer. Senior Developer is a full-time employee who works in the office every day and has the biggest number of tasks. In the ideal model, 'links out' are equally distributed, because nobody is overloaded.

3.7 Summing up

To sum it up, company's objective can affect our design in a very positive way. Company aims to produce high-quality interactive iOS Apps, which involves user-centered and goal-directed design approaches proposed by Cooper (2007). As we know, Scrum recommends the same techniques, so both tactics really match each other. However, some factors ensuing from the company background can affect our design negatively. Company does not have experience in designing iOS

interface and using formal methodologies. It is quite hard to maintain two novel fields at a time without sufficient knowledge of both. Product roadmap is quite acceptable within Scrum framework. However, there is a risk of losing enthusiasm of delivering something potentially shippable every sprint. It is important to notice that Scrum requires not to "do a great deal of additional work at the end of each sprint" (Cohn, 2007, p. 265) but to find breakpoints and split work intro parts. And finally, personnel volatility is obviously the most unpredictable factor. Infrastructure is suitable for implementing Scrum. However, the meeting room might be used more frequently, and be filled with Scrum elements, such as Task Board, Sprint Burndown Chart, meetings schedule, etc. Whiteboard can serve as a tool for distributing ideas. Team members can spend more time together and share the work rather than sit in their rooms doing only their part of the job. Their engagement can be also turned towards positive direction; more internal connections can be made.

Chapter 4

Detecting Initial Problems

This chapter is fundamental for our research because it reveals problematic factors of Company Sigma. These will be further improved by implementing Scrum. The main problems are related to human resources. There is low interaction between certain members and lack of mutual understanding. To reveal such problems, an ethnographic approach was applied, which included observations, interviewing, card sorting, and personal constructs.

4.1.1 Team Members Identities

The very first stage of our research started with general observation of company members. During this stage, data was collected as a set of notes and short conversations. Putting together comments and observations, the identities of 8 team members were described as presented below.

Product Owner or managing director represents the customers, meets with them and knows their needs. He is responsible for program logic. Product Owner was the one who decided to implement Scrum methodology and asked Usability Tester to provide tools for doing Scrum, bought the books and found educational material in the Internet. He has a huge poster in his office with the list of customers and important information about them. Product Owner has many ideas

and sometimes just comes to Junior Developer or Usability Tester and tells about the new feature which should be implemented. During the weekly meetings on Mondays he chaotically writes notes on the whiteboard. Also, he comments the design and always wants everything to be improved, seeking for the stage when „nothing can be improved". He is almost every day in the office except for business trips which happen ca 3 times per month.

Chief Developer programs business objects and deals with synchronization. He is interested in producing a good quality code, likes to do things which are not clear and need to be solved. He develops software which was never developed before. He is also the author of the very first product, has many practical ideas and solutions. Chief Developer is one of the oldest employees. He works more like a consultant, comes to an office every second Monday and stays until Tuesday, if his assistance is needed. Sometimes he chats with Junior Developer and Senior Developer via Skype. He belongs to the company, but his main workplace is not in the same office. He rarely uses shared online tools, does not use paper tools (such as Task Board). He is not interested in design at all.

Senior Developer does the maintaining part, is responsible for synchronization. When synchronization is done, he can move to account lists and embedded content in HTML5 (for a web-based version of a product), together with Chief Developer. The takes part in weekly meetings, makes suggestions considering design and usability. But most of the time he sits in his room completing a large number of tasks. One of the oldest employees; works full-time every day, sometimes stays longer; posts online; also likes discussing things orally.

Junior Developer is developing user interface via Xcode SDK using storyboarding. About 60% of his duties are related to design. Using the Interface Builder component of Xcode, Junior Developer drags and drops view controllers onto a canvas and designs user interface of each view. Junior Developer's work is

about trying new methods and experimenting (no preferences are proposed by Product Owner). In addition, he is responsible for functional part. Formally, he is a part-time employee but actually works every day plus on the weekends and at night. Junior Developer is very excited about the product but disappointed about the slow tempo of other team members. Likes Scrum and is upset that the team does not follow it; puts his tasks on the Task Board, actively posts on Kanbanery.

Software architect is also a system administrator who works in collaboration with Chief Developer. He discusses programming, holds a server, and creates company e-mail accounts for new employees. Everyone treats him as a specialist. He no longer works in the company but still has his own room and comes when called. Participates in weekly meetings, uses online tools but posts nothing there. He was presented an iPad before leaving the company, so that he can still come and be a part of the developing team.

Designer creates visual part of user interface. Since there was no layout, he designed it himself relying on his own competence. Later on, this design was taken as a standard for further developing. According to his design, application includes elements, which are hard to develop and require more effort from programmers' point of view. In spite of all this, split view is not essential for this type of App. Designer works distantly, comes once a week for a Scrum meeting and always disagrees with changes. It is always hard to reach him. In the beginning he shared only .jpg and .png files of an App screens design, hence all the buttons, labels, frames, and other elements were parts of entire image. As a consequence, Usability Tester had to do the "dirty job" and cut out all the elements.

Usability Tester's initial duties were black-box testing of an interface and (later) assisting the Designer. Since there was not much to test in the beginning (the first working piece of iOS application was ready only in 2 months), Usability Tester

was involved into management process and became a Scrum Master. He created Scrum instruments, including Task Board, Fibonacci Numbers for tasks estimation, Sprint Burndown Chart, Agile Manifesto printed version etc; visualized database tables (on the wall); proposed Paper prototypes; made Keynote prototypes as suggested by Product Owner. He does testing and documenting, posts results to Kanbanery. Usability Tester works as a mediator between Junior Developer and Designer, providing Junior Developer with missing elements, which Designer forgets to send (or create). He works every day, part-time.

Customer Support Specialist solves customers' problems, talks to customers, answers their questions etc. He is currently responsible for smooth working of an existing program (large Windows-based application). He works distantly, sometimes comes to office and works every day during the whole week. Does not take part in weekly meetings but is required when customers' needs are discussed. He never collaborates with Developers or Designers, works closer to Product Owner and financial department.

It might be quite a challenge to engage some of the team members into Scrum. To predict the resistance we can use classification, introduced by Discovery Learning, Inc. of Greensboro, North Carolina (2003), which assumes that there are three types of individual disposition to change: conservers, pragmatists, and originators. The majority always belongs to pragmatics group (50%), others are either conservers (25%) or originators (25%). We grouped Company Sigma team members using these categories (see Table 4.1) and added descriptions provided by Luecke (2003), who also analyzed individuals' resistance.

It is obvious that individuals' distribution within the company corresponds with general distribution. Team members were split into categories according to interviewees' comments and researcher's observation.

Conservers	Pragmatics	Originators
Designer Customer Support Specialist	Product Owner Senior Developer Chief Developer Software Architect	Junior Developer Usability Tester
✓ Prefer change that maintains current structure ✓ Enjoy predictability ✓ Honor tradition and established practice	✓ Prefer change that emphasizes workable outcomes ✓ Are more focused on results than structure ✓ Are open to both sides of an argument	✓ Prefer change that challenges current structure ✓ Will likely challenge accepted assumptions ✓ Enjoy risk and uncertainty

Table 4.1 – Distribution of team members within three types of individual disposition to change

4.1.2 Weekly Meeting Episode

The team had a weekly meeting every Monday. There were several purposes for such appointment. First, it was important to gather part-time, full-time employees, freelancers and distributed members in one place at one time and discusses the objective. This meeting was planned as a Retrospective in Scrum for discussing what is going wrong and what should be changed. In fact, meetings nave never had a common structure, and usually people were speaking chaotically, interrupting each other, moving from one topic to another inconsistently. Figure 4.1 presents a short overview of one meeting episode, documented by the researcher. In addition to such conversations, sometimes meetings cannot even be continued since no result was achieved during the week, and the application was not even launching. These circumstances certainly needed involvement of methodological approach, Scrum in our case.

Weekly Meeting Episode

Product Owner, Designer, Usability Tester, Junior and Senior Developer are sitting at the round table and discussing additional feature that should be implemented. Customer Support Specialist is sitting nearby doing his business not participating in the meeting.

Designer asks questions with a negative attitude. Such scene repeats several times, after discussing every new feature Designer tries to change Product Owner's view.

Finally, Product Owner stands up and explains as simply as possible how the feature should work. He shows the "door model", where door means button, and entering the door means activating this button. This looks funny but quite illustrative. Designer still disagrees but remains silent.

One hour has passed. Too much talking and no mockup created.

Suddenly Senior Developer stands up and starts drawing screen layouts on the whiteboard. Designer and Product Owner are talking between each other not paying attention to Senior Developer. Colleagues are speaking all at once. In a moment Senior Developer kindly asks to look at the whiteboard. Team members get interested and start commenting the layout, adding elements and passing marker to each other. Finally, several ideas get created. People seem more or less satisfied.

Figure 4.1 – One weekly meeting episode: conversation between Designer (green), Senior Developer (blue) and Product Owner (red)

4.1.3 Revealed Problems

Initial observation of people, their roles and attitude, showed that company meets challenges due to diverse interactional identities (Brown, Lindgaard, and Biddle,

2012) of team members, especially the Designer (conserver) being different from the majority (pragmatics) and from the opposite minority (originators). It is also crucial that the team splits into newcomers (recently hired employees) and seniors (more than ten years working employees). People need some time to get involved and feel as a team. Plus, different time schedule does not let members spend more time together, for instance, participate all at once in daily Standups.

We can notice that individuals have different goals, artifacts, objectives and tensions. For example, Junior Developer wants to work fast and methodologically, whereas Designer does not provide graphic elements for the interface consistently. Some members are more collaborative than the others. Also their level of adopting changes is different. When implementing the object of our design research, we should make it suitable for all members, according to their personal identities. They *all* should feel like a team having one goal and doing one thing together.

Problem illustrated by the meeting episode demonstrates lack of mutual understanding and low self-organization. Things, which should be discussed during the meeting, are not listed, time is wasted and no result is achieved. However, there is a positive tendency that some members try to improve the situation and organize the others. There was an attempt to prototype the function ("door model" in Figure 4.1), which indicates creative thinking. Another member tried to draw everyone's attention by making notes on whiteboard. By designing effective Scrum approach we should help active people to put their energy and ideas into the right channel.

4.1.3 Turning Problems into Goals

The revealed problems can be easily transformed into goals of our design. Table 4.2 demonstrates how this can be done, based on the qualitative data received from the interviews and observations. After analyzing Company Sigma

environment we stick to our second hypothesis: Design and Programming are difficult to combine *not* because of Scrum, as inappropriate framework, *but* because of the team being not enough self-organized. Therefore, an effective implementation of Scrum should change the situation.

The table of problems is based on Chapters 3-4, and qualitative analysis of thematically coded interviews. The list of codes can be found in Annex A.

PROBLEMS	WEAK POINTS	IMPROVEMENTS	GOALS
Team is not self-organized	Distributed team	Create possibilities for online collaboration	**Better self-organization**
	Unawareness of all possibilities of Scrum	Make presentations, motivate team, convince team that they have potential to do Scrum	
	No common understanding of product's functionality	Make prototypes	
	Wasting time during the meetings	Plan what should be discussed in advance	
Designer does not support Scrum	Designer prefers traditional Project Management approach	Adopt some methods of Scrum specially for Designer	**More effort to maintain design and Scrum**
	Designer's contributes are not regular	Contact Designer regularly. Find a mediator between Designer and the rest of the team	
	Designer needs documented information	Document iteratively, provide Designer brief overview of features needed soon	

Table 4.2 – Turning problems into goals

Chapter 5

Designing Effective Scrum Approach

This chapter finally covers the consequent process of tailoring our design object (implementation of Scrum) into specific working environment of a Company Sigma. In the beginning, we give an overview of existing approaches relevant to this research and adopted during the implementation cycles. Latter part allows tracking each cycle one by one.

5.1 Guidelines for Design Research

Various methods of designing effective Scrum approach were inspirited by works of Koskinen et al. (2011), Brown et al. (2012), Nelson, Ketelhut, Clarke, Bowman, and Dede (2004), Collins, Joseph and Bielaczyc (2007), and IDEO cards (2003).

The book by Koskinen et al. (2011) presents a great amount of constructive design research examples. The approach of Eureka project (p. 20) seemed to be very relevant to our methodological framework, because that project required intervention to a firm in order to improve its informational system and enable sharing their practical knowledge. The idea is that the most relevant information flows through individuals and should not be imposed by artificial methods. This is important for our research, since implementing new practices, such as Scrum, is similar to building informational systems and sharing knowledge.

Another relevant notion from the book by Koskinen et al. (2011) is that solving environmental and social problems cannot always be completely successful, and this is not a solid foundation for design research. The goal is to imagine "*something better than what exists*" (p. 17). Sticking to this goal would lead to more satisfying results than struggling for ultimate solution. And there is always "*a rich array of theory*", which *"gives constructive design research plenty of depth"* (p. 118) to keep moving towards better results.

Brown et al. (2012) provide sufficient result of studying enacted interactional identities of designers and developers. Their paper focuses on understanding collaborative work, and provides four categories of interactional identities: goals, shared objective, shared artifacts, and tensions. The material is highly relevant to our research because information presented there is very recent and covers exactly the field we are interested in: designing and programming.

More concrete technical approaches of this thesis were adopted from Nelson et al. (2004), Collins et al. (2007) an IDEO cards. Nelson et al. were designing The River City virtual world to promote learning for all students. They split implementation period into four cycles which, in their turn, were split into stages: 'implementation', 'findings' and 'implication'. We found this logic suitable for our research as well, with slightly changing the name of the last category into 'lessons learned'.

Collins et al. (2007) present well-documented strategy for developing design research. The major issue is the similarity between the complex environment of Company Sigma and learning environments, described by Collins et al.: *"there are many variables that cannot be controlled"* (p. 19). Therefore the goal is "*to optimize as much of the design as possible and to observe carefully how the different elements are working out*" (p 19.).

IDEO method cards (2003) describe design methods, each on one page. Cards have two sides, one illustrates the method with relevant picture, and another briefly describes it. These techniques are intended as inspiration for practicing designers. We found there several methods suitable for our design research, such as Quick-and-Dirty Prototypes (p. 43), Shadowing (p. 55), Social Network Mapping (p. 57), Still-Photo Survey (p. 59), Try It Yourself (p. 65), Activity Analysis (p. 73) etc.

5.2 First Cycle: Meeting Room Enhancement

5.2.1 Implementation

The first implementation of Scrum framework was held in Company Sigma in December 2011. We focused on 7 team members, who worked in the office full-time, part-time or whenever required. We concentrated our evaluation on members' reactions to changes and their attempts to use any of introduced artifacts. Modifications were applied only to the meeting room, which was filled with various Scrum elements, described in Chapter 1. Prior to adapting any instruments, a large poster was put on the wall. Team members stuck there screen designs of application, database tables and a blank sheet for suggestions.

Scrum elements were the following:

1. Task Board
2. Two User Stories
3. Fibonacci Numbers for tasks evaluation
4. Sprint Burndown Chart

In addition, people were informed that they have a Scrum Master in their team. The meeting room looked as shown in Figure 5.1.

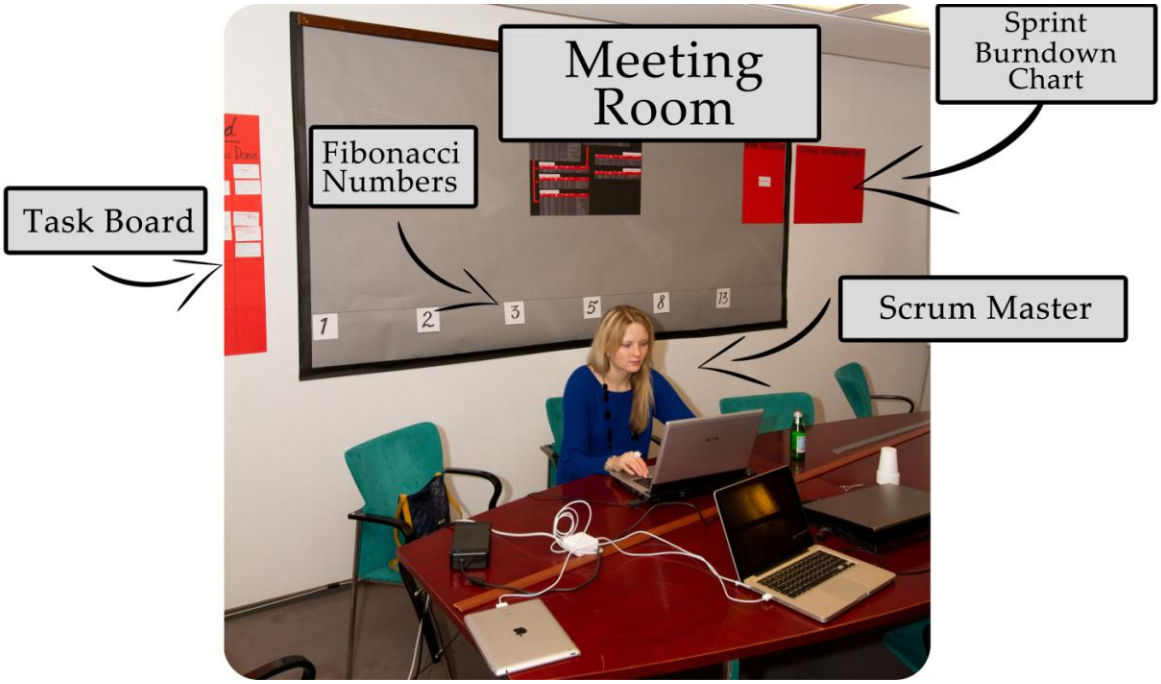


Figure 5.1 – Meeting room filled with Scrum elements

5.2.2 Findings

From the observation of focus group participants, we noticed that the most successful artifact was the Task Board. Two members (Junior and Senior Developer) used Task Board every day creating new tasks and moving them from one column to another. Evolution of the Task Board is shown in Figure 5.2.

Both User Stories were written by Product Owner and were 3 to 5 lines long, which is more than required by Scrum. Team members found them quite complex, therefore we can notice many tasks cards stuck to the same story (see Figure 5.2).

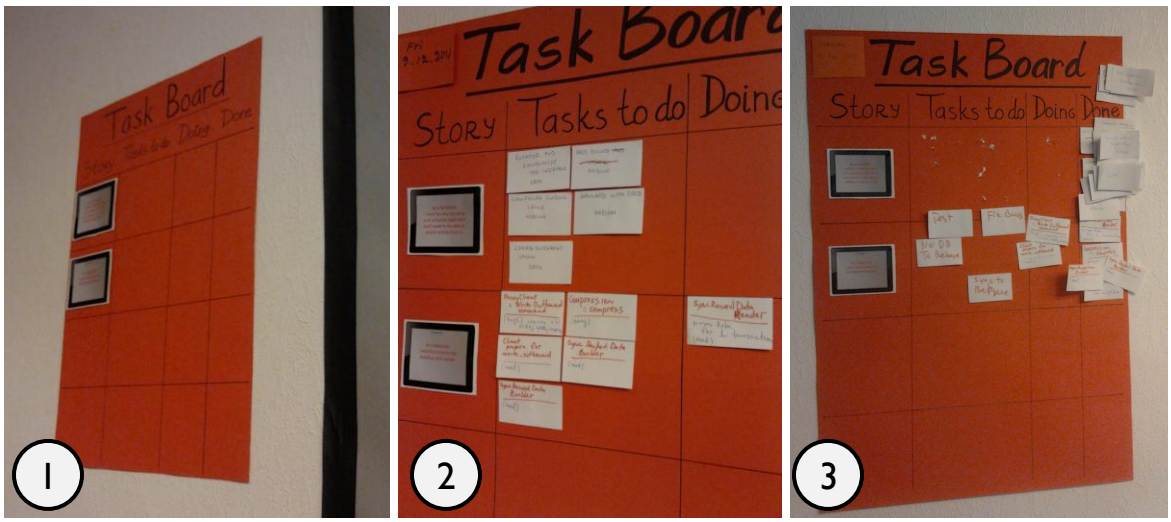


Figure 5.2 – Evolution of the Task Board

The last two elements (Sprint Burndown Chart and Fibonacci Numbers) were not used at all. According to Product Owner, it was too early to use Sprint Burndown Chart since we haven't completed the User Stories, therefore we cannot move further. Fibonacci numbers were also postponed until team finishes solving technical problems, which had the main priority at the moment.

Senior and Junior Developers used Task Board most of all because they worked in the office every day. However, Chief Developer, who comes every second Monday, never put his tasks cards onto Task Board. Software Architect used it a few times.

5.2.3 Lessons Learned

Based on this implementation, we decided that our changes had been positive and should be kept. They moved us further to achieving the main goal: better self-organization of the team. However, another goal has not been affected at all: how to improve collaboration with Designer. Several sub-problems also remained untouched, including distributed team and unawareness of all possibilities of Scrum. We took them as the next steps towards designing effective Scrum approach.

Before moving further, we emphasized additional modifications:

1. Clear and simple User Stories
2. Educative presentation about Scrum
3. More meetings
4. Paper prototypes of application

5.3 Second Cycle: Facing Challenges

5.3.1 Implementation

This one month-long cycle (January 2012) was the least successful due to several reasons. From the beginning, our plan to implement clear and simple User Stories was rejected because the first User Story, which was in its developing phase, included solving a big task, therefore Developers and Product Owner put it as a priority and nobody had interest or time for creating new User Stories. However, according to Scrum, the stories should be written by several members and Product Owner, not only by Scrum Master. The most we could do was splitting existing stories into parts and testing them as suggested by Sims and Someone (2011). The list of printed test cases was passed on to every member during the weekly meeting. However, it did not draw much attention and did not receive any comments. The list is illustrated in Figure 5.3.

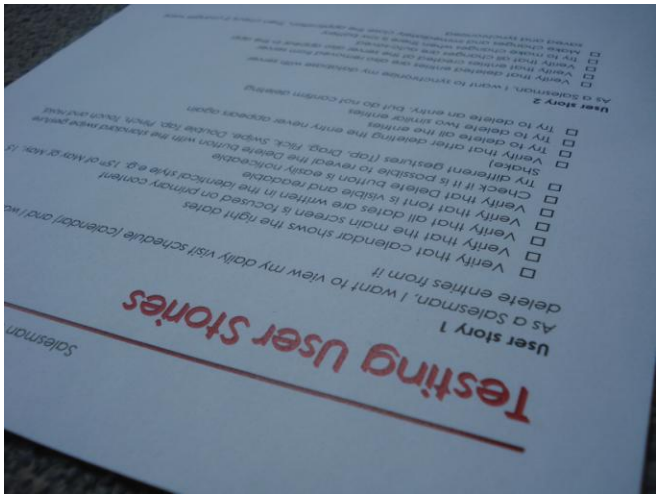


Figure 5.3 – Report on testing User Stories

The second modification was an educative presentation for team members covering the main features of Scrum. Six people were participating: Product Owner, Junior, Senior, Chief Developers, representative from financial department and Scrum Master as a speaker. Several posters and printouts were prepared in advance. Meeting room was enhanced once again: Scrum methodologies were illustrated on the opposite wall (see Figure 5.4).

Third modification was organizing more meetings, including daily Standups, when everyone should report about the work done yesterday and the work planned for today. A timetable was put on the wall so, that Scrum Master could update it when needed (see Figure 5.5).

The last modification was inspired by Cooper et al. (2007). These were paper prototypes meant to clarify features of the application and achieve common understanding of product's functionality. Two versions of prototypes were initially created. The first one was hand-drawn, another constructed from iPad stencils downloaded from the internet. Prototypes were located on the table in the meeting room.

Figure 5.4 – Scrum values presented on the wall of Company Sigma

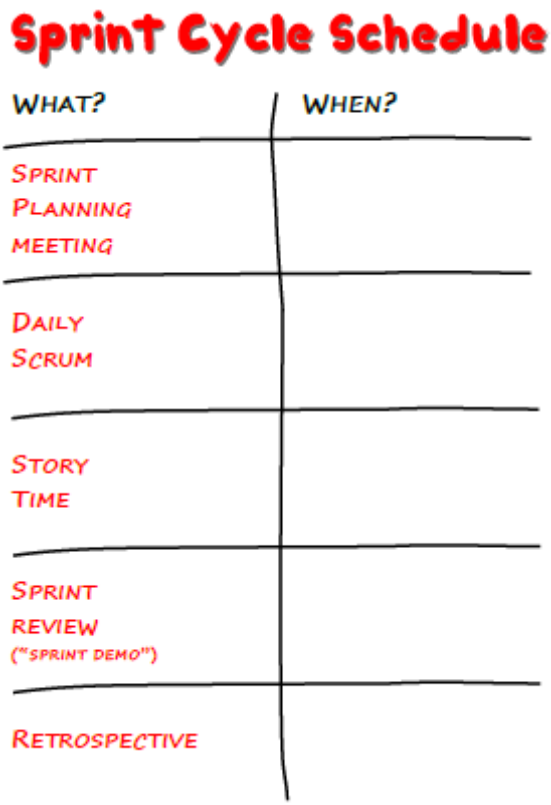


Figure 5.5 – Sprint Cycle Schedule that allows constant updating

5.3.2 Findings

After observing people's reactions to implemented changes, we realized that nothing actually worked. People became slightly more aware of Scrum possibilities, but they did not meet interventions with great enthusiasm. The positive movement was after the presentation, when an important question was asked: "How does design match Scrum?"

Team members were not interested in rebuilding User Stories due to occupation with technical problems. Junior Developer provided valuable feedback regarding the second cycle of tailoring Scrum saying that team members had more than enough instruments and never managed to adopt User Stories or Sprint Burndown Chart. They did not manage to break tasks into small tasks; had difficulties with finishing sprint that lasted more than a month. According to Junior Developer, the first task was defined very abstract and represented the main idea of application (see the transcription of interview on CD enclosed).

New meetings were not successfully implemented either, because the Developers were occupied with other tasks. Senior Developer described a situation when they had a highest priority task, and Product Owner told that if the task would not have been solved by the end of the week, they would stop all the other work. For the same reasons paper prototypes were not even tried (see Figure 5.6).

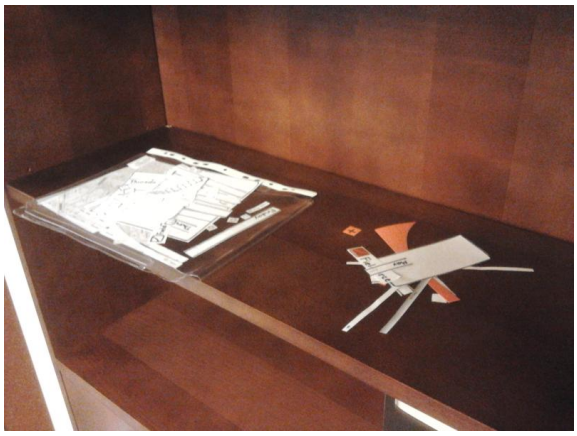


Figure 5.6 – Unused paper prototyped moved from table to shelf

5.3.3 Lessons Learned

The main lesson learned from the second cycle is the importance of factors that we cannot control: technical and time problems, decisions of managing director and team preferences. This indicates that Scrum should not be introduced artificially but needs to be adopted according to current situation. In addition, Scrum has to be flexible and well-understood by team members. As it was already mentioned by Koskinen et al. (2011), knowledge is spread from individual to individual, not through the theoretical presentations.

Introducing more meetings did not improve the situation of distributed employees being less active. Team members were not coming more often, so we had to check if Scrum methodology has methods to maintain distributed team. It is important that Designer could not even attend the presentation. Designer's contacts with the rest of the team were very inconsistent and rare.

Based on this implementation, we realized that previous modifications should be revised and additional improvements introduced:

1. Create possibilities for online collaboration
2. Involve people into using prototypes
3. Adopt some methods of Scrum specially for Designer
4. Contact Designer regularly. Find a mediator between Designer and the rest of the team

5.4 Third Cycle: Moving to Kanban

5.4.1. Implementation

Taking into account negative experience from the previous cycle, it was decided to collect additional data in order to understand how team members see each other

and how they establish priorities. We already knew that some tasks were estimated as more important than the others. Therefore finishing the sprint in time became a real challenge.

The feedback received from Product Owner demonstrates that the main challenge was fixing sprints and having a fixed set of features for the sprint. Moreover, it was difficult to estimate the features that could be done in a predefined time. As a result, people were under the pressure and tried to finish sprint sacrificing the quality.

For the research, it meant choosing between two ways: (1) either to keep sprints, saying that the main idea is exactly to learn how to split tasks into predefined time, or (2) to accept the situation how it was, and combine the rest of Scrum with other techniques that allowed more flexible iterations. Finally the last way was chosen, and Company Sigma started to use Kanbanery digital Task Board (Figure 5.7).

We already gave an overview of team members' identities, but it was not enough to reveal deeper connections between members. During the third implementation cycle, the obvious problems led us to consider Repertory Grid Technique (RGT) as a tool which might help to understand an individual's personal construction of surrounding environment. Hassenzahl and Wessler (2008) proved that RGT is quite usable from a design perspective; it helps to find hidden connections.

Five respondents were individually presented a randomly drawn triad of cards with team members' names on them. People had to separate two cards from one, explaining what differentiates them. Usually, this is expressed in personal perception of objects (kind-angry, active-passive, cold-warm). As a result, we got 24 bipolar categories, which were grouped by similarity. However, respondents preferred business connections rather than personal. All of them spoke about

working environment and not about individual attitude. As a result, we could not treat the results as 'personal constructs' and, therefore, grouped them in a different way (see Table 5.1).

Work-related constructs	Characteristic constructs	Obvious constructs
Makes design proposals - Receives feedback about the design Work on interface - Work on logic Front-end - Back-end Receive big tasks - Receives small tasks Vision - Implementation Work with graphics - Works with code Work with clients - Works with computers Find problems - Find solutions Ongoing continuous connections - Temporary connections Keep server - Keep database in the office - Outside the office Connected to other members – Individuals Maintain old system – Share new ideas Keep server - Keep database	Experienced - Assistant Agile - Traditional Young - Old Communicative - Unsociable	Bigger salary - Smaller salary Men - Women Full-time -Part-time Contract – Freelance Smokers - Non-smokers Employer - Employee

Table 5.1 – Constructs collected by RGT

Nevertheless, the positive result of this small research shows the high interest of individuals in work-related connections. Personal level does not interrupt ongoing work process; everyone treats each other respectfully, as colleagues, not as 'sworn brothers'. This makes design implementation easier, because people are aware of each other's duties and can easily identify who is responsible for what. Such situation demonstrates how big their potential is to become more self-organized.

Next thing to establish was separate connections with the Designer. Skype was used as a contacting method; a new Kanbanery project was open for design issues. Scrum Master became a mediator between Developers who worked in the office and the freelance Designer.

The idea of prototypes was still considered as a useful one, but this time prototypes were implemented in two different ways: whiteboard schemes and accurate hand-drawn paper layouts (see Figure 5.7).

The whole implementation period of the third cycle took one month, from the end of January until the end of February.

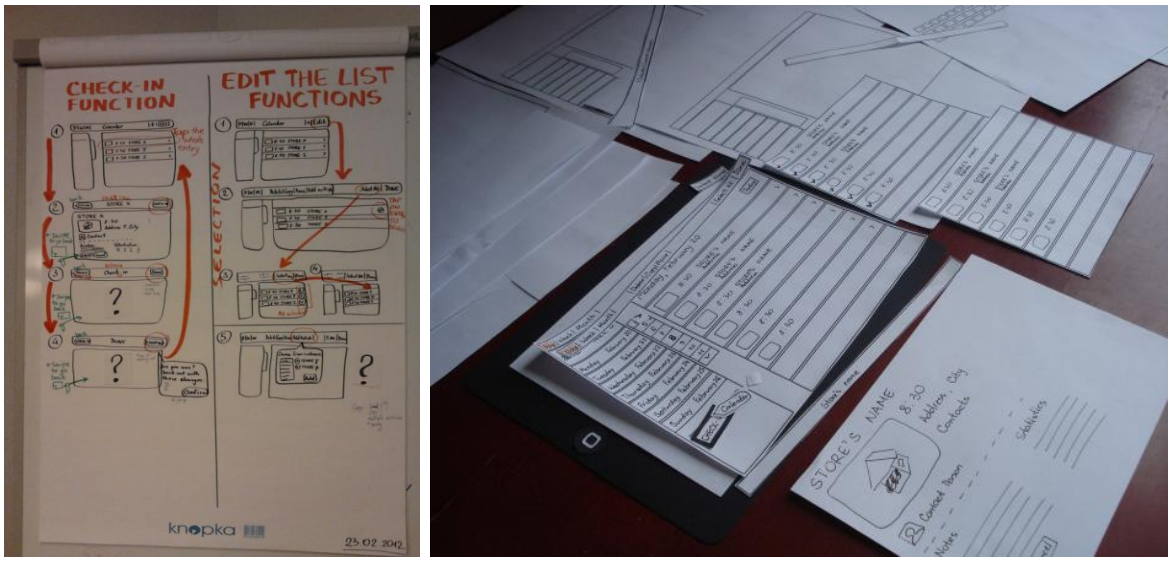


Figure 5.7 – Two ways of application prototypes

5.4.2 Findings

Kanbanery received positive feedback in general. Product Owner said that they always had a list of things to do, but using Kanban board helped to follow the throughput and allowed limiting the number of features developed simultaneously, so that they could get a well-working and tested increment of application. Senior Developer treated Kanbanery as one of the indicators of Agile team. Software architect had slightly different point of view, saying that real communication is more important than Kanbanery. He said that if he had worked in the office every day, he would organize Standups. Kanbanery is simply not enough. Designer was not satisfied with Kanbanery at all, claimed that it was inconvenient, and proposed an alternative tool Pivotaltracker with more human-

friendly interface, which, however, was not approved by Product Owner and therefore not implemented.

Nevertheless, the researcher's observations showed that Kanbanery tool was extremely helpful in terms of tracking workflow, automatic evaluation of tasks and getting distributed team members more involved into process. Figure 5.8 illustrates the main view of the tool.

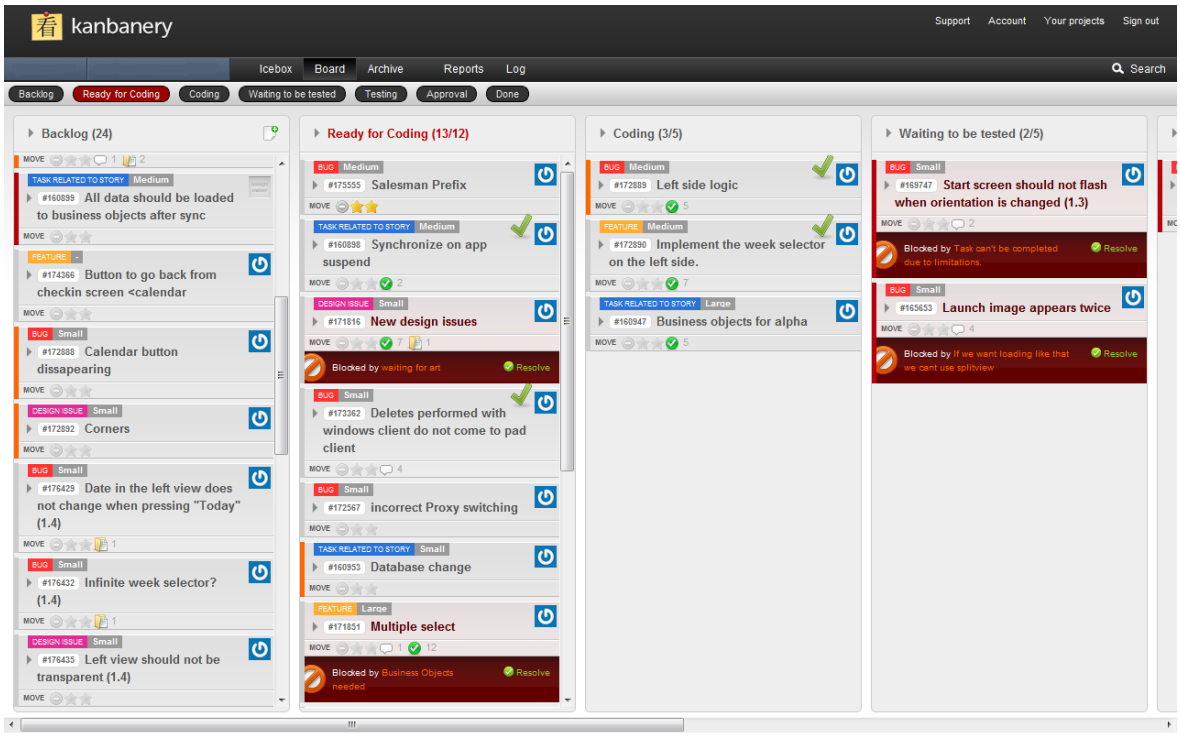


Figure 5.8 – Kanbanery project task management tool: main view

There are seven columns: Backlog, Ready for Coding, Coding, Waiting to Be Tested, Testing, Approval and Done. Kanban allowed to see graphical representation of tasks distributed by estimate, by type, by owner etc. (see Figure 5.9). And instead of Sprint Burndown Chart, created in the first cycle and remained unused haning on the wall, Kanbanery provided its own Cumulative Flow Chart illustrated in Figure 5.10.

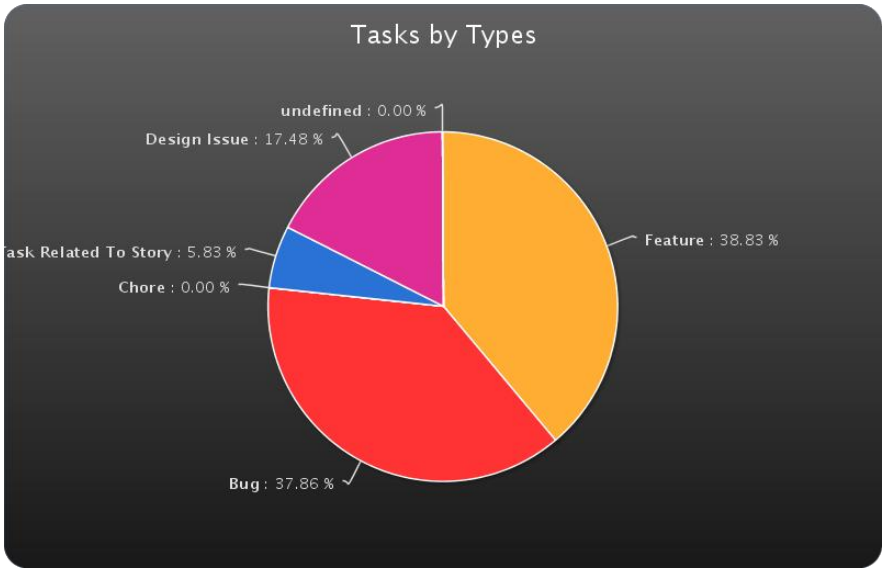


Figure 5.9 – Kanbanery pie chart illustrates proportion of tasks according to their types (Feature, Bug, Chore, Task Related to Story, Design Issue, and undefined)

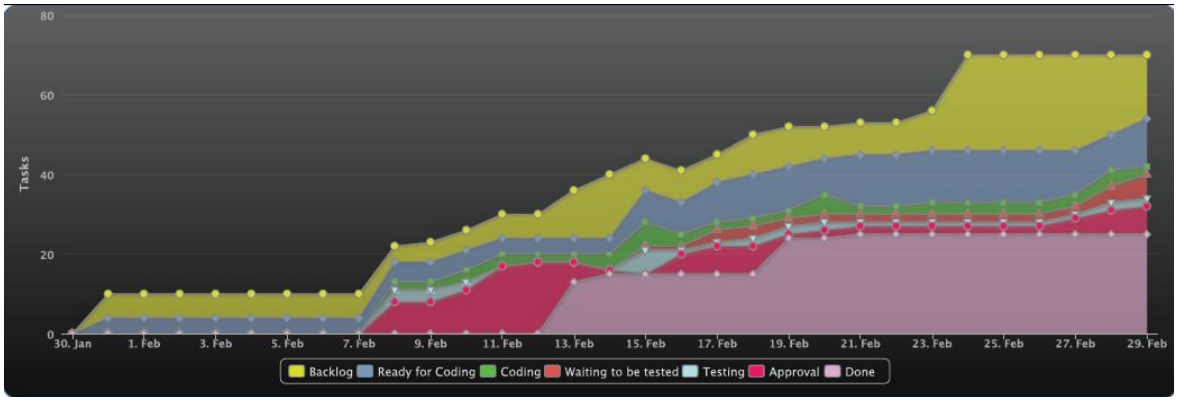


Figure 5.10 – Kanbanery Cumulative Flow Chart for the period January 30 – February 29, 2012.

During the third phase, frequent contact was established between Designer and Usability Tester. They started communicating by e-mail and Skype, and, as a result, all necessary graphical elements needed for coding the interface, were received in time, which allowed producing piece of working application iteratively. The concept map in Figure 5.11 illustrates three additional connections (compared to Figure 3.3).

This time prototypes received more attention comparing to previous cycle. People especially liked the whiteboard drawings, where everyone could add own notes.

To make prototypes interactive (otherwise they would be just mockups), some features were not drawn right on the board but on small pieces stuck to the board. They could be easily taken off and replaced. As a result, meeting went quite fast and gave positive outcomes. At least, everyone got common understanding of particular screens and functions that were seen on whiteboard. However, the method was inconvenient for the future use, since sheets on the whiteboard are constantly changing.

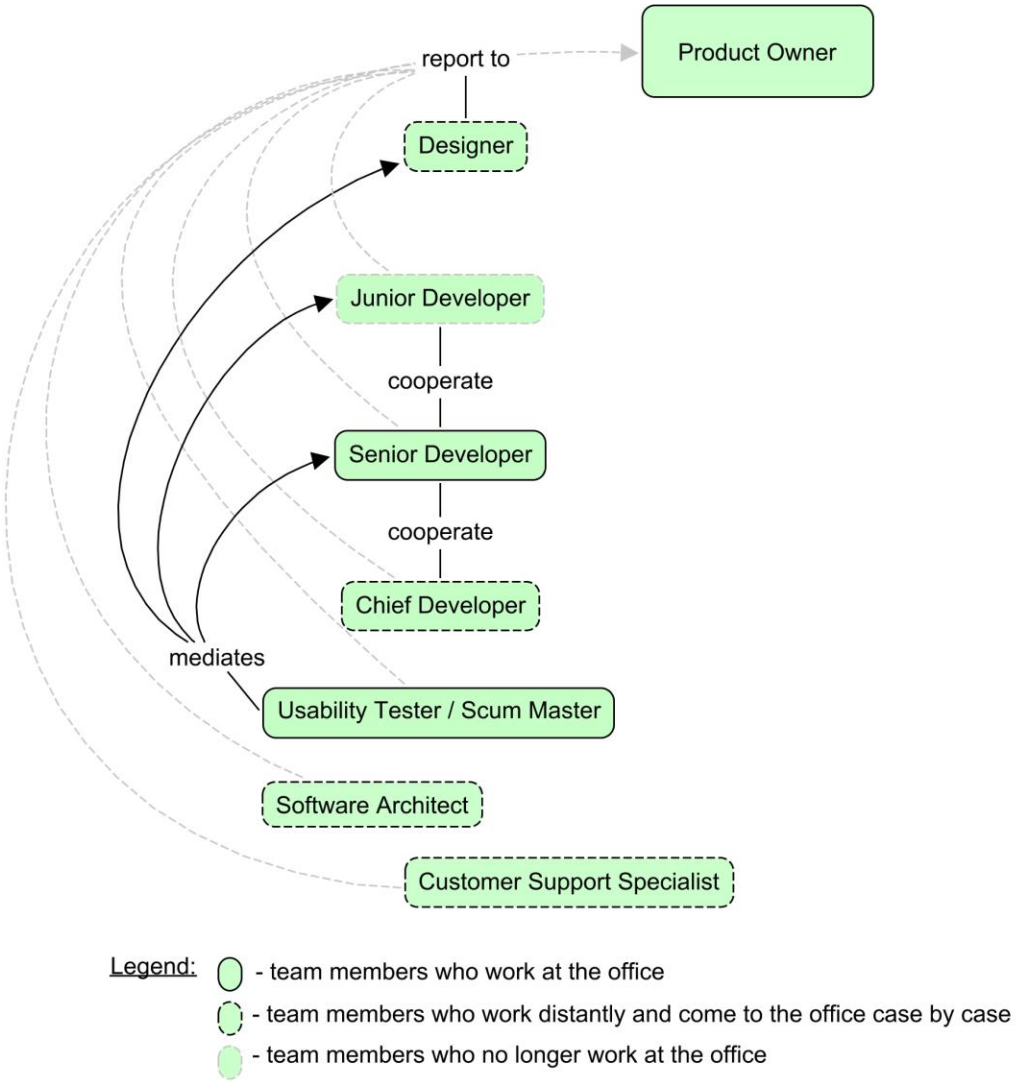


Figure 5.11 – New connections: Usability Tester as a mediator between Developers and Designer

5.4.3 Lessons Learned

Feedback and observations form the third cycle proved Cohn's (2011) hypothesis that there is no unique way of adopting Agile methodologies. Each company should do it in its own suitable way that matches conditions and purposes. We also realized that Scrum adoption might not go smooth right away. However, we should not draw a hasty conclusion that team does not fit Scrum or cannot be Agile at all. If at least one small positive tendency was noticed, it could become stronger and bring more results later on.

Moving closer to the last cycle of implementation period, here is list of goals that have not been yet investigated:

- Plan in advance what should be discussed during Scrum meeting
- Adopt some methods of Scrum specially for Designer
- Document iteratively, provide Designer brief overview of features needed soon

In addition, previous findings created a new goal, inspired by Software Architect's suggestion to use not only online Kanbanery tool but communicate more in real life.

5.5 Fourth Cycle: Not Ideal but Effective

5.5.1 Implementation

Changes in previous design session showed satisfying results, however the implemented approach has moved away from some of Scrum ideas, for example daily meetings (Standups). As we already learned, imposing ideas and forcing artificial adaptation can make individual's attitude even worse. To avoid such situation, new data retrieval was made. Turning to lists of code presented in

Annex A, there is an interesting setting called "Towards Ideal Situation". Interviewers gave their own recommendations about the perfect environment for the project, as they saw it. We joined the answers together and presented them as a Tag Cloud in Figure 5.12.

Figure 5.12 – Tag Cloud of concepts that could lead towards ideal working environment in Company Sigma

The Tag Cloud indicates frequently used *job-related* words: 'tasks', 'work', 'report', 'implement', 'responsible' and *collaboration-related* words: 'everyone', 'discuss', 'people', 'collaborate', 'communicate', 'meetings'. The word 'scrum' is also noticeable, which indicates that respondents see Scrum as a part of ideal future.

Implementation started in the beginning of March and lasted until April, but hopefully an effective Scrum approach, achieved by the fourth cycle, would last longer. The final elements to be adopted were:

1. Daily Standups
2. Improvised mockups
3. Kanban doubled on the wall
4. Dropbox
5. Planned meetings

In fact, daily Standups were not implemented due to personnel volatility in Company Sigma. One programmer has quit the job; hence managing director has been searching for new employees. There were only two people staying in the office every day during the fourth implementation cycle. Nevertheless, the rest 4 modifications were implemented and evaluated.

5.5.2 Findings

Improvised mockups, such as illustrated in Figure 5.13, appeared to be more convenient than white board sketches or accurate cutouts prepared in advance.

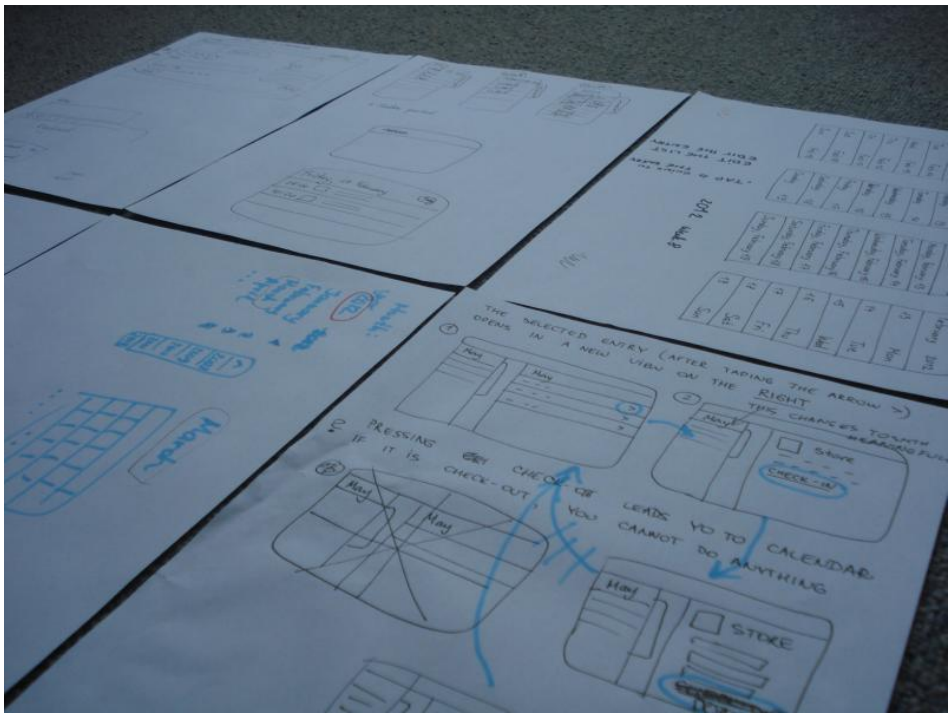


Figure 5.13 – Improvised application prototypes

Dropbox was implemented for files sharing with the Designer. Finally, we received .psd designs, which included all necessary elements separately (buttons, navigation bar, background etc). Senior Developer considered Dropbox as a helpful tool for such sharing. However, we did not go further than that and cannot state that introducing special Scrum techniques worked well for a Designer. To understand how critical it was, we decided to interview two members regarding

design and Scrum and asked them to sort 8 cards, representing Scrum-related ideas, by priority: the most important to the left (or top). Both respondents put a *"Special Approach to Design Issues"* card aside, saying that this was much less important than organizational tools and collaboration between team members.

Then we moved to the meeting schedule. It was organized so that participants moved towards main topics faster without unrelated conversations. There were subjects listed on the white board supplied with schemes and comments. Observation and direct participation proved that more issues were discussed than usual. Product Owner remained very pleased and happy.

Real-life version of Kanban project task management tool was the last modification made within the frames of this design research. Previously created Task Board had not been used already for two weeks. We reorganized the meeting room by creating a wall-copy of Kanbanery interface, where Developers put their tasks cards in the same way as they did it online (see Figure 5.14).

Figure 5.14 – New task tracking system looked exactly as online version of Kanbanery.

5.5.3 Lesson Learned

Four of five elements planned for this cycle were successfully adapted, which is a sufficient result. From this cycle we learned that forcing Designer to do Scrum should not be treated as a goal. Designer had his own vision of project management, and as long as his work was done, the team was satisfied. However, this did not match with our research problem – building an effective Scrum approach. In terms of this research we could not make Scrum effective for Designer. But if we analyze Designer's feedback (Annex A.1), we will see that Designer adequately understood what Scrum is. The point is that he did not distinguish Scrum from other methodologies, claiming that they all are about the same: *"It's a common Project Management"*. Apart from everything else, Designer knew that Scrum is about:

1. Doing something by certain deadline
2. Constant process of changing everything simultaneously
3. A-la demo version

What is more, Designer commented that the whole process was not properly organized. And when we asked for suggestions how to make it better, Designer proposed following solutions:

1. Someone distributes the tasks
2. *Designer does planning with Senior Developer*
3. *Make back-end first*
4. Group report
5. Plan program
6. *Prepare for the future*
7. *Stick to planned mockup*
8. I need information
9. *Senior developer should be responsible for Junior developer*

The proposals in cursive do not correspond with Scrum framework since they

imply traditional linear approach and hierarchy; whereas Scrum team should not be divided hierarchically.

Finally, the most interesting Designer's expression was: "*My task is to make a good, convenient and nice product. If I need to beat someone for it, I will beat someone for it*". On one hand, the attitude to the product as such is quite positive and shows the responsible approach to the matter. On the other hand, Designer treats his work separately from others and sees others separately from each other. However, this might be due to his freelance contract, which is a factor that cannot be changed within this research.

The rest team adopted modifications successfully. We planned in advance what should be discussed during Scrum meeting and started to communicate more in real life.

5.6 Implementation Analysis and Feedback

If we look back at all four implementation cycles, the full pass included 19 modifications presented in Table 5.2.

	Successful	**Partly successful**	**Not successful**
1	Scrum Master	Task Board	Pilot User Stories
2	Contact Designer regularly	Sprint Burndown Chart	Fixed Sprints
3	Improvised mockups	More frequent meetings	Fibonacci Numbers to evaluate tasks
4	Kanbanery online tool	Paper prototypes of application	More clear and simple User Stories
5	Kanbanery tool doubled on the wall	Involve people into using prototypes	Educative presentation about Scrum
6	Dropbox		Adopt some methods of Scrum specially for Designer
7	Preplanned topics for meetings		Daily Standups

Table 5.2 – The list of modifications done during the whole implementation period

From the first column we see that successfully adopted elements were mostly related to the tools being used. Scrum Master was positive in terms of managing several organizational moments, such as contacting Designer, being a mediator between Designer and Developers, planning meetings. Kanbanery tool lead us further from Scrum but was still an Agile approach. Improvised mockups were good replacement for paper prototypes, which also correspond to Scrum.

Partly successful modifications were highly related to Scrum. These are guidelines for further improvements, because the team should never stop when something is achieved. Continuous development is one of Scrum values. Scrum team always tries new processes and tools (see Figure 2.3).

The most crucial failed elements were User Stories and Sprints. To be more concrete, these are the basics of Scrum. But this does not mean that the whole Scrum has failed. For Company Sigma, all the achievements are valuable, because the company have not used *any* methodology before. Scrum was the first step towards being Agile.

In addition, we interviewed team members on the matter, what actually worked and what did not. People gave their general feedback, and we extracted concepts which indicate gaps in adapting Scrum. The most popular negative comments were regarding separately working people and low experience in iOS programming. Unfortunately, these variables could not be changed by our Scrum approach, because they are parts of working environment. Several issues that *should not* be part of Scrum were also named, such as: no concrete plan, broken hierarchy, no organization, weird mess. Scrum does not assume plans, hierarchy and might seem messy for someone who is used to traditional approach.

The most adequate comments regarding what did not work in Scrum are: no fixed sprints; the first task defined abstract; not focusing; meeting *only* once a week;

Scrum does not work at all; does not work to full extent; throwing out completed tasks; spent a lot of time; deadlines not valid; no enthusiasm.

While some respondents supposed that Scrum did not work, others said that the team worked relatively close to what Scrum is. Overall positive feedback includes these statements: good people; good company; interaction between team members; breaking application into parts; not a big resistance; communication case by case; connections are almost equally strong; meeting *at least* once a week; 70% percents of Scrum were successful; more than enough artifacts; potentially marketable product; building the version – version appears.

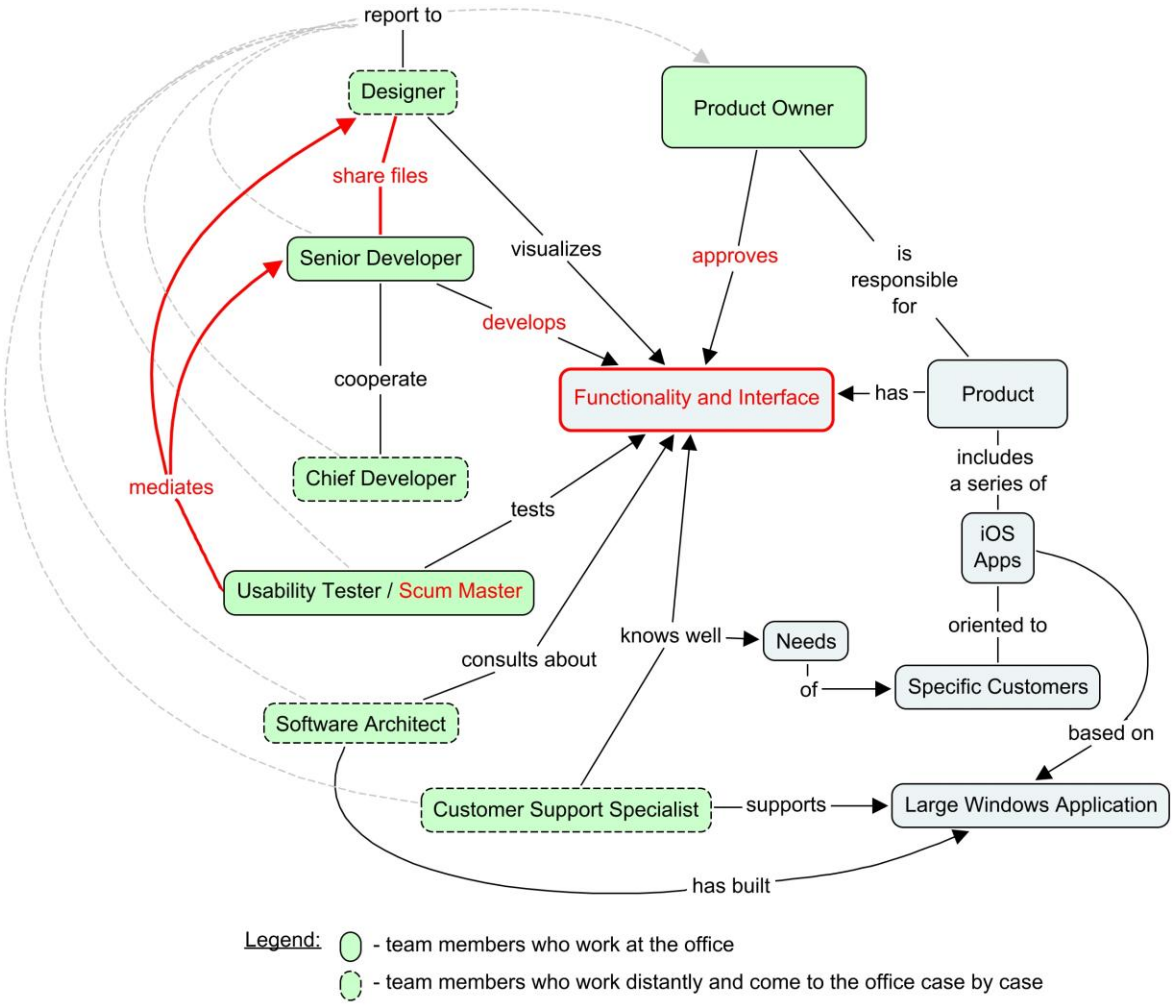


Figure 5.15 - Internal connections and responsibilities of team members in a Company Sigma at the last cycle of implementing Scrum

Turning back to our concept maps about Company Sigma initial and ideal infrastructures, here is a final model (Figure 5.15) that highlights the new connections appeared due to Scrum approach (marked with red color). However, there was one unexpected change: there is no Junior Developer anymore. Scrum Master is a mediator, Designer shares files, and Senior Developer develops functionality and interface, instead of just building. Product Owner rather approves than controls the process.

Chapter 6

Conclusion

The purpose of this master thesis is to design an effective Scrum approach for Company Sigma. Designed approach covers two major problems of the company: how to adapt Scrum and manage user interface design. This paper results in establishing the direction towards appropriate Agile techniques. Final recommendations can be sufficiently used by future employees and ensure their productivity.

At the beginning, Agile and Scrum are studied in their 'ideal' theoretical state to be further compared with the practical usage in Company Sigma. The overview of software development methodologies demonstrates that Agile is an iterative approach opposite to linear. Iterative approach excludes initial planning and focuses on constant changes.

We discovered that the term 'agile' is quite new and covers previously existing lightweight iterative incremental methodologies, as for instance Scrum. Scrum is commonly used as a synonym for Agile. This is so due to its overwhelming popularity as compared with other Agile approaches (e.g. RAD, Crystal, Lean, Kanban). By 2011 Scrum has been the most used framework. In this paper we refer to Scrum as a sub-term of Agile and do not interchange them.

After defining the core of Scrum, which is a self-organizing and cross-functional

team, we analyze perspective of adopting Scrum in a small company. Some of the interviewees claimed that Company Sigma is too small to follow Scrum. This statement was disclaimed, since an ideal Scrum team consists of 5-9 members, which is true for the case of Company Sigma.

One of company problems is managing user interface design. In order to cover this issue, we study whether Scrum can bridge the gap between designers and programmers. We conclude that design is indeed unlikely iterative and therefore hard to manage via Scrum. Scrum is not meant to bridge that gap.

The next step is to describe Company Sigma's environment, where our design object is implemented. We evaluate potential factors that can and cannot be affected by design research. Company's background, objective, product roadmap, and personnel volatility can be observed but not altered. Company's background points out the lack of experience in user interface design and iterative methodologies. The objective corresponds to Apple Human Interface Guidelines (HIG, 2011). Product roadmap ensures long application development cycle. And personnel volatility is important since one developer left company during the last implementation phase. Infrastructure and team engagement are modified during the implementing period which makes them corresponding to Agile methodologies.

Ethnographic approach reveals weak spots of Company Sigma application development process. The problems are discussed in details and turned into two major goals: better self-organization and more effort to maintain design and Scrum. These goals are taken into account when introducing the object of the research.

Designing an effective Scrum approach is split into four implementation cycles. The first cycle brings positive results and indicates that developers are able to use

Task Board and track their tasks. The second cycle is negative because it imposes artificial instruments that are not appreciated by team members. This highlights the importance of factors that cannot be changed. We also indicate that Company Sigma prioritizes tasks in the middle of the process and changes sprint length – not according to Scrum framework. The third cycle is devoted to deeper team analysis. Collected data shows that team members see each other as colleagues and do not use personal constructs as grouping factors. Hence we have a good potential for professional collaboration and self-organization. This cycle moves us away from pure Scrum framework to other Agile techniques, such as Kanbanery. However, this is still a positive change. Scrum Master is more active during this cycle and establishes contacts between Designer and developers. Overall estimation of the third cycle is "a turning point". The fourth cycle is the last one. It finally establishes an effective design approach. The goal is not to find the absolute solution but to make things work better than before. As a result, we have a team that is organized enough to track each others' tasks via digital Task Board. Distributed members are also involved. Team uses improvised mockups proving the ability to clarify things when needed. Developers constantly share files with Designer. Real-life Task Board is synchronized with digital version. Communication within the office has increased.

Thus, one part of effective Scrum approach is finished: team is self-organized. Another part is not completed as planned: Designer is left out of Scrum. However, we claim that unless Designer's tasks are ready in time and satisfy the requirements, it does not matter which methodology he uses. The effective approach in case of Company Sigma can be the following: involve as much team members as possible into Scrum, keep tracking others and slack their resistance. The most important outcomes are highlighted in the last part of Chapter 5.

To conclude, we can develop Scrum further and implement more cycles. But the

goal is to design an effective Scrum *approach*, where approach means *a way of dealing with a situation or problem*. The way of becoming Agile by starting with Scrum is found. We prove that the team should not be forced to use Scrum, but be free to choose only those methods, which are most suitable for company's environment. The list of such methods is provided in this research.

A future work direction includes continuous studying the additional problems related to Scrum in Company Sigma, such as two distributed teams, increasing the scope, dealing with newcomers, etc. A survey may be conducted in order to generalize the results and develop a set of unique Agile strategies for companies similar to Company Sigma. Another interesting notion to be further developed (partly covered with this thesis) is bridging the gap between designers and programmers via Agile methodologies.

Kokkuvõte

Käesolev magistritöö hõlmab agiilsete meetodite rakendamise propbleeme disainis ja programmeerimises. Töö on pühendatud firma Company Sigma töökeskonnale ning uurib kuidas see firma rakendab kõige populaarsemat agiilse tarkvara arendamise metoodit ehk Scrum´i. Uurimistööprobleemina on käsitletud Scrum´i kasutamine väikses firmas mis ei ole kunagi kasutanud mitte ühtegi konkreetset tarkvara arendusmetoodit. Lisaprobleemideks on enesejuhtimise puudus meeskonnas ja mittekompetsentsus disaini juhtimise valdkonnas.

Töö eesmärgiks on kujundada effektiivne üleminek Scrum´ile Company Sigma jaoks ning tagada selle ülemineku edukat edasiandmist tulevaste töötajatele. Õige üleminek vastab ka ettevõtte eesmärgile ning meeskonna liikmete vajadustele ja oskustele.

Käesolev uuring on läbi viidud kvalitatiivsete meetodite põhjal. Peamiselt kasutati disaini uurimust ja lisaks etnograafilist uurimust. Disaini objektina on välja toodud Scrum´i efektiivne rakendus spetsiifilises keskkonnas. Disaini uurimus koosneb neljast rakendustsüklist mis lõpevad üldstrateegia esitamisega. Etnograafilist uurimust on kasutatud uuringu osalejate tagasiside saamiseks Scrum´i rakendamise kohta. Uurimuse aluseks on teoreetiline ülevaade erinevatest agiilsetest meetoditest. Uurimuse strateegia põhineb disaini uurimuse kirjandusel.

Magistritöö tulemuseks on saanud firmale sobiv üleminek Scrum´ile mille põhimõtteks on võimalikult suure meeskonna liikmete hulga kaasamine, teiste

jälgimine ja nende vastutuleku vähendamine. Sai kinnitatud, et meeskonda ei pea sundima kasutama Scrum´i vaid neil võib lubada vastavaid meetodeid ise valida. Vastavate meetodite nimekiri on välja töötatud käesolevas töös.

Uurimuse edasine arendus võib keskenduda uute meetodite läbi töötamisele disaini ja programmeerimise ühendamiseks Scrum´i raamides, sest hetkel on näha vastava teooria puudust.

Annex A

List of Codes Retrieved from Interviews

A.1 Codes from Interview with Designer

Settings	Acts	Activities	Meanings	Participation	Relationships
Role in the project	Retrieving information; Collecting grain by grain; Beat someone if needed;	Visualization of interface; Responsible for usability; Front-end;	No need in full-time designer;	Freelancer; There are no mockups;	Designers are also usability testers
Attitude to Scrum	Doing something by certain deadline;	Constant process of changing everything simultaneously;	Should have job experience; A-la demo version; Processes are all the same; Three ways of managing design;	User stories were not clear to anybody; My task is to make a good, convenient and nice product;	Need a person who will push the whole project; Leader; It's a common Project Management;
Current situation	Searching for concrete information;	No concrete plan; Good ideas;	Hierarchy and organization is broken; Weird mess is happening;	Everyone is responsible for his work;	Constant presence is not required; Good people, good company;
Towards ideal situation	Someone distributes the tasks; Designer plans with senior developer; Make back-end first;	Group report; Plan program; Prepare for the future; Stick to planned mockup; I need information;	Job experience; Responsible people;	The whole work can be done in 2 months; Software architect and Junior developer report to Senior Developer;	Product Owner is someone from developers; Senior developer should be responsible for Junior

A.2 Codes from Interview with Product Owner

Settings	Acts	Activities	Meanings	Participation	Relationships
Role in the project	Give tasks; Making product-do stuff;	Proposing new ideas;	Interesting for the customers;	Without needing to exactly tell how they are done;	I talk with Chief developer about the general architecture, something inside; Give tasks to Senior developer ; Junior Developer needs more assistance;
Attitude to Scrum	Thinking; Understand; Criticize; Agree	Formalized way of doing Agile; Arguing; Forcing things to be the same;	Well-documented; Easy to understand for everybody; Sacrifice quality; Rather theoretical; Hope we just converted it;	Haven't had a formal methodology previously; Not a huge enthusiasm, not a big resistance either;	Agile is an umbrella term for Scrum; Artificial connection;
Current situation	Working relatively close to what is Scrum; Stopped fixing sprints;	Interaction; Breaking application into parts;	Depending who is working on what;	Don't so much have Junior Developer; Don't have a big resistance	Communication case by case; All links are about the strong; Connections are almost equally strong; Chief developer – general architecture, something inside;
Towards ideal situation	Introduce morning meetings; Update the board; Sorted already; Discuss; Practice; Changes become smaller and smaller;	Making it a little bit more methodological; Deifning stroies when they are implemented	Design is important; Problems of design come out quicker; Technical mistakes are more expensive; Nobody will see underlying hidden architectural decisions;	Finish first what you are doing and then you text the next one; Smoother and smoother;	Design has to go in same steps

A.3 Codes from Interview with Junior Developer

Settings	Acts	Activities	Meanings	Participation	Relationships
Role in the project	Do; Write; Work; Give something to show;	Building the interface of iOS App; Implementing logic; Writing controllers; Working with the code;	Interesting; For the future;	Tasks; Containers; Views; Transitions; Prototypes;	Designing; The only who was dealing with iPad for the first three months;
Attitude to Scrum	Joined the team; Agile motivates programmers; Push their knowledge;	Trying; Studying deeply; Follow the discipline; Achieve results; Succeed;	Interested; The best solution; Productive; Good results; The discipline; Motivation;	Agile techniques; Scrum; Kanban; Extreme programming; Test-driven development	Course and lectures at the university; Schedule;
Current situation	Meeting once a week; Trying; Struggling; Go down; Realized; The first task defined abstract; Were not there;	Had difficulties; Can be improved; Not focusing; Having technical problems;	Useless; Separated; Absent; Not enough for being effective; Not enough experience; 70% percents of Scrum; Successful; More than enough artifacts; Conservative;	Meeting only once a week; Framework; Rules of Scrum; Potentially marketable product; No technique used before; One-month sprint;	Only two local developers; Designer; Software architect; Product owner; Did not have people Programmers abroad; Part-time workers;
Towards ideal situation	Be ready Doing; Define the task; Break tasks into small tasks; Work forward; They really didn't care; Release sort of a product; Cannot waste a lot of time; Doing nothing; Work; Divide tasks into subtasks; Communicate; Tell; Do;	Any technique can be tried or applied; Implement; Achieve the result; Undertaking tasks;	Confident; Attitude of some programmers; New people; Flexible; Deadlines; Cycle; Scheduling time; High and low priority tasks; Do something better	Automated testing; Time; Team of five people; Daily Scrum meeting;	Communicate with each other; Agile programming

A.4 Codes from Interview with Senior Developer

Settings	Acts	Activities	Meanings	Participation	Relationships
Role in the project	Trying to do;	Tasks are established; Decisions are made; Lots of tasks; Regular release of application;	In recognition of years spent in a company;		How is my role different from others – I don't know; Don't have anyone under the ferule; No hierarchy; Based on trust;
Attitude to Scrum	Is not very suitable; Cannot even speak about it; I am skeptical about it; All depends on the result;	Everything is predefined by manages;	Has to be 7 people; Not applicable; How can I support something that ends up bad; In sake of an idea – it's stupid;	Our enterprise; Must be a team; Unaware about all advantages; Scrum is somewhere outside;	There are at least seven people in rugby;
Current situation	It is going more or less better now; Doing something; I build the version; Version appears; Scrum does not work at all; Throwing out completed tasks; Spent a lot of time;	Everything goes very slowly; iOS took much time; Putting limits; Rotation;	Don't know exactly what else is needed; Small companies are always Agile; Not enough experience; iOS specifics; Upsetting; Deadlines not valid;	No enthusiasm; Enough online and physical instruments; Product Owner participates in design; Customer Support Specialist does not take part in a project	Too few people; Not enough people; Design did not affect implementation of Agile; Designer is not interested in Agile; Me and Chief Developer;
Towards ideal situation	Getting feasible tasks; Spin around; Can't make many tasks fast; Discuss right away;	Collaboration between colleagues; Devote time; Everyone works in the name of the goal, not in the name of Agile;	My work should be really needed for somebody;	Ready to take part in organizational moments of Scrum; If there is opportunity; All are equally interested;	People; Everyone synchronizes task board; no special approach to design; Programmers don't need to (but may) collaborate with each other; Instructions of Scrum Master not important

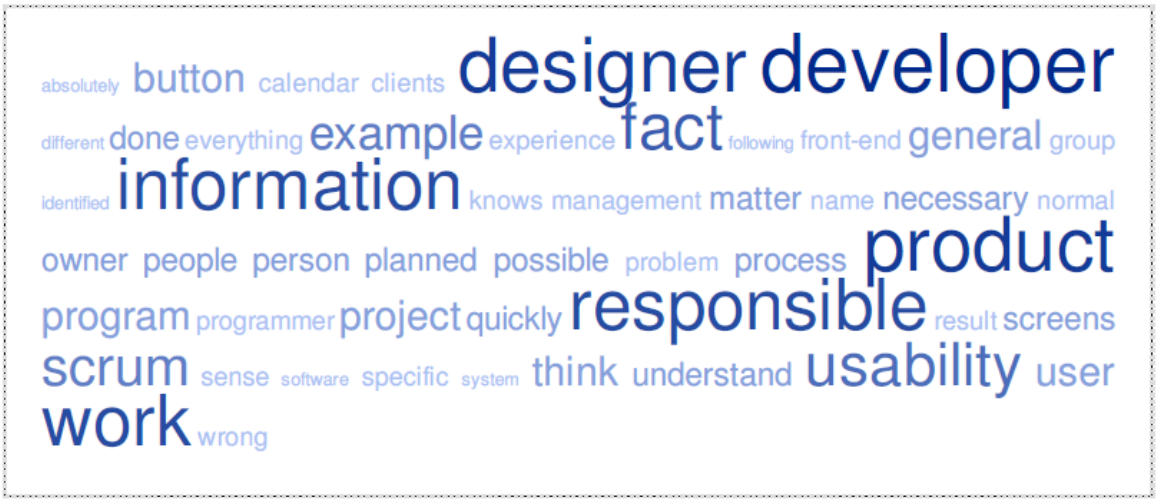
A.5 Codes from Interview with Software Architect

Settings	Acts	Activities	Meanings	Participation	Relationships
Role in the project	Find algorithms	Inheritance	Based on code for the previous product	Functionality I know everybody	No communication with tester; Hierarchy;
Attitude to Scrum	Helps; Doesn't fully work; Tasks are chosen;	Prioritization; Sprints existed before Scrum;	If you don't have enough experience in prioritizing tasks;	Started using Kanban	I know everybofy;
Current situation	Discussing; Ready for coding; We are trying;	Sprints don't really work; Does not work to full extent;	No implementation; Task is not reproducible;	Isolation;	Design and architecture; Chief Developer and Designer don't communicate;
Towards ideal situation	Organizing Standups; Gather together; Discuss weekly tasks and problems;	Live communication; Meetings should be organized on different days.	Not bad;	Everyone is aware of what others are doing;	Real contacts; Not only online

Annex B

Tag Clouds

B.1 Designer Tag Cloud



B.2 Product Owner Tag Cloud

B.3 Junior Developer Tag Cloud

B.4 Senior Developer Tag Cloud

B.4 Software Architect Tag Cloud

References

1. Ambler, S. W. (2010). Agile Modeling. Agile Design. Ambysoft Inc. Retrieved from http://www.agilemodeling.com/essays/agileDesign.htm #Philosophies

2. Apple, Inc. (2011) *iOS Human Interface Guidelines [HIG]. User Experience.* Retrieved from http://developer.apple.com/library/ios/documentation/ UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

3. Apple, Inc. (2011). Apple's App Store Downloads Top 15 Billion. *Apple Press Info.* Retrieved from http://www.apple.com/pr/library/2011/07/07Apples-App-Store-Downloads-Top-15-Billion.html

4. Arslan, Y. (2012). What is wrong with design and Scrum? Retrieved from http://yusufarslan.net/what- wrong-design-and-scrum

5. Beck, K., Beedle, M. et al. (2011). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/

6. Brown, J. M., Lindgaard, G., & Biddle, R. (2012). Interactional Identity: Designers and Developers Making Joint Work Meaningful and Effective. *Qualitative Studies of Software Development II,* 1381-1390.

7. Centers for Medicare and Medicaid Services [CMS]. (2008). Selecting a development approach. Department of Health and Human Services. Retrieved from http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/SystemLifecycleFramework/downloads/SelectingDevelopmentApproach.pdf

8. Cocburn, A. (2008). Using Both Incremental and Iterative Development. *Humans and Technology. Crosstalk*. Retreived from http://www.crosstalkonline.org/storage/issue-archives/2008/200805/200805-Cockburn.pdf

9. Cohen L., Manion, L., & Morrison, K. (2007). *Research Methods in Education.* (6th ed.). London and New York: Routledge.

10. Cohn, M. (2010). *Succeeding With Agile: Software Development Using Scrum*. Upper Saddle River, NJ: Addison-Wesley.

11. Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *The Journal of the Learning Sciences*, 13(1), 15-42.

12. Cooper, A., Reimann, R., & Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design* (3rd ed.). Indianapolis: Wiley Publishing.

13. Design. (2010). In Stevenson, A. & Lindberg, C.A. (Ed.), *The New Oxford American Dictionary* (3rd ed.). Oxford: Oxford University Press. Retrieved April 23, 2012 from Dictionary, Version 2.2.2 (118.1), Apple Inc.

14. Discovery Learning, Inc. (2003). Research Summary Number 8. Change Style Indicator and MBTI – Is there a connection? Retrieved from http://www.discoverylearning.com/

15. Hassenzahl, M., & Wessler, R. (2000). Capturing Design Space From a User Perspective: The Repertory Grid Technique Revisited. *International Journal Of Human–Computer Interaction*, 12(3&4), 441–459. Retrieved from http://www.uni-landau.de/hassenzahl/pdfs/ijhci_hassenzahl _wessler00.pdf

16. IDEO. (2003) *IDEO Method Cards: 51 Ways to Inspire Design*. William Stout. Retreived from http://students.washington.edu/kenliu8/INFO360/IDEO MethodCards.pdf

17. Infrastructure. (2010). In Stevenson, A. & Lindberg, C.A. (Ed.), *The New Oxford American Dictionary* (3rd ed.). Oxford: Oxford University Press. Retrieved April 23, 2012 from Dictionary, Version 2.2.2 (118.1), Apple Inc.

18. Jensen, J. F. (1998). Interactivity: Tracking a new concept in media and communication studies. *Nordicom Review*, 19(1), 185-204. Retrieved from http://www.nordicom.gu.se/common/publ_pdf/38_jensen.pdf

19. Koskinen, I., Zimmerman, J., Binder, T., Redstrom, J., & Wensveen, S. (2011). *Design Research Through Practice. From the Lab, Field, and Showroom* (1st ed.). Waltham: Morgan Kaufmann.

20. Larman, C. & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. *Computer* 36, 47-56. IEEE Computer Society. Retrieved from http://ftp.rta.nato.int/public//PubFullText/RTO/TR/RTO-TR-IST-026///TR-IST-026-ANN-E.pdf

21. Linear. (2010). In Stevenson, A. & Lindberg, C.A. (Ed.), *The New Oxford American Dictionary* (3rd ed.). Oxford: Oxford University Press. Retrieved from Dictionary, Version 2.2.2 (118.1), Apple Inc.

22. Lofland, J. & Lofland, L. H. (1995). *Analyzing social settings* (3rd ed.). Belmont: Wadsworth.

23. Luecke, R. (2003). *Managing Change and Transition*. Boston: Harvard Business School Press.

24. Manning, A. (2008). Agile Designer / Developer collaboration with Scrum. Retrieved from http://www.allenmanning.com/?p=17

25. Mountain Goat Software. (2012). What is Scrum. *Introduction to Scrum.* Retrieved from http://www.mountaingoatsoftware.com/topics/scrum

26. Nelson, B., Ketelhut, D. J., Clarke, J., Bowman, C., & Dede, C. (2004). *Design-based Research Strategies for Developing a Scientific Inquiry Curriculum in a Multi-User Virtual Environment*. Cambridge: Harvard University.

27. Norman, D. A. (2002). *The design of everyday things.* New York: Basic Books.

28. Patton, M. Q. (1980). *Qualitative evaluation methods*. London: Sage.

29. Pledgerwood. (2012). The Relationship of Scrum to Agile. *The Cutting Ledge.* Retrieved from http://thecuttingledge.com/?p=198

30. Pugh, K. (2011). *Lean-Agile Acceptance Test-Driven Development Better Software Through Collaboration*. Upper Saddle River, NJ: Addison-Wesley.

31. Rasmusson, J. (2010). *The Agile Samurai. How Agile Masters Deliver Great Software.* Raleigh, North Carolina Dallas, Texas: The Pragmatic Bookshelf.

32. Ratcliff, D. (n.d.). 15 Methods of Data Analysis in Qualitative Research. Retrieved from http://qualitativeresearch.ratcliffs.net/

33. Royce, W. (1970). Managing the Development of Large Software Systems. *Proceedings, IEEE WESCON* 26, 328–338. Retrieved from http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

34. Sims C., Johnson H. L. (2011). *The Elements of Scrum*. *Version 1.01.* Foster City, CA: Dymaxicon.

35. Welch, S. (2011). *From Idea to App: Creating iOS UI, Animations, and Gestures*. Berkeley: New Riders Press

36. VersionOne, LLC. (2007). 2nd Annual Survey. "The State of Agile Development". Conducted: June-July 2007. Survey Highlights & Full Data

Report. Retrieved from http://www.versionone.com/pdf/StateOfAgile Developmet2_FullDataReport.pdf

37. VersionOne, LLC. (2007). Survey: "The State of Agile Development". Retrieved from http://trailridgeconsulting.com/surveys/state-of-agile-development-survey-2006.pdf

38. VersionOne, LLC. (2008). 3rd Annual Survey: 2008. "The State of Agile Development". Conducted: June-July 2008. Full Data Report. Retrieved from http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf

39. VersionOne, LLC. (2009-2010). State of Agile Survey. 2009. "The State of Agile Development". 4th Annual. Retrieved from http://www.versionone.com/pdf/2009_State_of_Agile_Development_Survey_Results.pdf

40. VersionOne, LLC. (2010). State of Agile Survey. 2010. "The State of Agile Development". 5th Annual. Retrieved from http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf

41. VersionOne, LLC. (2012). State of Agile Survey. 2011. "The State of Agile Development". 6th Annual. Retrieved from http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf

42. Williams, L. (2007). A Survey of Agile Development Methodologies. Retrieved from http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf