Agile methodologies & Scrum:  
When is a team ready to start using it?

Author: Chrysomalli Eleni
Supervisor: Stamatiou Ioannis

Three-member Examining Board:
Karayani Despoina
Peppas Pavlos
Stamatiou Ioannis

February 22, 2018
Acknowledgments

This is the point where I must acknowledge that this thesis is not an individual piece of work. On the contrary, it is the result of the interaction and help of many people.

I would like to begin by thanking all Albelli employees for their time, support and valuable help. This research could not have been done without them. Thank you wholeheartedly!

Afterwards, I want to thank my supervisor, Mr. Yianni Stamatiou, for the support and the trust he has shown me throughout this period. His encouraging words have always made me want to become better.

A big thank you to Albelli's front-end(e commerce team), and more specifically to Eveline Sparreboom, Mascha Welling, Emanuele Pane, Constant Zomer & Robbert-Jan Klein for the trust they showed me when I even did not believe in myself. I feel very lucky that I had the chance to work with each of you individually and to learn so many things from all of you. You are all special to me.

Continuing, I would like to thank my family for their unlimited love, support and continuous encouragement.

A special thank you to my close friends. I recognize and I am grateful for everything you have done for me and a further great thank you to my sisterly friend Georgia Tsekoura.

Finally, this diploma thesis is devoted to two people:
I. My grandmother, who died two years ago but has always been always a model for me and a source of inspiration. She taught me how not to quit and that there is nothing more precious than being alive one more day.
II. Miguel Pereira, who has been for the last 1.5 year, a true mentor for me. His support, his immediate help, his patience and his guidance have helped me mature professionally and personally.

Thank you all!
Table of contents

Abstract ................................................................................................................................. 8
Introduction ........................................................................................................................ 9
  Purposes and goals of research .................................................................................. 9
  Structure of thesis .................................................................................................. 10
Chapter 1: Waterfall versus Agile .................................................................................... 12
  1.1 Waterfall Model design .................................................................................... 12
  1.2 Agile Methodologies .......................................................................................... 15
Chapter 2: Most commonly used Agile Methods & Practices ......................................... 24
  2.1 Introduction ......................................................................................................... 24
  2.2 Agile Methodologies ............................................................................................ 24
    2.2.1 Adaptive software development (ASD) ....................................................... 24
    2.2.2 Agile modelling ......................................................................................... 26
    2.2.3 Crystal Clear methods: ............................................................................. 26
    2.2.4 Dynamic systems development method (DSDM): ..................................... 27
    2.2.5 Extreme systems development method (XP): ........................................... 28
    2.2.6 Feature-driven development (FDD): ......................................................... 29
    2.2.7 Lean software development: ..................................................................... 29
    2.2.8 Kanban ....................................................................................................... 30
    2.2.9 Scrum ......................................................................................................... 31
    2.2.10 Scrumban ................................................................................................. 32
    2.2.11 Rapid application development (RAD) ..................................................... 32
  2.3 Agile practices ........................................................................................................ 33
    2.3.1 Acceptance test-driven development (ATDD) ............................................ 33
    2.3.2 Behaviour-driven development (BDD) ...................................................... 33
    2.3.3 Business analyst designer method (BADM) .............................................. 34
    2.3.4 Cross-functional team ............................................................................... 35
    2.3.5 Continuous integration (CI): .................................................................... 36
    2.3.6 Domain-driven design (DDD) .................................................................... 36
    2.3.7 Pair programming ..................................................................................... 36
    2.3.8 Planning poker ........................................................................................... 37
    2.3.9 Refactoring ................................................................................................. 37
    2.3.10 Test-driven development (TDD) ............................................................... 38
    2.3.11 User story ................................................................................................ 38
    2.3.12 Story-driven modelling ......................................................................... 38
Chapter 5: Analysis of Results

5.1 Demographic characteristics ................................................. 60
5.2 Using scrum at the moment (categories 1,2,3) .......................... 65
5.3 Team Member Using Scrum Now ........................................ 67
5.4 Product Owner Using Scrum Now ..................................... 77
5.5 Scrum Master Using Scrum Now ...................................... 80
Graph 8: Ability of communication of ideas ................................................................. 75
Graph 9: Satisfaction by scrum .................................................................................. 77
Graph 10: Personal abilities of team members ......................................................... 78
Graph 11: Abilities of personalities of Scrum Masters ............................................ 81
Graph 12: Unexpected cases-Knowledge sharing- Collaboration- Communication in the team ................................................................. 81
Graph 13: Which of these is NOT a characteristic of Waterfall method of software development? ........................................................................... 82
Graph 14: How satisfied are you with Scrum ............................................................ 83
Graph 15: Personal abilities of team members that have used scrum ....................... 84
Graph 16: Time needed-distraction-lead of the process and daily tasks in the team .... 85
Graph 17: Estimation of agile teams ......................................................................... 89
Graph 18: Projects and progress of other teams ....................................................... 89
Graph 19: Profil of team members having used agile ................................................ 91
Graph 20: Collaboration among team members ....................................................... 92
Graph 21: Which of these is NOT a characteristic of the Waterfall method of software development ................................................................. 93
Graph 22: How satisfied are you with Agile .............................................................. 94
Graph 23: Age of team members .............................................................................. 96
Graph 24: Total work experience ............................................................................ 96
Graph 25: Personality of team members .................................................................. 97
Graph 26: Matureness-Knowledge sharing-Workflow-Communication in the team ... 98
Graph 27: How satisfied are you by the collaboration achieved among the team members? ............................................................................. 99
Graph 28: Disagreements in the team ..................................................................... 99
Graph 29: Do you believe that your team could achieve better results if they would listen to your opinion more often; .......................................................................... 100
Graph 30: Agile Software development .................................................................. 100
Graph 31: Way of working ..................................................................................... 101

List of tables
Table 1: Pros and cons of Waterfall model .................................................................. 14
Table 2: Demographic characteristics ..................................................................... 61
Table 3: How satisfied are you with scrum? ................................................................ 65
Table 4: Descriptive Statistics of personalities .................................................................. 67
Table 5: Do you believe that you work with big complex projects that cannot be broken down? ........................................................................... 70
Table 6: How much time do you usually need to finish a unit of work? ....................... 71
Table 7: Does the team usually agree with its PO decisions? ........................................ 76
Table 8: Do you feel that you provide enough feedback and transparency regarding the cases you work on to your Product Owner? .................................................................. 76
Table 9: How often do you do pair programming in your team? .................................. 76
Table 10: Do you recognize the value of your daily meetings? .................................... 77
Table 11: Do you feel your team agrees with your decisions? ..................................... 79
Table 12: Do you feel that you respect his role? .......................................................... 79
Table 13: How satisfied are you with scrum? ............................................................ 80
Table 14: How often do you do pair programming in your team? .................................. 83
Table 15: Do you recognize the value of your daily meetings? .................................... 83
Table 16: Did the team use to agree with its PO decisions? ...........................................86
Table 17: Do you feel that you used to provide enough feedback and transparency regarding the cases you were working on to your Product Owner? ...........................................86
Table 18: How often did you do pair programming? ........................................................86
Table 19: Did you recognize the value of your daily meetings? ........................................86
Table 20: Personal characteristics of Scrum masters that has used scrum ........................87
Table 21: Were you satisfied by the time your sprint lasted? ...........................................90
Table 22: Were the team’s goals realistic? ......................................................................90
Table 23: Did you recognize the value of your daily meetings? ........................................90
Table 24: Which framework are you using? .................................................................94
Table 25: Why did you decide on this framework? .........................................................95
Table 26: Would you consider switching to Scrum? ......................................................95
Table 27: Which of these is NOT a characteristic of the waterfall method of software development? .................................................................101
Table 29: On a scale 1-5 how confident do you feel about this change? ......................102
Table 30: Correlation among decription, organization and distraction of team members ...103
Table 31: Correlation among communication of ideas and lead of the process ..............103
Table 32: Correlation among communication of ideas, lead of the process and retrospectives .................................................................103
Table 33: Correlation among creativity and other characteristics of team members ....104
Table 34: Correlations among daily tasks and other variables ..................................104
Table 35: Correlations among workflow and collaboration among team members ......106
Table 36: Correlations among matureness of the team and knowledge sharing ..........107
Table 37: Correlations among "True ScrumMaster" and other variables .....................107
Table 38: Correlation among satisfaction by scrum and collaboration of the team ......108
Table 39: Correlations among personal characteristics ................................................109
Table 40: Correlations among abilities and skills .......................................................110
Table 41: Correlation among ability of communication and this of explanation in detail ....110
Table 42: Correlation among daily tasks, matureness and workflow of the team ..........111
Table 43: Correlation between the workflow and the matureness of the team ..............111
Table 44: Correlations among workflow and collaboration of the team ......................112
Table 45: Correlation among Knowledge sharing and team distraction ......................113
Table 46: Correlation among abilities of communication-explanation in detail ..........113
Table 47: Correlation among abilities of creativity-collaboration in the team ..............113
Table 48: Correlation among abilities of daily tasks-matureness in the team ..............114
Table 49: Correlation among abilities of workflow-collaboration and matureness in the team ..................................................................................114

Page 7 of 121
Abstract

This research is aimed at exploring the factors that lead a group to use Scrum methodology optimally and effectively.

The method of measurement is done with the help of a questionnaire whose questions were formulated by the researcher herself. The sample collected for the survey consists of 59 people working at Albelli, based in Amsterdam, the Netherlands.

In the framework of the research effort, extensive use of the static analysis (descriptive analysis, frequency analysis and batch correlation analysis) is used to extract the statistical conclusions. For this purpose, the Spss 20 statistical package was used.

The results of the survey reveal that while the demographic and personal attributes of a group's members may affect the results, 88% success depends on the group's composition: the workflow followed, the sharing of knowledge among the members, the creation of a stable and safe environment in which each member can freely express their opinion. Also, emphasis should be placed on group unity and support for finding solutions to BAU / unexpected cases / legacy systems / larger and more complicated projects.

Finally equally important is the right choice of people for the role of Scrum Master that can inspire the team, help it add value to each meeting and recognise its needs as well as the Product owner to improve communication with the stakeholders, without violating the role of the Scrum Master.
Introduction

“One of the first questions we ask ourselves for each of our project implementations is “Which development methodology should we use?” This is a topic that gets a lot of discussion as it’s the way of organizing the work for the project and not as often misinterpreted about a style of project management or a specific technical approach. The two basic and most popular methodologies are:

1. Waterfall: which is known as the “traditional” approach, and
2. Agile: a specific type of Rapid Application Development and newer than Waterfall, but not that new, which is often implemented using Scrum.

Both are usable, mature methodologies. Lately Agile and especially Scrum have become more and more popular. A lot of people in higher management see the agile way of working as the key to success. But is this actually true? Have we found the silver bullet? Can every individual work in an agile way? If yes, does this mean that the team that he is part of will also adopt and use the agile way of working and thinking successfully?

All those questions triggered this research.

Purposes and goals of research

The research took place within a Dutch company called “Albelli”, in which I have been working, as a software developer, for the last year.

The company at this point is in the middle of an agile transformation, creating cross-functional squads and training all its employees in order for them to be able to work in an agile way. The IT teams have been already using agile (Scrum-Kanban) for the last couple of years, so we tried to find out what is successful and what is not according to their experience. Why did they fail (when they did) and which parts of Agile and more specific of Scrum they found appealing and helpful and which not.

We continued the research with people working in different departments (affiliate marketing, online marketing, brand and design, UX) that have recently started using agile or they are under training to start using it in the near future. The reason for that was to capture their feelings about this new way of working.

Based on the above we have the following scenarios of people joining the research:

1. Team Member Using Scrum Now
2. Product Owner Using Scrum Now
3. Scrum Master Using Scrum Now
4. Team Member that has used Scrum
5. Product Owner that has used Scrum
6. Scrum Master that has used Scrum
7. Team member that has used Agile in the past
8. Team member that plans to use Agile in the future

And those basic factors under examination:
- Demographic characteristics
- Personal characteristics/Qualities of the person
- Team values
- Interaction within the team
- Qualities of the scrum master and the product owner
- Projects of the team
- Evaluation of Agile Framework.

Our final goal is to define the combinations of factors that might make it easier for a team to work using agile successfully.

Structure of thesis

This thesis consists of 5 chapters.

We start by providing basic information about both methodologies (Waterfall and Agile). We try to make the differences between the two methodologies visible and by giving more emphasis to the second one we present the history behind the methodology, the advantages and disadvantages and we try to come to a conclusion on what is agile and what is not. Next, at chapter two we briefly present the basic Agile Methods and Practices.

We continue with chapter 3, that we explain Scrum and all the ceremonies and roles related to this methodology. In chapter 4 we explain in detail the methodology of the
research, we present the sample of the research, the method used in order to collect that
data and the measurement method used. Then we refer to the analysis and data processing.
We proceed with the analysis of the results of our research by describing in detail the
descriptive statistics, the frequencies of all categories and the correlation among them. All
that information should make it easier for the reader to follow, deeper understand and
challenge the research and its results.
Chapter 1: Waterfall versus Agile

1.1 Waterfall Model design

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

In more details, waterfall approach was first SDLC (System Development Life Cycle) Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.

![Figure 1: Waterfall software development life cycle model](Source: (Gingco, 2016))

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture.
• **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

• **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

• **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

• **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

### Waterfall Model Pros & Con

The following table lists out the pros and cons of Waterfall model:
Table 1: Pros and cons of Waterfall model

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Simple and easy to understand and use</td>
<td>• No working software is produced until late during the life cycle.</td>
</tr>
<tr>
<td>• Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.</td>
<td>• High amounts of risk and uncertainty.</td>
</tr>
<tr>
<td>• Phases are processed and completed one at a time.</td>
<td>• Not a good model for complex and object-oriented projects.</td>
</tr>
<tr>
<td>• Works well for smaller projects where requirements are very well understood.</td>
<td>• Poor model for long and ongoing projects.</td>
</tr>
<tr>
<td>• Clearly defined stages.</td>
<td>• Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.</td>
</tr>
<tr>
<td>• Well understood milestones.</td>
<td>• It is difficult to measure progress within stages.</td>
</tr>
<tr>
<td>• Easy to arrange tasks.</td>
<td>• Cannot accommodate changing requirements.</td>
</tr>
<tr>
<td>• Process and results are well documented.</td>
<td>• Adjusting scope during the life cycle can end a project.</td>
</tr>
<tr>
<td></td>
<td>• Integration is done as a &quot;big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.</td>
</tr>
</tbody>
</table>

In conclusion, the main advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.
1.2 Agile Methodologies

**What Is Agile**

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations (http://www.agilenutshell.com, 2017).

![Figure 2: Agile Methodologies](image)

Agile Methodologies are models used in the system development arena. The agile methodology has evolved in the mid-1990s as a part of reaction against traditional waterfall methods. That the Agile Methodologies were originating resulted from the use of the waterfall model were inflexible, slow, and inconsistent with the ways that software developers perform effective work. Agile development methods mark a return to development practice from early in the history of software development.

The use of the word agile in this context derives from the agile manifesto (http://leadinganswers.typepad.com/, 2017).
History: The Agile Manifesto

Figure 3: The Agile Manifesto

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met and what emerged was the Agile ‘Software Development’ Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, software development processes convened. This group of independent thinkers about software development, and sometimes competitors to each other, agreed on the Manifesto for Agile Software Development. At the core, Agile Methodologists are about delivering good products to customers by operating in an environment that does more than talk about "people as our most important asset" but “acts” as if people were the most important, and lose the word "asset".

The meeting at Snowbird was incubated at an earlier get together of Extreme Programming proponents, and a few "outsiders," organized by Kent Beck at the Rogue River Lodge in Oregon in the spring of 2000. At the Rogue River meeting attendees voiced support
for a variety of "Light" methodologies, but nothing formal occurred. During 2000, several articles were written that referenced the category of "Light" or "Lightweight" processes. A number these articles referred to "Light methodologies, such as Extreme Programming, Adaptive Software Development, Crystal, and SCRUM". In conversations, no one really liked the moniker "Light", but it seemed to stick for the time being. In September 2000, Bob Martin from Object Mentor in Chicago, started the next meeting ball rolling with an email; "I'd like to convene a small (two day) conference in the January to February 2001 timeframe here in Chicago. The purpose of this conference is to get all the lightweight method leaders in one room. All of you are invited; and I'd be interested to know who else I should approach." Bob set up a Wiki site and the discussions raged. Early on, Alistair Cockburn weighed in with an epistle that identified the general disgruntlement with the word ‘Light’: "I don't mind the methodology being called light in weight, but I'm not sure I want to be referred to as a lightweight attending a lightweight methodologists meeting. It somehow sounds like a bunch of skinny, feeble-minded lightweight people trying to remember what day it is." The fiercest debate was over location! There was serious concern about Chicago in wintertime—cold and nothing fun to do; Snowbird, Utah—cold, but fun things to do, at least for those who ski on their heads like Martin Fowler tried on day one; and Anguilla in the Caribbean—warm and fun, but time consuming to get to. In the end, Snowbird and skiing won out; however, a few people—like Ron Jeffries—want a warmer place next time.

We hope that our work together as the Agile Alliance helps others in our profession to think about software development, methodologies, and organizations, in new—more agile—ways. If so, we’ve accomplished our goals.” (http://agilemanifesto.org/principles.html)

The Core Principles of Agile
Without standards of excellence for agile, anyone can call anything agile. Over the years several myths have formed around Agile delivery. Here are some of the more popular ones.

Agile is a silver bullet:
You can fail just as spectacularly on an Agile project as you can be using any other traditional method. The difference to that is that You’ll fail faster using Agile (due to the transparency and visibility it brings) but unfortunately, it’s not a silver bullet or an excuse to stop thinking.

There's nothing inherently magical about Agile. It basically says
Bring your development team and customer as close together as you can, give them what they need, and then get out of the way.

**Agile is anti-documentation:**
Where Agile pushes back on documentation is as a means of communication. Agile prefers face-to-face communication over relying on the written word. But isn’t anti-documentation. Documentation gets treated like any other deliverable on an Agile project. It gets estimated, sized, and prioritized like any other user story.

**Agile is anti-planning:**
Agile is just anti-static planning. Users of Agile expect their plans to change and use tools like burndown charts to track and make these changes visible.
There’s a lot of planning that goes on in Agile projects like daily planning, iteration planning meetings, sprint planning meetings, release planning.

**Agile is undisciplined:**
Agile is a very disciplined way of delivering software which requires a lot of hard work, courage, and discipline. With Agile you are obliged to:

I. Test

II. Get feedback.

III. Ship software regularly.

IV. Be ready to change and update the plan.

V. Deliver bad news early.

**Agile requires a lot of rework:**
When it comes to rework you’ve got the rework of requirements (customers discovering what they really want) and the rework of the software (development teams discover better ways to design the software).
Both need to be balanced and tempered. Agile deals with this tension by empowering both sides with the power to iterate, so long as they work within the project’s means. Burndown charts play in big role in tracking how Agile project are doing. Just as tools like the Agile Inception Deck make sure everyone is on the same page with regards to time and money (http://agilemanifesto.org/principles.html).
Agile is anti-architecture:

Agile created terms like YAGNI (You Ain't Gonna Need It) to remind teams to keep things simple until proven otherwise.

That doesn’t mean Agile teams stop thinking, or don’t leverage previous experiences. It’s mostly building the mind-set that the best way to build systems is to keep things simple, and only add the complexity when you need it.

![Don’t build this ... if all you need is this.](image)

Figure 4: Agile like YAGNI

Source: (Kourounakis, 2014)

Agile doesn't scale:

There is no easy way to magically coordinate, communicate, and keep large groups of people all moving in the same direction towards the same cause. It’s hard work.

The one thing Agile does, is instead of looking for ways to scale up your project, look for ways to scale things down. In other words, if we know we are good at delivering with small, nimble, agile teams of ten, why don’t we structure our work that way.
Main Agile principles based on the Agile Manifesto

![Figure 5: Principles behind the Agile Manifesto](image)

### Agile Model Pros & Con

#### Pros:

- **Transparency:** The customer has frequent and early opportunities to see the work being delivered, and to make decisions and changes throughout the development project (from prioritizing features to iteration planning and review sessions to frequent software builds containing new features.).

- **Stakeholder Engagement:** The customer gains a strong sense of ownership by working extensively and directly with the project team throughout the project. By involving the client in every step of the project, there is a high degree of collaboration between the client and project team, providing more opportunities for the team to truly understand the client’s vision. Delivering working software early and frequently increases stakeholders’ trust in the
team’s ability to deliver high-quality working software and encourages them to be more deeply engaged in the project

☑ Early and Predictable Delivery: If time to market for a specific application is a greater concern than releasing a full feature set at initial launch, Agile can more quickly produce a basic version of working software which can be built upon in successive iterations.

☑ Improved Quality: By breaking down the project into manageable units, the project team can focus on high-quality development, testing, and collaboration. Also, by producing frequent builds and conducting testing and reviews during each iteration, quality is improved by finding and fixing defects quickly and identifying expectation mismatches early.

☑ Development is often more user-focused: Agile commonly uses user stories with business-focused acceptance criteria to define product features. By focusing features on the needs of real users, each feature incrementally delivers value, not just an IT component. This also provides the opportunity to beta test software after each Sprint, gaining valuable feedback early in the project and providing the ability to make changes as needed.

☑ Predictable Costs: Because each Sprint is a fixed duration, the cost is predictable and limited to the amount of work that can be performed by the team in the fixed-schedule time box.

Cons:

 ✓ Transparency: clients to understand that they are seeing a work in progress in exchange for this added benefit of transparency.

 ✓ Agile works best when members of the development team are completely dedicated to the project.

 ✓ Because Agile focuses on time-boxed delivery and frequent reprioritization, it’s possible that some items set for delivery will not be completed within the allotted timeframe. Additional sprints (beyond those initially planned) may be needed, adding to the project cost. In addition, customer involvement often
leads to additional features requested throughout the project. Again, this can add to the overall time and cost of the implementation.

✓ The close working relationships in an Agile project are easiest to manage when the team members are in the same physical space, which is not always possible. However, there are a variety of ways to handle this issue, such as webcams, collaboration tools, etc.

✓ The iterative nature of Agile development may lead to a frequent refactoring if the full scope of the system is not considered in the initial architecture and design. Without this refactoring, the system can suffer from a reduction in overall quality. This becomes more pronounced in larger-scale implementations, or with systems that include a high level of integration.

Agile is a powerful tool for software development, not only providing benefits to the development team, but also providing a number of important business benefits to the client. Agile helps project teams deal with many of the most common project pitfalls (such as cost, schedule predictability and scope creep) in a more controlled manner. By reorganizing and re-envisioning the activities involved in custom software development, Agile achieves those same objectives in a leaner and more business-focused way.

In conclusion: What is being Agile?

1. Responding to change
2. Early feedback
3. Continuous improvement with inspection and adaptation
4. Reduced risk
5. Shorter development cycles and more frequent releases
6. Visible and transparent
Figure 6: Software Development Estimation: Minimum Viable Product

Source: (Makarov, 2016)

Waterfall versus Agile

Figure 7: Waterfall versus Agile

Source: (Lotz, 2013)
Chapter 2: Most commonly used Agile Methods & Practices

2.1 Introduction

There are several agile development methods widely used today. Sometimes, people refer to agile software methods as agile techniques. Some of the most popular agile techniques are: Scrum, Crystal, Adaptive Software Development, Lean Software Development, Extreme Programming.

The Agile Methodologies differ in the approaches of development and management they propose. [http://martinfowler.com/articles/newMethodology.html](http://martinfowler.com/articles/newMethodology.html)

Some agile methods focus more extensively on project management and collaboration practices such as Scrum, Adaptive Software Development (ASD), and Lean Development. However, some agile methods concentrate heavily on software implementation practices such as Extreme Programming (XP), Agile Modelling (AM), and Feature-driven Development (FDD). At this thesis, we will focus only on Scrum and we will only give a short definition for the rest of methodologies.

2.2 Agile Methodologies

2.2.1 Adaptive software development (ASD)

Adaptive software development (ASD) is a software development embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs. Adaptive software development replaces the traditional waterfall cycle with a
repeating series of *speculate, collaborate*, and *learn* cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time boxed, risk driven, and change tolerant.

The word *speculate* refers to the paradox of planning — it is more likely to assume that all stakeholders are comparably wrong for certain aspects of the project’s mission, while trying to define it. During speculation, the project is initiated and adaptive cycle planning is conducted. Adaptive cycle planning uses project initiation information—the customer’s mission statement, project constraints (e.g., delivery dates or user descriptions), and basic requirements—to define the set of release cycles (software increments) that will be required for the project. *Collaboration* refers to the efforts for balancing the work based on predictable parts of the environment (planning and guiding them) and adapting to the uncertain surrounding mix of changes caused by various factors, such as technology, requirements, stakeholders, software vendors. The *learning* cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During these iterations the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience mastery in the problem domain and eventually (Highsmith J., 2000; Highsmith & Addison-Wesley, 2004)

- "Messy, Exciting, and Anxiety-Ridden: Adaptive Software Development".
- "Adaptive SD".
2.2.2 Agile modelling

Agile modelling (AM) is a methodology for modelling and documenting software systems based on best practices. It is a collection of values and principles, that can be applied on an (agile) software development project. This methodology is more flexible than traditional modelling methods, making it a better fit in a fast changing environment (State of Agile Development Survey Results, 2011). It is part of the agile software development toolkit.

Agile modelling is a supplement to other agile development methodologies such as Scrum, extreme programming (XP), and Rational Unified Process (RUP). It is explicitly included as part of the disciplined agile delivery (DAD) framework. As per 2011 stats, agile modelling accounted for 1% of all agile software development.[2]

- Agile modelling (AM) home page, effective practices for modelling and documentation

![Agile Modeling Lifecycle](image)

*Figure 9: Agile Modeling Lifecycle*

*Source: (Nierstrasz, 2013)*

2.2.3 Crystal Clear methods:

Crystal Clear is a member of the Crystal family of methodologies and is considered an example of an agile or lightweight methodology. (Cockburn)

Crystal Clear can be applied to teams of up to 6 or 8 co-located developers working on systems that are not life-critical. The Crystal family of methodologies focus on efficiency and habitability as components of project safety. Crystal Clear focuses on people, not processes or artifacts.
Crystal Clear requires the following properties:

3. Frequent delivery of usable code to users
4. Reflective improvement
5. Osmotic communication preferably by being co-located

Crystal Clear additionally includes these optional properties:

1. Personal safety
2. Focus
3. Easy access to expert users
4. Automated tests, configuration management, and frequent integration

2.2.4 Dynamic systems development method (DSDM):

Dynamic systems development method (DSDM) is an agile project delivery framework, primarily used as a software development method. (Richards, 2007) First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method (Abrahamsson, 2003). In 2007 DSDM became a generic approach to project management and solution delivery. DSDM is an iterative and incremental approach that embraces principles of Agile development, including continuous user/customer involvement.

![DSDM Lifecycle](image)

*Figure 10: Dsdm lifecycle*
DSDM fixes cost, quality and time at the outset and uses the MoSCoW\textsuperscript{2} prioritisation of scope into musts, shoulds, coulds and won't haves to adjust the project deliverable to meet the stated time constraint. DSDM is one of several Agile methods for developing software and non-IT solutions, and it forms a part of the Agile Alliance.

2.2.5 Extreme programming (XP):

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, (Human Centred Technology Workshop 2005) it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted. (Design Patterns and Refactoring). Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers (Agile Alliance, 2001).

![Extreme Programming (XP) at a Glance](image)

\textit{Figure 11: Extreme Programming (XP) at a Glance}

Source: (Meier, 2014)

\textsuperscript{2} The MoSCoW method is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement; it is also known as MoSCoW prioritization or MoSCoW analysis.
The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed continuously, i.e. the practice of pair programming.

2.2.6 Feature-driven development (FDD):

**Feature-driven development (FDD)** is an iterative and incremental software development process. It is one of many lightweight or Agile methods for developing software. FDD blends industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.

![Feature-driven development (FDD)](image)

*Figure 12: Feature-driven development (FDD)*

Source: (Gayal & Ruhaim, 2014)

2.2.7 Lean software development:

**Lean software development (LSD)** is a translation of lean manufacturing and lean IT principles and practices to the software development domain. Adapted from the Toyota Production System, a pro-lean subculture is emerging from within the Agile community (Kanban, 2011).
2.2.8 Kanban

Kanban (看板?) (literally signboard or billboard in Japanese) is a scheduling system for lean manufacturing and just-in-time manufacturing (JIT) (Ohno, 1998). Kanban is an inventory-control system to control the supply chain. Taiichi Ohno, an industrial engineer at Toyota, developed Kanban to improve manufacturing efficiency. Kanban is one method to achieve JIT (Shingō, 1989).

Kanban became an effective tool to support running a production system and an excellent way to promote improvement. One of the main benefits of Kanban is to establish an upper limit to the work in process inventory, avoiding overloading of the manufacturing system. Other systems with similar effect are for example CONWIP (Hopp, 2004). A systematic study of various configurations of Kanban systems, of which CONWIP is an important special case, can be found in Tayur (1993), among other papers (Tayur, 1993; Muckstadt & Tayur, 1995; Muckstadt & Tayur, 1995; Tayur, 1992; Schonberger, 2001).
An English-language term that captures the meaning of the Japanese word, Kanban, is *queue limiter*; and the beneficial result is *queue limitation* (Waldner, 1992). Operationally, then, as process problems are dealt with, the queue limit (or maximum) should be reduced; for example, a former upper limit of five pieces is reduced to four, with queue time in the process reduced by 20 percent.

### 2.2.9 Scrum

Scrum is an iterative and incremental *agile software development* framework for managing product development (Verheyen, 2016). It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved (Takeuchi & Nonaka, 1986).

A key principle of Scrum is its recognition that during *product development*, the customers can change their minds about what they want and need (often called *requirements volatility*) (Henry & Henry, 1993), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, Scrum adopts an evidence-based *empirical approach*—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team’s ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in

2.2.10 Scrumban
Scrumban is an Agile management methodology describing hybrids of Scrum and Kanban and was originally designed to transition from Scrum to Kanban. Today, Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work (Pugh, 2011; Adzic, 2009).

2.2.11 Rapid application development (RAD)
Rapid application development (RAD) is both a general term used to refer to alternatives to the conventional waterfall model of software development as well as the name for James Martin's approach to rapid development. In general, RAD approaches to software development put less emphasis on planning and more emphasis on process.

In contrast to the waterfall model, which calls for rigorously defined specification to be established prior to entering the development phase, RAD approaches emphasize adaptability and the necessity of adjusting requirements in response to knowledge gained as the project progresses. Prototypes are often used in addition to or sometimes even in place of design specifications.
RAD is especially well suited (although not limited to) developing software that is driven by user interface requirements. Graphical user interface builders are often called rapid application development tools.

![Prototype Cycles Diagram](www.wavemaker.com)

**Figure 16: Prototype Cycles**

Source: (www.wavemaker.com)

2.3 Agile practices

Agile development is supported by several concrete practices as well, covering areas like requirements, design, modelling, coding, testing, planning, risk management, process, quality, etc. Some notable agile practices include:

2.3.1 Acceptance test-driven development (ATDD)

**Acceptance test–driven development** (ATDD) is a development methodology based on communication between the business customers, the developers, and the testers (Chelimsky, και συν., 2001). ATDD encompasses many of the same practices as specification by example (Adzic, How successful teams deliver the right software. Manning., 2011), behaviour-driven development (BDD), example-driven development (EDD), (Example Driven Design". , 2013) and support-driven development also called story test–driven development (SDD) ("Story Test-Driven Development", 2013). All these processes aid developers and testers in understanding the customer’s needs prior to implementation and allow customers to be able to converse in their own domain language. ATDD is closely related to test-driven development (TDD). It differs by the emphasis on developer-tester-business customer collaboration. ATDD encompasses acceptance testing, but highlights writing acceptance tests before developers begin coding (Beck K., 2002).

2.3.2 Behaviour-driven development (BDD)

**Behaviour-driven development (BDD)** is a software development process that emerged from test-driven development (TDD) (Behaviour-Driven Development, 2012) (Haring & de
Behaviour-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development (Solis & Wang, 2011).

Although BDD is principally an idea about how software development should be managed by both business interests and technical insight, the practice of BDD does assume the use of specialized software tools to support the development process (Bellware, 2008). Although these tools are often developed specifically for use in BDD projects, they be specialized forms of the tooling that supports test-driven development. The tools serve to add automation to the ubiquitous language that is a central theme of BDD.

BDD is largely facilitated using a simple domain-specific language (DSL) using natural language constructs (e.g., English-like sentences) that can express the behaviour and the expected outcomes. Test scripts have long been a popular application of DSLs with varying degrees of sophistication. BDD is considered an effective technical practice especially when the "problem space" of the business problem to solve is complex (Tharayil, 2016).

2.3.3 Business analyst designer method (BADM)

Business analysis is a research discipline of identifying business needs and determining solutions to business problems. Solutions often include a software-systems development component, but may also consist of process improvement, organizational change or strategic planning and policy development. The person who carries out this task is called a business analyst or BA (Kathleen, Vander, & Kimi, 2008).

Business analysts do not work solely on developing software systems. Those who attempt to do so run the risk of developing an incomplete solution ("Business Analysis Body of Knowledge v2.0", 2008).

Although there are different role definitions, depending upon the organization, there does seem to be an area of common ground where most business analysts work. The responsibilities appear to be:

3. To investigate business systems, taking a holistic view of the situation. This may include examining elements of the organisation structures and staff development issues as well as current processes and IT systems.
4. To evaluate actions to improve the operation of a business system. Again, this may require an examination of organisational structure and staff development needs, to ensure that they are in line with any proposed process redesign and IT system development.

5. To document the business requirements for the IT system support using appropriate documentation standards.

In line with this, the core business analyst role could be defined as an internal consultancy role that has the responsibility for investigating business situations, identifying and evaluating options for improving business systems, defining requirements and ensuring the effective use of information systems in meeting the needs of the business (Alternative definition of Business Analysis from BCS ISEB, 2008).

2.3.4 Cross-functional team

A cross-functional team is a group of people with different functional expertise working toward a common goal (Krajewski & Ritzman, 2005). It may include people from finance, marketing, operations, and human resources departments. Typically, it includes employees from all levels of an organization. Members may also come from outside an organization (in particular: from suppliers, key customers, or consultants). Cross-functional teams often function as self-directed teams assigned to a specific task which calls for the input and expertise of numerous departments. Assigning a task to a team composed of multi-disciplinary individuals increases the level of creativity. Each member offers an alternative perspective to the problem and potential solution to the task. In business today, innovation is a leading competitive advantage and cross-functional teams promote innovation through a creative collaboration process. Members of a cross-functional team must be well versed in multi-tasking as they are simultaneously responsible for their cross-functional team duties as well as their normal day-to-day work tasks.

Some researchers have viewed cross-functional interactions as cooperative or competitive in nature, while others have argued that organization’s functional areas are often forced to compete and cooperate simultaneously with one another (“coopetition”) and it is critical to understand how these complex relationships interplay and affect firm performance (Luo, Slotegraaf, & Xing, 2006).

Decision making within a team may depend on consensus, but often is led by a manager/coach/team leader. Leadership can be a significant challenge with cross-functional
teams. Leaders are charged with the task of directing team members of various disciplines. They must transform different variations of input into one cohesive final output. Cross-functional teams can be likened to the board of directors of a company. A group of qualified individuals of various backgrounds and disciplines are assembled to collaborate in an efficient manner to improve the organization or solve a problem.

Some organizations are built around cross-functional workflows by having reporting lines to multiple managers. This type of management is called matrix management, and such organizations are often called matrix organizations.

2.3.5 Continuous integration (CI):
In software engineering, continuous integration (CI) is the practice of merging all developer working copies to a shared mainline several times a day. Grady Booch first named and proposed CI in his 1991 method, although he did not advocate integrating several times a day (Booch, 1991).

2.3.6 Domain-driven design (DDD)

Domain-driven design (DDD) is an approach to software development for complex needs by connecting the implementation to an evolving model.[1] The premise of domain-driven design is the following:

i. placing the project's primary focus on the core domain and domain logic;
ii. basing complex designs on a model of the domain;
iii. initiating a creative collaboration between technical and domain experts to iteratively refine a conceptual model that addresses domain problems (http://dddcommunity.org, 2017).

2.3.7 Pair programming

Pair programming is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently (Ramsey, Bourque, & Dupuis, 2001).

While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This frees the driver to focus all his or her attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.
2.3.8 Planning poker

Planning poker, also called Scrum poker, is a consensus-based, gamified technique for estimating, mostly used to estimate effort or relative size of development goals in software development. In planning poker, members of the group make estimates by playing numbered cards face-down to the table, instead of speaking them aloud. The cards are revealed, and the estimates are then discussed. By hiding the figures in this way, the group can avoid the cognitive bias of anchoring, where the first number spoken aloud sets a precedent for subsequent estimates (Cohn, "Agile Estimating and Planning". Mountain Goat Software., 2005).

It is most commonly used in agile software development, in Scrum and Extreme Programming.

The method was first defined and named by James Grenning in 2002 and later popularized by Mike Cohn in the book Agile Estimating and Planning, whose company trade marked the term ("Planning poker - Trademark, , 2008) and a digital online tool (Cohn, 2016).

2.3.9 Refactoring

Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behaviour. Refactoring improves non-functional attributes of the software. Advantages include improved code readability and reduced complexity. These can improve source-code maintainability and create a more expressive internal architecture or object model to improve extensibility. Typically, refactoring applies a series of standardised basic micro-refactoring’s, each of which is (usually) a tiny change in a computer program's source code that either preserves the behaviour of the software, or at least does not modify its conformance to functional requirements. Many development environments provide automated support for performing the mechanical aspects of these basic refactoring’s. If done extremely well, code refactoring may also resolve hidden, dormant, or undiscovered computer bugs or vulnerabilities in the system by simplifying the underlying logic and eliminating unnecessary levels of complexity. If done poorly it may fail the requirement that external functionality not be changed, introduce new bugs, or both.

By continuously improving the design of code, we make it easier and easier to work with. This is in sharp contrast to what typically happens: little refactoring and a great deal of attention paid to expediently adding new features. If you get into the hygienic habit of
refactoring continuously, you'll find that it is easier to extend and maintain code (Kerievsky, 2004).

### 2.3.10 Test-driven development (TDD)

**Test-driven development (TDD)** is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.

Kent Beck, who is credited with having developed or 'rediscovered' (Kent, 2012) the technique, stated in 2003 that TDD encourages simple designs and inspires confidence (Beck, 2003).

Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999 (Lee, 2001), but more recently has created more general interest in its own right (Newkirk & Vorontsov, 2004).

Programmers also apply the concept to improving and debugging legacy code developed with older techniques (Feathers, 2004).

### 2.3.11 User story

In software development and product management, a **user story** is a description consisting of one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function. User stories are used with agile software development methodologies as the basis for defining the functions a business system must provide, and to facilitate requirements management. It captures the "who", "what" and "why" of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a small paper notecard.

A user story encapsulates the action of one function making it possible for software developers to create a vertical slice of their work.

### 2.3.12 Story-driven modelling

Story-driven modelling is an **Object-oriented modeling technique**. Other forms of Object-oriented modeling focus on class diagrams (Diethelm, Geiger, & Zündorf, 2004). Class diagrams describe the static structure of a program, i.e. the building blocks of a program and how they relate to each other (Van Gorp, 2008; Eickhoff, Geiger, Hahn, & Zündorf,
2012). Class diagrams also model data structures, but with an emphasis on rather abstract concepts like types and type features (Norbisrath & Ulrich, 2013) (Zündorf, Schürr, & Winter, 1999; Ryser & Glinz, 2000).

2.3.13 Retrospective

An Agile retrospective is a meeting that’s held at the end of an iteration in Agile software development (ASD). During the retrospective, the team reflects on what happened in the iteration and identifies actions for improvement going forward. Each member of the team answers the following questions:

- What worked well for us?
- What did not work well for us?
- What actions can we take to improve our process going forward?

The Agile retrospective can be thought of as a "lessons learned" meeting. The team reflects on how everything went and then decides what changes they want to make in the next iteration. The retrospective is team-driven, and team members should decide together how the meetings will be run and how decisions will be made about improvements.

2.3.14 Velocity tracking

**Velocity** is a capacity planning tool sometimes used in agile software development. Velocity tracking is the act of measuring said velocity. The velocity is calculated by counting the number of units of work completed in a certain interval, the length of which is determined at the start of the project ("Velocity:Measuring and Planning an Agile Project", 2010).

2.3.15 User Story Mapping

Story mapping consists of ordering user stories along two independent dimensions. The "map" arranges user activities along the horizontal axis in rough order of priority (or "the order in which you would describe activities to explain the behaviour of the system"). Down the vertical axis, it represents increasing sophistication of the implementation.

Given a story map so arranged, the first horizontal row represents a "walking skeleton", a barebones but usable version of the product. Working through successive rows fleshes out the product with additional functionality.

2.3.16 More Practices:

- **Requirements**
  - Product Vision
- Product Backlog
- Use Cases
- Personas

**Design**
- Emergent Design / Evolutionary Design
- Design by Contract
- System Metaphor

**Construction**
- Coding Style / Coding Guidelines / Coding Standard
- Collective Code Ownership
- Daily Builds / Automated Builds
- Software Metrics / Code Metrics & Analysis
- Source Control / Version Control
- Issue Tracking / Bug Tracking
- Frequent Delivery / Frequent Releases

**Testing - Unit Testing**
- Smoke Testing / Build Verification Test
- Integration Testing
- System Testing
- Test Automation
- Acceptance Criteria / Acceptance Testing

**Process**
- Timeboxing / Fixed Sprints / Fixed Iteration Length
- Sprint Backlog
- Task Board
- Definition of Done
- Daily Stand-up Meeting / Daily Scrum
Sprint Review / Iteration Demo
Root Cause Analysis / 5 Whys
Burn Down Charts
Big Visible Charts

Organization
Small Team
Self-Organizing Team
Sitting Together / Common Workspace
Sustainable Pace
Move People Around

The Agile Alliance has provided a comprehensive online guide to applying agile these and other practices (http://agilemanifesto.org/principles.html).

Chapter 3 – Scrum

Scrum is an Agile framework for completing complex projects. Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework is deceptively simple.
3.1 Roles

3.1.1 Product Owner

The Scrum product owner is typically a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product. The product owner is commonly a lead user of the system or someone from marketing, product management or anyone with a solid understanding of users, the market place, the competition and of future trends for the domain or type of system being developed.

This, of course, varies tremendously based on whether the team is developing commercial software, software for internal use, hardware or some other type of product. The key is that the person in the product owner role needs to have a vision for what is to be built. Although the agile PO prioritizes the product backlog during the sprint planning
meeting, the team selects the amount of work they believe they can do during each sprint, and how many sprints will be required.

The product owner does not get to say, "We have four sprints left, therefore you must do one-fourth of the product backlog this sprint." The Scrum product owner's job is to motivate the team with a clear, elevating goal. Team members know best what they are capable of, and so they select which user stories from the top of the product backlog they can commit to delivering during any sprint. In return for the Scrum team's commitment to completing the selected user stories from the top of the product backlog, the product owner makes a reciprocal commitment to not throw new requirements at the team during the sprint. Requirements can change (and change is encouraged) but only outside the sprint. Once the team starts on a sprint, it remains focused on the goal of that sprint.

The product owner role requires an individual with certain skills and traits, including **availability, business savvy and communication skills**.

i. First, the Scrum product owner needs to be available to his or her team. The best product owners show commitment by doing whatever is necessary to build the best product possible – and that means being actively engaged with their teams.

ii. Business savvy is important for the agile product owner because he or she is the decision maker regarding what features the product will have. That means, the agile PO should understand the market, the customer and the business to make sound decisions.

iii. Finally, communication is a large part of the product owner responsibilities. The product owner role requires working closely with key stakeholders throughout the organization and beyond, so he or she must be able to communicate different messages to different people about the project at any given time.

### 3.1.2 Scrum Master

A scrum master is the facilitator for an agile development team.
In product development, team members huddle together each morning for a stand-up meeting where they review progress and essentially restart the project. During the daily meetings, which are sometimes called "scrum," the scrum master asks the team members these three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

Although the title of scrum master sounds powerful, the scrum master is not the project leader and is not held accountable for outcomes. The team as a whole is responsible for outcomes. The scrum master is responsible for:

i. Helping the team to reach consensus for what can be achieved during a specific period.
ii. Helping the team to reach consensus during the daily scrum.
iii. Helping the team to stay focused and follow the agreed-upon rules for daily scrums.
iv. Removing obstacles that are impeding the team's progress.
v. Protecting the team from outside distractions.

3.1.3 Scrum Team

Responsible for the project’s creation and delivery. The team members will normally be comprised of developers, QA, marketers, designers and wherever is needed to deliver a quality project. They are responsible for planning, design, development, testing, and project delivery.
3.1.4 Stakeholders
Represents a broad category of people who can be users, managers of users, operations, support, Portfolio Managers, other Agile teams with dependencies, executive team, investors, and more.

3.2 Artifacts
3.2.1 Product Backlog
In the simplest definition, the Scrum Product Backlog is simply a list of all things that needs to be done within the project. It replaces the traditional requirements specification artifacts. These items can have a technical nature or can be user-centric (for instance: in the form of user stories). The owner of the Scrum Product Backlog is the Scrum Product Owner. The Scrum Master, the Scrum Team and other Stakeholders contribute it to have a broad and complete To-Do list.

Working with a Scrum Product Backlog does not mean that the Scrum Team is not allowed to create and use other artifacts. Examples for additional artifacts could be a summary of the various user roles, workflow descriptions, user interface guidelines, storyboards, or user interface prototypes. However, these artifacts do not replace the Scrum Product Backlog but complement and detail its content.

The Scrum Product Owner uses the Scrum Product Backlog during the Sprint Planning Meeting to describe the top entries to the team. The Scrum Team then determines which items they can complete during the coming sprint.

Each Scrum Product Backlog has certain properties that differentiate it from a simple to-do list:

- An entry in the Scrum Product Backlog always add value for the customer
- The entries in the Scrum Product Backlog are prioritized and ordered accordingly
- The level of detail depends on the position of the entry within the Scrum Product Backlog
- All entries are estimated
✓ The Scrum Product Backlog is a living document
✓ There are no action-items or low-level tasks in the Scrum Product Backlog

i. **Only entries that add value**
   Each entry in the Scrum Product Backlog must have customer value. Entries without any customer value are pure waste and should not be present anyway. The Scrum Product Backlog can include entries for the exploration of customer needs or various technical options, a description of both functional and non-functional requirements, the work necessary to launch the product, and other items as well, such as setting up the environment or remediating defects. Some tasks may not add direct value to the functionality. Nevertheless, they might add value by increasing quality or reducing incidents in the long term.

ii. **Living document**
   The Scrum Product Backlog is changed throughout the whole project. If needed, new requirements are added and existing requirements may be modified, defined in more detail or even deleted. Requirements are no longer frozen early on. Instead the final set of requirements within the Scrum Product Backlog is also developed iteratively, together with the resulting software. This is different to traditional requirements engineering but allows maximizing customer value and minimizes development effort.

iii. **Different level of details**
   The requirements in the Scrum Product Backlog have a different level of details. Only those requirements that shall be implemented during one of the next sprints are defined in greater detail. The simple reason for this is that it does not make sense to invest time and effort into the specification of these requirements, as most of these requirements will have changed anyway until implementation starts.

iv. **No low-level tasks**
   The Scrum Product Backlog shall not contain the detailed requirement information. Ideally the final requirements are defined together with the customer during the sprint. Breakdown and distribution of these requirements is the responsibility of the Scrum Team.
v. **The Scrum Product Backlog is ordered**

All entries are prioritized and the Scrum Product Backlog is ordered. The Scrum Product Owner with the help of the Scrum Team does the prioritization. Added Value, Costs and Risks are the most common factors for prioritization. With this prioritization, the Scrum Product Owner decides what should be done next.

vi. **All entries are estimated**

All the entries within the Scrum Product Backlog must be estimated according to the agreed definition (e.g. story points). This estimation can then be used to prioritize entries in the Scrum Product Backlog and to plan releases.

vii. **Working with the Backlog**

The backlog needs regular attention and care - it needs to be managed carefully. At the start of the project the Scrum Team and its Scrum Product Owner start by writing down everything they can think of easily. This is almost always more than enough for a first sprint.

After this initial setup, the Scrum Product Backlog has to be maintained in an ongoing process that comprises the following steps:

1. As new items are discovered they are described and added to the list. Existing ones are changed or removed as appropriate.
2. Ordering the Scrum Product Backlog. The most important items are moved to the top.
3. Preparing the high-priority entries for the next Sprint Planning Meeting
4. (Re-)Estimating the entries in the Scrum Product Backlog.

The Scrum Product Owner is responsible for making sure that the Scrum Product Backlog is in good shape this is a collaborative process. When using the Scrum Framework about 10% of the Scrum Teams total time should be reserved for maintaining the Scrum Product Backlog (discussion, estimation etc.) [http://www.scrum-institute.org/](http://www.scrum-institute.org/).
3.2.2 Sprint Backlog

The Sprint Backlog is an ordered list of Product Backlog Items, preferably User Stories, that the Team believes it can complete during the coming Sprint. These items are pulled from the top of the Product Backlog during the Sprint Planning Meeting.

Each story should have a Point value assigned to it based on the Estimated amount of relative effort it will take to complete the story. **It is important that the Team estimates in points and not hours.** The Team determines how best to work through the Sprint Backlog. However, when possible, they should work on the highest value items first.

Once the Team forecasts the number of stories they feel they can accomplish in the Sprint Backlog, there should be no additions or changes until the Sprint ends. However, if during the Sprint management or the Product Owner decide there is a feature of higher business value that needs to come into the Sprint, the Product Owner should use the interruption procedure.

If an interruption arises that so dramatically changes the priorities or scope of the Sprint and cannot be dealt with as an interruption, the Product Owner may abort the Sprint. In this case the Team stops, a new Sprint Planning meeting is held and a new Sprint is started.
can be extremely disruptive to the Team so the Product Owner should be very leery of stopping mid-Sprint (https://www.scruminc.com/sprint-backlog/).

3.2.3 Product increment
The increment (or potentially shippable increment, PSI) is the sum of all the product backlog items completed during a sprint, integrated with the work of all previous sprints. At the end of a sprint, the increment must be complete, according to the scrum team's definition of done (DoD), fully functioning, and in a usable condition regardless of whether the product owner decides to release it.

![Product Increment Diagram]

Figure 19: Product Increment
Source: (Sumiti, 2017)

3.3 Workflow
3.3.1 The Scrum process
A sprint (or iteration) is the basic unit of development in Scrum. The sprint is a timeboxed effort restricted to a specific duration. The duration is fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common. Each sprint starts with a sprint planning event and ends with a sprint review and sprint retrospective.

Scrum emphasizes working product at the end of the sprint that is really done. In the case of software, this likely includes that the software has been fully integrated, tested and documented, and is potentially shippable.

3.3.2 Sprint Planning
In Scrum, every iteration begins with a sprint planning meeting. At this meeting, the Product Owner and the team negotiate which stories a team will tackle that sprint. This meeting is a time-boxed conversation between the Product Owner and the team. It’s up to the Product Owner to decide which stories are of the highest priority to the release and which will
generate the highest business value, but the team has the power to push back and voice concerns or impediments. When the team agrees to tackle the work, the Product Owner adds the corresponding stories into the sprint backlog. At this point, the Product Owner may choose to leave while the team decomposes the forecasted backlog items into tasks. This meeting is sometimes called Sprint Planning Part 2. In Large Scale Scrum, multiple teams pull items from one Product Backlog. Multiple backlogs for one product, and multiple Product Owners lead to localize sub optimizations, longer work-in-progress queues, thus are harmful to agility.

3.3.3 The Daily Scrum

Every day, the Scrum team gathers in front of their task board to discuss the progress made yesterday, goals for today, and any impediments blocking their path.

• What have I done since the last Scrum meeting (yesterday)?
• What will I do before the next Scrum meeting (tomorrow)?
• What prevents me from performing my work as well as possible?

This meeting should not exceed 15 minutes. If members of the team need to discuss an issue that cannot be covered in that amount of time, we recommend they attend a sidebar meeting following the stand-up. This allows team members to attend meetings that directly involve their work, instead of sitting through irrelevant meetings. Unfortunately, daily Scrums often last longer than 15 minutes. To compensate, many teams use stop watches or timers to uphold the time limitations. Also, to limit distracting small talk, many teams employ a talking stick or mascot, which a team member must hold to speak in the meeting. Upon finishing an update, the talking stick is then passed to the next team member, who reports, and so on.

3.3.4 Sprint Review

When the sprint ends, it’s time for the team(s) to demonstrate a potentially shippable product increment to the Product Owner and other stakeholders. The Product Owner declares which items are truly done or not. Teams commonly discover that a story’s final touches often excise the most effort and time. Partially done work should not be called “done.”
This public demonstration replaces status meetings and reports, as those things do not aid transparency. Scrum emphasizes *empirical* observations such as working products.

### 3.3.5 Sprint Retrospective

After the sprint review meeting, the team and the Scrum Master get together in private for the retrospective meeting. During this meeting, the team inspects and adapts their process. When the Scrum Master and outer organization create an environment of psychological safety, team members can speak frankly about what occurred during the Sprint and how they felt about it. After all team members thoroughly understand each other, they work to identify what they’d like to do differently the next Sprint, typically focusing only on one or two specific areas of improvement each Sprint. The Scrum Master may also observe common impediments that impact the team and then work to resolve them (Schwaber & Sutherland, 2017).

*Figure 20: Albelli retrospective*
3.4 Work & Delivery Flow-Example

We work with:
- master branch
- acceptance branch
- testing branch

One branch per case

New case flow:

![Image of case flow]

*Figure 21: Case flow*

Step 1: Drag it to WIP & Create new branch from Latest Master Branch:

Naming the branch
- PATCH: Fix bugs
- REVISION: Improving existing features
- RELEASE: Adding new features
- MASTER RELEASE: Breaking changes

Number of sprint – PATCH/REV/ RELEASE - Case number

Example: S018-REV-CMS-5903

*TIP:* After creating a branch remember to publish it and at the end of the day to push all your commits from your local HEAD branch to the chosen remote branch, so in any case, always someone from the team can pick it up and continue working with it.

Step 2: Coding / Merge Latest Master branch to yours if needed

Step 3: Code Review: Check your code changes with another member of the team

Step 4: Functional Review – Check your changes with the person who issued the problem
Merge it to the latest branch of acceptance. By pushing the latest branch, you automatically trigger the building process of your code in Team City. Check if passes all the unit tests and afterwards deploy to acceptance using AWS opsworks.

- Set up a meeting so you can present & explain what you have done
- Otherwise if it’s a small change & easy to test it, ask them to check it and wait for feedback

**Step 5: Testing**
- Always make sure that you have clear explanation on how to set up section, so whoever picks this up has the all the information needed to test the case effectively.

**Step 6: Done**
- Always when merging with latest master branch update also the files CHANGEMLOG.md & ap-wp-config.php, so we can track all the code changes.

*Figure 22: Steps of a case flow*
Chapter 4: Methodology of research

This chapter focuses on the methodological design of research, where research methods are recorded. The exploratory method chosen in this research is also justified. Special reference is made to the structure, content and measurements of the study questionnaire. The selection of the specific sampling method is documented and the final sample of the survey is presented. Finally, the steps of the data collection process and the statistical analyzes carried out as part of this survey, are described in details.

4.1 Purpose of research - Definition of the research question

The purpose of this research is exploring when a team is ready to perform agile and which development methodology should the team use. The aim is to examine the factors that influence the choice of the agile method to be used by the team.

The value of these results is of great importance given the research gap in the international literature as it adds significant knowledge about the field of project organization research in companies.

4.2 Research plan

4.2.1 Selection of research plan

According to (Siomkos & Mavros, 2008), there are three different types of research, exploratory that aims to explore unknown or broad problems, descriptive research that aims to describe the characteristics of a population, and causal research that aims to find the relationship between cause and effect. Given the fact that in the present survey questionnaires were used to determine the relationships between some variables, we would say that the present research is a causal research.

4.2.2 Method of collecting data

In the present study both secondary and primary data were used. The collection of secondary data was based on published material. The data collected are particularly useful for the creation and development of the assumptions we made in our research (Stathakopoulos, 2005; Siomkos & Mavros, 2008).
Regarding to the collection of primary data, which according to Stathakopoulos (2005) is influenced by the degree of construction and the degree of masking, in the present study the primary data collection method chosen was to investigate the views of the workers on the usefulness of the method agile using questionnaires and is characterized by a high degree of structure (the questions and answers are completely predefined and standardized by the researcher) and is also non-disguised, that is to say it discloses its purpose study of the questions uses (Stathakopoulos, 2005).

4.2.3 Choice of measurement method

The measurement method carried out in this survey was the completion of a questionnaire consisting of 130 questions (items), of which the first 33 questions relate to the demographic characteristics of the sample. Demographics characteristics are related to: age group, total work experience, experience with agile, time working in your company, time working in your team, description of yourself from 1 to 5 scale, organization of yourself from 1 to 5 scale, creativity on a scale from 1-5, the ability of explaining something in detail on a scale from 1-5, estimation of the time needed for your daily tasks on a scale from 1-5, how easy is to be distracted on a scale from 1-5, the ability to be in a lead of a process on a scale from 1-5, the awareness of the team about the daily tasks on a scale from 1-5, the opinion of working big complex projects that can be broken down or not, the time needed in order to finish a unit of work: 1/2 - 2 days, the experience cases that might be blocked because of dependencies, the unexpected cases which pop up, the project failure or success in your team, the number of people including in the team, being part of a cross functional team, the immaturity of the team, the knowledge sharing in your team on scale 1-5, all team members follow the workflow of the team as expected or not, the capability of the members of the team picking up their daily tasks, the knowledge of what case other team members are working on, the satisfaction of the collaboration achieved among your team members, the disagreements in the team, the achievement of better results of the team if they would listen to your opinion more often, communication on ideas above team members, primary measure of progress in agile software development, the aim of agile software referring to improvement of productivity by eliminating unknowns and getting things right the first time, characteristics that are not included in the waterfall method of software development, the opinion of the foundation of agile software development being
a set of processes and tools that helps teams adhere to a well-defined plan as they make software.

The remaining questionnaire questions were closed-type and open-ended and corresponded essentially to 8 different scales (based mostly on scrum) developed by the researcher. The scales used are the following:

I. Team Member Using Scrum Now
II. Product Owner Using Scrum Now
III. Scrum Master Using Scrum Now
IV. Team Member that has used Scrum
V. Product Owner that has used Scrum
VI. Scrum Master that has used Scrum
VII. Team member that has used Agile in the past
VIII. Team member that plans to use Agile in the future

The respondents from the first three scales were asked to respond if they are using scrum at the moment with a yes or no (second-degree scale, 1: Yes, 2: No). Then they respond on what their role is in the team (open-typed: Scrum Master, Product Owner, Team member). If the respondent replied that is a team member then he/she asked to respond on questions which refer to Scrum Master & Product owner evaluation, team values and evaluating scrum. If the respondent is a Product Owner Using Scrum now then he/she had to respond on questions related to self-evaluation and evaluating scrum (satisfaction by using scrum, 3 most difficult side effects in Agile ext.). A Scrum Master Using Scrum Now had to respond on questions related to self-evaluation, team Values and Evaluating scrum.

The respondents from fourth, fifth, sixth scale (Team Member that has used Scrum, Product Owner that has used Scrum, Scrum Master that has used Scrum) were asked to respond if they are using scrum at the moment and if they have ever used scrum (second-degree scale, 1: Yes, 2: No). Then they respond on what their role is in the team (open-typed: Team member, Product Owner, Scrum Master). Team member that has used Scrum in the past had to explain what he/she thinks about Scrum Master & Product owner evaluation, team Values, evaluating scrum. Product Owner that has used Scrum in the past had to respond on questions related to self-evaluation and evaluating scrum. Scrum Master
that has used Scrum in the past answered in questions which are focusing on self-
evaluation, team values and evaluating Scrum.

As far as respondents, that are not using scrum at the moment and haven't ever
used scrum, are concerned, are divided into 2 categories, team member that has used agile
in the past (scale 7) and team member that plans to use Agile in the future (scale 8). Team
member that has used Agile in the past and he/she has used agile methodologies in the
past, answered in questions related on evaluating Agile. Team member that plans to use
Agile in the future and hasn't ever used any agile methodologies respond on questions
which are referring to the use of agile methods in the future.

4.3 Data collection- Timeline and place of research
Definition of population
With regard to the population in this survey, it could be defined as all employees in the
company Albelli based in Amsterdam, Netherlands. The given population of 59 items is
defined by 4 parameters: the element, the sampling unit, the extension, and time "in this
research were as follows (Stathakopoulos, 2005):

- Item: Workers aged 20 and over of both sexes
- Sampling Unit: Employees
- Extent: in the Netherlands
- Time: from September 2017 to November 2017

Specification of the sampling frame
Since the sample used was non-probability, no sampling frame was required (Siomkos &
Mavros, 2008).

Choice of sampling method
With regard to the sampling technique in the present study a non-probability sample was
used (the probability that a person is selected in the sample is unknown) and more
specifically, a convenience sample. Although for this method it is often mentioned that a
problem is the researcher's ignorance of whether the participants represent the population,
this problem is cured by the definition of the target population. The non-representation of
the population from the sample is not considered possible if the population is defined by
general and minimally restrictive criteria (Stathopoulos, 2005).
Specification of the size of sample

In order to obtain a sample of 59 people as it was an experimental approach, it was considered appropriate to distribute more questionnaires. In total, 70 questionnaires were distributed and 59 valid questionnaires were finally collected.

The survey was conducted by the researcher herself during the first three weeks of September. The questionnaire was shared only to people who wanted to respond. In cases of queries, the investigator made explanations mainly for the process of completing the questionnaire.

4.4 Implementation - analysis and data processing

Collecting the information, the researcher did not encounter particular problems with the respondents (refusal to participate etc.). Respondents were willing to complete the questionnaire which did not exceed over 15 minutes.

At this stage of the research effort there is an extensive use of statistics for the extraction of statistical conclusions. For this purpose, the SPSS 20 statistical packet was used to make the required statistical analyzes. The data was recorded in files that are recognized by that program. The answers to the questions were recorded in the form of a data matrix, which consists of lines, each corresponding to a questioner and columns corresponding to the questions (variables). Query responses were encoded with numeric symbols as variable values and entered into the data table with labels for both variables and their values. We also encoded the incomplete values, we set the values of the missing values of a variable (missing value: code 999) (Siomkos & Mavros, 2008).

The descriptive statistical measures of each variable / query with frequency tables are presented. Our frequency tables give the percentage (%) of the prices / answers of each question. In some cases, we also use other statistical measures (such as mean, median, standard deviation). A frequency table as derived from SPSS has the following columns: Initially, the first column gives the variable-class values, ie the possible answers to each query. Then the "Frequency" column gives us the number of people in each category. The "Percent" column gives the percentage (%) for each value of the variable, and how many missed values are shown. The "Valid Percent" column again returns the percentage but the total of the valid respondents to this question.
Finally, in the "Cumulative Percent" column we give the cumulative percentages of each category-price, that is, the percentage of valid replies of each value to which all previous valid percentages have been added. Note that in the comment on the results we will refer to the valid rate. For the relationship between two questions / variables, the affinity matrices were used where for each "cross" of two variable values the frequency of the common answers and the respective percentages are calculated. The statistical measures for the numerical variables we use are (in brackets the English terminology that appears in SPSS results): The mean (mean) or otherwise the average of our observations, the median that shows us the value below (and above) from which is 50% of our observed values, the standard deviation which is a measure of the variance of the values and also of the minimum and the maximum value (Siomkos & Mavros, 2008).

The averages and the standard deviations of all variables are calculated for both the whole sample and each group separately. Finally, a correlation analysis was performed to determine whether there is a positive or negative linear correlation between the variables for each sample group separately.
Chapter 5: Analysis of Results

This chapter includes the analyses carried out, as well as the results thereof. Initially, the characteristics of the sample, which are divided into demographics and elements of each respondent’s character, are presented. More specifically, the first part refers to the description of the whole sample, but also the profile of the sample respondents to each group, according to their responses to the demographic questionnaire questions. In the second part we report the results from the 8 categories in which we have divided our analysis. Finally, a correlation analysis is performed to determine whether there is a positive or negative linear correlation between the variables for each group of the sample separately.

5.1 Demographic characteristics

The total sample is 59 respondents. The majority of respondents are aged 30-35. Table 1 shows the largest percentage (valid percentage) of responses to demographic research questions.

The 44.2% of the sample has more than 10 years of work experience but the majority of the sample (69.2%) has 0-5 years’ experience with agile. The 34.6% of 59 respondents has been working in their team over 3 years.

As far as personal characteristics of individuals are concerned, we can figure out, from the table below, that half of the sample would describe itself with a grade of 3 on a scale 1 to 5. Half or respondents think that they are very organized and very creative at a level 4 out of 5 (mean:3.60, 3.39). The majority of the sample (57.7%) believes that it is very easy for them to describe something in detail (level 4) during a project or a task. They also think that it is easy for them to estimate the time needed for their daily tasks at a level 3 (46.2%).
<table>
<thead>
<tr>
<th>Demographic characteristics</th>
<th>Valid percent (%)</th>
<th>Missing values</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age group: 30-35</td>
<td>26.40</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>How long is your total work experience: &gt;10</td>
<td>44.2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>How long is your experience with agile: 0-5 years</td>
<td>69.2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>How long have you been working in your company: 2-5 years</td>
<td>42.4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>How long have you been working in your team: &gt;3 years</td>
<td>34.6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>On a scale from 1-5 how would you describe yourself: 3</td>
<td>48.1</td>
<td>7</td>
<td>2.96</td>
</tr>
<tr>
<td>On a scale from 1-5 how organized would you say you are: 4</td>
<td>51.9</td>
<td>7</td>
<td>3.60</td>
</tr>
<tr>
<td>On a scale from 1-5 how creative would you say you are: 4</td>
<td>42.4</td>
<td>0</td>
<td>3.39</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to explain something in detail: 4</td>
<td>57.7</td>
<td>7</td>
<td>3.63</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to estimate the time needed for your daily tasks: 3</td>
<td>46.2</td>
<td>7</td>
<td>3.23</td>
</tr>
<tr>
<td>Question</td>
<td>Rating</td>
<td>Frequency</td>
<td>Average</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>On a scale from 1-5 how easily can you be distracted: 3</td>
<td>40,4</td>
<td>7</td>
<td>3.37</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to be in a lead of a process: 4</td>
<td>40,4</td>
<td>7</td>
<td>3.71</td>
</tr>
<tr>
<td>On a scale from 1-5 how aware is your team of what your daily tasks are: 4</td>
<td>34,6</td>
<td>7</td>
<td>3.48</td>
</tr>
<tr>
<td>Do you believe that you work with big complex projects that cannot be broken down: No</td>
<td>55,9</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>How much time do you usually need to finish a unit of work: 1/2 - 2 days</td>
<td>64,4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>How often do you experience cases that might be blocked because of dependencies: 3</td>
<td>42,3</td>
<td>7</td>
<td>3.19</td>
</tr>
<tr>
<td>How often do unexpected cases pop up: 3</td>
<td>46,2</td>
<td>7</td>
<td>3.37</td>
</tr>
<tr>
<td>Who owns the project failure or success in your team: team</td>
<td>49,2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>How many people are there in the team: 5-7 people</td>
<td>40,7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Rating 1-5</td>
<td>Rating 7</td>
<td>Score</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>Are you part of a cross functional team: No</td>
<td>47,5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>On a scale 1-5 how mature do you believe that you are as a team: 4</td>
<td>38,5</td>
<td>7</td>
<td>3.27</td>
</tr>
<tr>
<td>On a scale 1-5 how much knowledge sharing do you feel that you do in your team: 4</td>
<td>40,4</td>
<td>7</td>
<td>3.46</td>
</tr>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected: 4</td>
<td>36,5</td>
<td>7</td>
<td>3.52</td>
</tr>
<tr>
<td>Is everybody in your team capable to pick up your daily tasks: No</td>
<td>44,1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>How often do you know what other team members are working on: 4</td>
<td>32,7</td>
<td>7</td>
<td>3.62</td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members: 4</td>
<td>32,7</td>
<td>7</td>
<td>3.87</td>
</tr>
<tr>
<td>How often do you have disagreements in the team: Every week</td>
<td>33,9</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Response</td>
<td>Confidence</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>Do you believe that your team could achieve better results if they would listen to your opinion more often: Neither Agree or Disagree</td>
<td>54.2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>How often do you feel that you can communicate your ideas easily: 4</td>
<td>40.4</td>
<td>7</td>
<td>3.79</td>
</tr>
<tr>
<td>What is the primary measure of progress in agile software development: Working software</td>
<td>64.4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Agile software development aims to improve productivity by eliminating unknowns and getting things right the first time: False</td>
<td>64.4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Which of these is NOT a characteristic of the waterfall method of software development: Iteration</td>
<td>54.2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>The foundation of agile software development is a set of processes and tools that helps teams adhere to a well-defined plan as they make software: False</td>
<td>42.4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
5.2 Using scrum at the moment (categories 1, 2, 3)
33 out of 59 respondents are using Scrum at the moment the research took place. 4 of them are scrum masters, 3 product owners, 1 both scrum master and team member and the rest 25 team members.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>9.1</td>
<td>9.7</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>15.2</td>
<td>25.8</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>51.5</td>
<td>80.6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>18.2</td>
<td>100.0</td>
</tr>
<tr>
<td>Total</td>
<td>31</td>
<td>93.9</td>
<td>100.0</td>
</tr>
<tr>
<td>Missing</td>
<td>2</td>
<td>6.1</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3: How satisfied are you with scrum?**

We asked all the participants to list the most difficult side effects in Agile and that's what we got as a result:

- Impediments on other teams
- Estimations and velocity
- Big projects get pushed back because of not delivering value fast enough
- Other teams that do not understand how we work (in the beginning)
- Big projects still being pushed down, because the management wants them
- Migration to the whole company going agile
- Balancing the amount of meetings and the value of them (and actually getting work done)
- Persisting outcomes of retros
- Long term planning often seems to fall to the sideline
- Confusion
- Lack of organization
- Lack of purpose at first
- Disciplin
- Deadlines
- Traditional Mindsets that conflict with Agile
- Disorder
- Long meetings
- No big picture
- Doesn't work well with distractions (in case there's a bug that needs to be fixed right away)
- Generalist team members, requires responsible and committed people to work
- Requires the team to agree which is sometimes time consuming.
- Long range goals not immediately visible.
- Planning cross team resourcing
- Domain definition vs team roles. They don't always align and things get lost in the cracks. (who owns what? Last person to push owns etc...)
- Long term focus
- Urgent tasks
- Taking over tasks of other team members
- Lot of meetings, smaller blocks of time available for software development
- Maintaining overview on the projects
- Managing multiple tasks in a short period (sprint)
- Iterations taking longer than expected

In conclusion we can see that the most important problem that people face are impediments on other teams, lack of the bigger picture and many meetings that do not allow them to concentrate.

5.3 Team Member Using Scrum Now

Table 5.3 shows the descriptive statistics of team members who are using scrum at the moment. Mean and standard deviation of some variables concerning the personal characteristics of team members are presented at the table below. Worth noting that we have a high mean value in the following variables: on a scale from 1-5 how aware is your team of what your daily tasks are, how often do you know what other team members are working on, how often do you feel that you can communicate your ideas easily, how satisfied are you with scrum (scale 1-5). This fact proves the high levels of agreement and satisfaction of team members who use scrum now.

Table 4: Descriptive Statistics of personalities

<table>
<thead>
<tr>
<th>N</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how introvert would you describe yourself?</td>
<td>18</td>
<td>1</td>
<td>4</td>
<td>2.72</td>
</tr>
<tr>
<td>On a scale from 1-5 how organised would you say you are?</td>
<td>18</td>
<td>2</td>
<td>5</td>
<td>3.72</td>
</tr>
<tr>
<td>On a scale from 1-5 how creative would you say you are?</td>
<td>18</td>
<td>2</td>
<td>5</td>
<td>3.56</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to explain something in detail?</td>
<td>18</td>
<td>1</td>
<td>5</td>
<td>3.56</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to estimate the time needed for your daily tasks?</td>
<td>18</td>
<td>2</td>
<td>5</td>
<td>3.67</td>
</tr>
<tr>
<td>Question</td>
<td>Mean Score</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On a scale from 1-5 how easily can you be distracted?</td>
<td>3.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to be in a lead of a process?</td>
<td>3.78</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>On a scale from 1-5 how aware is your team of what your daily tasks are?</strong></td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How often do you experience cases that might be blocked because of dependencies?</td>
<td>2.61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How often do unexpected cases pop up?</td>
<td>3.17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>On a scale 1-5 how mature do you believe that you are as a team?</strong></td>
<td>3.44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>On a scale 1-5 how much knowledge sharing do you feel that you do in your team?</strong></td>
<td>3.67</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected?</td>
<td>3.72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How often do you know what other team members are working on?</strong></td>
<td>4.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members?</td>
<td>3.89</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We can see from the results of the analysis that 18 out of 59 respondents are team members aged between 35 and 40. One respondent out of 18 has responded that his/her role in the team is both scrum master and team member. 14 out of 18 have total work experience above 10 years, 1 of them has 2-5 years of total experience while 3 of them have 5-10 years’ work experience.

50% of them have 5-10 years experience with Agile, 6 out of 18 have 2-5 years, 2 have 0-2 years and only one of them has more 10 years experience with Agile.

Most of the responders have been working 2-5 in the company (10/18), 4 out of the them 5-10 years, 3 out of 18 from 0-2 years and only one responder has been working more than 10 years.

We also asked them for how long they have been working in their current team, with the scope to see if in what way could this affect their answers. Moreover we saw that 8 out 18 have been working in the same team for more than 3 years, 4 out 18 1-3 years and the rest 0 to 1 year.
**Graph 2: Profile of respondents**

**Table 5: Do you believe that you work with big complex projects that cannot be broken down?**

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No</strong></td>
<td>11</td>
<td>61,1</td>
<td>61,1</td>
<td>61,1</td>
</tr>
<tr>
<td>Sometimes</td>
<td>6</td>
<td>33,3</td>
<td>33,3</td>
<td>94,4</td>
</tr>
<tr>
<td><strong>Yes</strong></td>
<td>1</td>
<td>5,6</td>
<td>5,6</td>
<td>100,0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td>100,0</td>
</tr>
</tbody>
</table>
Table 6: How much time do you usually need to finish a unit of work?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2 - 2 days</td>
<td>15</td>
<td>83,3</td>
<td>83,3</td>
<td>83,3</td>
</tr>
<tr>
<td>2-5 days</td>
<td>3</td>
<td>16,7</td>
<td>16,7</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Graph 3: Dependencies and unexpected cases

14 out 18 responded that the team owns the failure of the project and only 3 out of 18 that the failure is owned by individuals related to the task. Finally, 1 out of 18 responded that the manager is the one who owns the project failure.

Most of the teams consist of 5-7 people (11/18), but we also have bigger teams with more than 7 people (3/18). Moreover, 10 out of 18 people are part of a cross functional team.
Half of 18 respondents using scrum at the moment believe that everybody in the team is capable of picking up the daily tasks whereas the other half of the sample don't have the same opinion.
**Graph 5:** How often do you know what other team members are working on

- 39% (0%)
- 44% (17%)
- 33% (0%)
- 22% (0%)
- 3% (0%)

**Graph 6:** How satisfied are you by collaboration achieved among your team members

- 45% (33%)
- 33% (22%)
- 22% (0%)
- 3% (0%)
- 1% (0%)
Therefore 3 out of 18 believe that their team could achieve better results if they would listen to their opinion more often. However 15 out of 18 respondents point out that neither agree or disagree with this opinion.
It is interesting to point out also that only 50% of the responders claim that everybody from their team are able to pick up their daily tasks and that 3 out of 18 believe that their team could achieve better results if they would listen to their opinion more often.

Following we asked the participants to reply to four basic questions regarding Agile. The first question was “What is the primary measure of progress in agile software development?”, where 14 out of 18 responded correctly. The second question was “Agile software development aims to improve productivity by eliminating unknowns and getting things right the first time” where 15 out of 18 gave again the right response.

The third was “Which of these is NOT a characteristic of the waterfall method of software development? ” where 16/18 responders knew the answer.

The last question was “The foundation of agile software development is a set of processes and tools that helps teams adhere to a well-defined plan as they make software.” In which the percentage of the correct answers was a bit lower (10/18), whereas 5/18 responded wrong and the rest said that they didn’t know the correct answer. Following, we tried to see what is working good and what not within each team.
Table 7: Does the team usually agree with its PO decisions?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost always</td>
<td>7</td>
<td>38,9</td>
<td>38,9</td>
<td>38,9</td>
</tr>
<tr>
<td>Often</td>
<td>6</td>
<td>33,3</td>
<td>33,3</td>
<td>72,2</td>
</tr>
<tr>
<td>Sometimes</td>
<td>5</td>
<td>27,8</td>
<td>27,8</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Do you feel that you provide enough feedback and transparency regarding the cases you work on to your Product Owner?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost always</td>
<td>9</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Often</td>
<td>5</td>
<td>27,8</td>
<td>27,8</td>
<td>77,8</td>
</tr>
<tr>
<td>Sometimes</td>
<td>4</td>
<td>22,2</td>
<td>22,2</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: How often do you do pair programming in your team?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost always</td>
<td>1</td>
<td>5,6</td>
<td>5,6</td>
<td>5,6</td>
</tr>
<tr>
<td>Never</td>
<td>1</td>
<td>5,6</td>
<td>5,6</td>
<td>11,1</td>
</tr>
<tr>
<td>Often</td>
<td>4</td>
<td>22,2</td>
<td>22,2</td>
<td>33,3</td>
</tr>
<tr>
<td>Seldom</td>
<td>3</td>
<td>16,7</td>
<td>16,7</td>
<td>50,0</td>
</tr>
<tr>
<td>Sometimes</td>
<td>9</td>
<td>50,0</td>
<td>50,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>
Table 10 shows the value of the daily meetings as it is recognized by the respondents. We can figure out that 5 out of 18 respondents realize sometimes the value of their daily meetings.

**Table 10: Do you recognize the value of your daily meetings?**

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost always</td>
<td>6</td>
<td>33,3</td>
<td>33,3</td>
<td>33,3</td>
</tr>
<tr>
<td>Often</td>
<td>6</td>
<td>33,3</td>
<td>33,3</td>
<td>66,7</td>
</tr>
<tr>
<td>Seldom</td>
<td>1</td>
<td>5,6</td>
<td>5,6</td>
<td>72,2</td>
</tr>
<tr>
<td>Sometimes</td>
<td>5</td>
<td>27,8</td>
<td>27,8</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

**Graph 9: Satisfaction by scrum**

5.4 Product Owner Using Scrum Now

3 people of our sample mention themselves as product owners using scrum now. We can conclude from the results that 2 of 3 are aged 25-30 years and one of them is between 30 and 35 years old. One product owner has 2-5 years total work experience and 2 of them
have 5 to 10 years of total working experience. All product owners of the sample have 2 to 5 years experience with agile, one of them has been working in the company for 0-2 years (>3 work in the team), the other one for 2-5 years (1-3 years work in the team) and the other one for 5-10 years (6months-1year work in the team).

Graph 10: Personal abilities of team members

All 3 product owners feel that they work with big complex projects that cannot be broken down (scale 3). They also state that they usually need ½ to 2 days to finish a unit of work. Two of them experience often cases that might be blocked because of dependencies while one of them experience very often cases like these. Two of them claim that unexpected cases usually pop up. All of them state that the team owns the project failure or success (on a scale 3). One of them report that there are more than 7 people working in his/her team while the other two claim that there are between 3 to 7 people working in their team. Two of them are not part of cross functional team while 1 of them is actually part of it. All product owners of our sample think that their team is very mature (scale 4). Two of them believe that in their team there is not much knowledge sharing. All three of them feel that
all team members follow very much the workflow of the team as expected. 66.7% of the product owners doesn’t believe that everybody in their team is capable to pick up their daily tasks while 33.3% do so. The three of them know very often what other team members are working on. Two out of three are not so satisfied by the collaboration achieved among team members.

One product owner has disagreements in the team every 2-3 weeks, the other one has disagreements in the team every month or more while the other one claims that every week has disagreements in the team. One of three product owners disagree that his/her team could achieve better results if they would listen to his/her opinion more often while the other two neither agree nor disagree. Moreover, two of them don’t feel that they can communicate their ideas easily. One out of three product owners state that formal documentation is the primary measure of progress in agile software development. Two out of three claim that working software is the primary measure of progress. All of them state that iteration is not a characteristic of the waterfall method of Software development. Two of them believe that the foundation of agile software development is not a set of processes and tools that helps teams adhere to a well-defined plan as they make software.

### Table 11: Do you feel your team agrees with your decisions?

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>2</td>
<td>66.7</td>
<td>66.7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33.3</td>
<td>33.3</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

### Table 12: Do you feel that you respect his role?

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>2</td>
<td>66.7</td>
<td>66.7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33.3</td>
<td>33.3</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Table 13: How satisfied are you with scrum?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing System</td>
<td>4</td>
<td>2</td>
<td>66,7</td>
<td>100,0</td>
</tr>
<tr>
<td>System</td>
<td>1</td>
<td>1</td>
<td>33,3</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>1</td>
<td>100,0</td>
<td>100,0</td>
</tr>
</tbody>
</table>

According to the two of three product owners of the sample the 3 most difficult side effects in agile are the following:

➢ Other teams that do not understand how we work (in the beginning) 2. Big projects still being pushed down, because the management wants them 3. Migration to the whole company going agile

➢ No deadlines or urgency, unclear roles, ineffective decision making

5.5 Scrum Master Using Scrum Now

We can see from the results of the sample that there are 4 Scrum Masters who are using scrum now aged between 25 to 45 years. Two of them have over 10 years total work experience while 1 of them has 2-5 years. Three of them have 2-5 years experience with agile and are working 2-5 years in their company while one of them has 0-2 years experience with agile and is working in his/her company. 3 out of 4 have been working in their team for 6 months to 1 year.
All scrum masters believe that they work with big complex projects that cannot be broken down (scale 4). Two out of four need ½ to 2 days in order to finish a unit of work while the other two need 2 to 5 days to complete a unit of work.

Graph 11: Abilities of personalities of Scrum Masters

Graph 12: Unexpected cases-Knowledge sharing- Collaboration- Communication in the team
All scrum masters feel that all team members follow the workflow of the team as expected (scale 4). Two out of 4 believe that everydovy in the team is capable to pick up his/her daily tasks in contrary to the other two who don’t have the same opinion. 2 out of 4 have every 2-3 weeks disagreements in the team and they neither agree or disagree that their team could achieve better results if they would listen to their opinion more often. The primary measure of progress in agile software development for two of them is working software and for one of them is formal documentation. All of them believe that agile software development doesn’t aim to improve productivity by eliminating unknowns and getting things right the first time.

**Graph 13: Which of these is NOT a characteristic of Waterfall method of software development?**

- **50%** Clearly defined roles
- **25%** Iteration
- **25%** Set requirements and design up front

Two out of four respondents think that the foundation of agile software development is a set of processes and tools that helps teams adhere to a well-defined plan as they make software. 3 out of 4 believe that as scrum masters support the team besides helping with the ceremonies and they feel that their team respects their role.
Table 14: How often do you do pair programming in your team?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seldom</td>
<td>2</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Sometimes</td>
<td>2</td>
<td>50,0</td>
<td>50,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 15: Do you recognise the value of your daily meetings?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost always</td>
<td>2</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Often</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>75,0</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Graph 14: How satisfied are you with Scrum

Finally scrum masters were called to list the 3 most difficult side effects in Agile. Two of them mentioned the following most difficult side effects in agile:
➢ Long range goals not immediately visible. Planning cross team resourcing. Domain definition vs team roles. They don't always align and things get lost in the cracks. (who owns what? Last person to push owns etc...)

➢ Takes time for the team to understand Agile/SCRUM and grow, keeping the momentum going.

5.6 Team Member that has used Scrum

Team Members that are not using scrum at the moment but have used Scrum are 3, two of them aged 25-30 years while one of them 35-40 years with total work experience 5-10 years. 2 out of 3 have 0-2 years experience with agile while 1 out of 3 has 2-5 years agile experience. 2 out of 3 are working 2-5 years in their company and 6 months to 3 years in their team. All of them are part of a cross functional team.

*Graph 15: Personal abilities of team members that have used scrum*
One of the three team members that has used scrum believe that he/she works with complex projects that cannot be broken down while one out of three has the opposite idea. One team member think that sometimes he/she can work with big complex projects. 2 out of 3 usually need ½ to 2 days in order to finish a unit of work. 1 out of 3 believes that direct manager owns the project failure or success in his/her team while the other 2 mention that Individuals related to the task and team own the failure or success. Two of them mention that there are 5 to 7 people in their team. They also believe they are enough mature as a team (scale 3 and 4). Furthermore it is said from one member team that there is not much knowledge sharing in his/her team. However one out of three feels that all team members follow the workflow of the team as expected (scale 5). 2 out of 3 believe that there is a lack of capability in his/her team from some team members who are not always capable to pick up their daily tasks. Team member mention that he/she doesn’t know always what other team members are working on. One team member is very much satisfied by the collaboration achieved among team members. 2 out of 3 respondents claim that they have disagreements in their team every 2-3 weeks and they believe that their team could achieve better results if they would listen to their opinion more often. 2 of them feel that they can communicate enough with the team members about their ideas. All team members mention that the primary measure of progress in agile software development is working software. 2 out of 3 don’t agree with the idea that agile aims to improve productivity by
eliminating unknowns and getting things right the first time, although they believe that iteration is not a characteristic of the waterfall method of software development.

Table 16: Did the team use to agree with its PO decisions?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>33,3</td>
</tr>
<tr>
<td>Seldom</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>66,7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 17: Do you feel that you used to provide enough feedback and transparency regarding the cases you were working on to your Product Owner?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>2</td>
<td>66,7</td>
<td>66,7</td>
<td>66,7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 18: How often did you do pair programming?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>2</td>
<td>66,7</td>
<td>66,7</td>
<td>66,7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 19: Did you recognize the value of your daily meetings?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>2</td>
<td>66,7</td>
<td>66,7</td>
<td>66,7</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>33,3</td>
<td>33,3</td>
<td>100,0</td>
</tr>
</tbody>
</table>
According to the 2 team members that have used agile, the 3 most important benefits which their team got from using Scrum are the following:

➢ stability, reduction of waste
➢ transparency

Finally one out of three team members mention that the most difficult side effect in Agile is the difficulty of finding right people for the team.

5.7 Product Owner that has used Scrum

As we can assume from the results of our analysis we don’t have valid results for the category of product owner who has used scrum in the past. There is only one product owner that has used scrum aged between 25 to 30 years with total work experience 2-5 years and 5-10 years with agile. He/she is working in the company over 10 years and believes that he/she is not much creative (scale 2).

5.8 Scrum Master that has used Scrum

At this section we analyse the results regarding Scrum masters who have used Scrum in the past. The table below shows the personal characteristics of Scrum masters. We can see that we have 2 Scrum masters who have used scrum in the past.

<table>
<thead>
<tr>
<th></th>
<th>1. Scrum master</th>
<th>2. Scrum master</th>
</tr>
</thead>
<tbody>
<tr>
<td>age group</td>
<td>25-30</td>
<td>30-35</td>
</tr>
<tr>
<td>total work experience</td>
<td>2-5</td>
<td>5-10</td>
</tr>
<tr>
<td>experience with agile and work in the company</td>
<td>2-5</td>
<td>2-5</td>
</tr>
<tr>
<td>On a scale from 1-5 how creative would you say you are</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Question</td>
<td>I don't know</td>
<td>Working software</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>primary measure of progress in agile software development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>you as a scrum master were coaching/supporting the team besides helping with the ceremonies?</td>
<td>Sometimes</td>
<td>Yes</td>
</tr>
<tr>
<td>you could have given extra support to the team?</td>
<td>maybe</td>
<td>no</td>
</tr>
<tr>
<td>On a scale from 1-5 how close to the term &quot;Scrum Guy&quot; do you believe that you were?</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>On a scale from 1-5 how close to the term &quot;True ScrumMaster&quot; do you believe that you were?</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Did you feel that your PO respected your role?</td>
<td>Sometimes</td>
<td>Sometimes</td>
</tr>
<tr>
<td>Did you feel that your team respected your role?</td>
<td>Sometimes</td>
<td>Sometimes</td>
</tr>
<tr>
<td>How often did you do pairprogramming?</td>
<td>Often</td>
<td>sometimes</td>
</tr>
<tr>
<td>During your estimation you use:</td>
<td>Big/Uncertain</td>
<td>T-shirt sizes</td>
</tr>
</tbody>
</table>
**Graph 17: Estimation of agile teams**

- How often did you join meetings of other agile teams in the company?
- Did you skip retrospectives?
- Did you plan your sprint all together?
- During the estimation, how often did you use to agree within the team?
- How often did you estimate correctly?
- How often did your team estimate correctly?

**Graph 18: Projects and progress of other teams**

- How valuable did you find the sprint retrospectives for the team?
- How informed were you regarding the projects and progress of other teams in the company?
Table 21: Were you satisfied by the time your sprint lasted?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Neither satisfied or dissatisfied</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 22: Did the team use to reach its goals?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sometimes</td>
<td>2</td>
<td>100,0</td>
<td>100,0</td>
<td>100,0</td>
</tr>
</tbody>
</table>

Table 23: Were the team’s goals realistic?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 24: Did you recognize the value of your daily meetings?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seldom</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Sometimes</td>
<td>1</td>
<td>50,0</td>
<td>50,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

Finally we see that the 2 Scrum masters of the sample have different point of view regarding the team’s goals and the daily meetings that they had in their team.
5.9 Team member that has used Agile in the past

According to the results of the research team members that have used agile methods in the past are 4 aged between 30 to 45 years with total work from 5 to over 10 years. 2 out of 3 have experience with agile for 2-5 years. 1 out of 3 has 0-2 years while the other one has 5-10 years experience with agile. Half of them are working in the company between 2 to 5 years while the other half is working for 5 to 10 years. They have been working in the team between 1 to 3 years.

*Graph 19: Profil of team members having used agile*

2 out of 4 believe that sometimes they work with big complex projects that cannot be broken down. 3 out of 4 usually need ½ to 2 days in order to finish a unit of work.
1 out of 4 team members state that team (concluding 5 to 7 people) owns a project failure or a success in the team while the other 3 believe that the blame is for individuals related to the task in a team concluding over 7 people. 3 out of 4 correspondents are a part of a cross functional team. 2 out of 4 don’t know if everybody in their team is capable to pick up their daily tasks while the other half feel that not everybody in the team has capabilities.

2 out of 4 report that they have disagreements in the team every 2 to 3 weeks, one of them claims that disagreements appear every month or more while the other one states that every week at least one disagreement pops up.

1 out of 4 disagree with the idea that their team could achieve better results if they would listen to their opinion more often whereas one of them agree with this idea.
As the primary measure of progress in agile software development is concerned, 2 out of 4 team members point out the working software while one of them mentions the velocity. 2 out of 4 disagree with the idea that agile software development aims to improve productivity by eliminating unknowns and getting things right the first time although 3 of them state that the foundation of agile software development is not a set of processes and tools that helps teams adhere to a well-defined plan as they make software.

*Graph 21: Which of these is NOT a characteristic of the Waterfall method of software development*
According to the team members (3 answers) who have used agile methods, the 3 most important benefits which their team got from using an Agile framework are the following:

- Flexibility in work selection
- Insight into throughput and delays
- Knowledge sharing

The 3 most difficult side effects in Agile (3 answers) are the following:

- Actually getting results out of the retrospective feedback loop
- Knowledge transfer outside of specialties (Dev vs QE vs System Engineer)
- Interfacing with stakeholders that don't know much about Agile
Managing stakeholders who aren’t aware
  - Unclear how to use it when you are working in multiple teams; How to use it in marketing teams; How to work agile with other scrum-teams

**Table 26: Why did you decide on this framework?**

<table>
<thead>
<tr>
<th>Reason</th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>25,0</td>
</tr>
<tr>
<td>Alignment of working</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>50,0</td>
</tr>
<tr>
<td>Because the whole company is going there</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>75,0</td>
</tr>
<tr>
<td>Chosen over Scrum because of easier inclusion of testing within the workflow</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>100,0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4</strong></td>
<td><strong>100,0</strong></td>
<td><strong>100,0</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Table 27: Would you consider switching to Scrum?**

<table>
<thead>
<tr>
<th>Response</th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>25,0</td>
</tr>
<tr>
<td>Maybe</td>
<td>2</td>
<td>50,0</td>
<td>50,0</td>
<td>75,0</td>
</tr>
<tr>
<td>Yes</td>
<td>1</td>
<td>25,0</td>
<td>25,0</td>
<td>100,0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4</strong></td>
<td><strong>100,0</strong></td>
<td><strong>100,0</strong></td>
<td></td>
</tr>
</tbody>
</table>

**5.10 Team member that plans to use Agile in the future**

It is interesting to point out that 10 team members would like to use agile methods in the future. This fact is very hopeful for many companies as the interest and the attract for agile methods is increasing more and more.

These 10 team members are not using scrum at the moment, haven’t used scrum in the past or agile methodologies. Their demographic characteristics and their character profile are presented in the following graphs and tables.
8 out of 10 team members have 0 to 2 years experience with agile while the rest 2 have 5-10 years experience. 1 of them has been working between 5 to 10 years in the company while the rest 9 team members have been working from 0 to 5 years. 6 out of 10 have been
working in their team from 6 months to 3 years and they mention that the team included 3 to 7 people.

7 out of 10 point out that they don’t work with big complex projects that cannot be broken down. 8 out of 10 need ½ to 2 days in order to finish a unit of work. 6 out of 10 state that they experience very often cases that might be blocked because of dependencies. 5 team members think that unexpected cases pop up often (scale 3) while 4 out of 10 believe that unexpected cases pop up very often (scale 4). 9 out 10 point out that Individuals related to the task usually take the blame for a project failure or take the credits for a success in their team while one out of ten mention that direct manager owns the project failure or success for a project. 7 out of 10 don’t belong at a cross functional team while 3 out of 10 are part of this team.
Everybody in their team is capable to pick up their daily tasks according to 5 team members while four of them believe that there is a lack of capability in their team.

We can see from the graph below that team members planning to use agile in the future seem to be very satisfied by the collaboration achieved among other team members at a level 3 (67%).
We can see from graph 28 the opinion of team members about the disagreements taking place in the team. 4 of team members claim that they have disagreements in the team every week while 2 of them mention that they never disagree in their team.

The majority of the sample neither agree or disagree with the idea that their team could achieve better results if they would listen to his/her opinion more often.
Graph 29: Do you believe that your team could achieve better results if they would listen to your opinion more often;

Graph 30: Agile Software development

According to 2 respondents, handoffs between specialized groups are not characteristics of the waterfall method of software development while 2 out of 10 believe that set
requirements and design up front don’t belong to the waterfall method of software development

Table 28: Which of these is NOT a characteristic of the waterfall method of software development?

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>A sequence of stages</td>
<td>1</td>
<td>10,0</td>
<td>10,0</td>
<td>10,0</td>
</tr>
<tr>
<td>Handoffs between specialized groups</td>
<td>2</td>
<td>20,0</td>
<td>20,0</td>
<td>30,0</td>
</tr>
<tr>
<td>I don’t know</td>
<td>4</td>
<td>40,0</td>
<td>40,0</td>
<td>70,0</td>
</tr>
<tr>
<td>Iteration</td>
<td>1</td>
<td>10,0</td>
<td>10,0</td>
<td>80,0</td>
</tr>
<tr>
<td>Set requirements and design up front</td>
<td>2</td>
<td>20,0</td>
<td>20,0</td>
<td>100,0</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>100,0</td>
<td>100,0</td>
<td></td>
</tr>
</tbody>
</table>

1 out of 10 team members states that it takes over 5 days to start working on a task (Business requirements, understanding systems, components related to the task).

As fas way of working is concerned, it is interesting to point out that 5 out of 10 feel that agile transformation can help their team. 3 out of 10 mention that their team is ready to switch to the agile way of working while 4 out of 10 are ready to change their way of working.

Graph 31: Way of working
Table 29 On a scale 1-5 how confident do you feel about this change?

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Valid Percent</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>10,0</td>
<td>10,0</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>60,0</td>
<td>60,0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>30,0</td>
<td>30,0</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>100,0</td>
<td>100,0</td>
</tr>
</tbody>
</table>

Half of team members listed three problems that they must overcome to make the team feel confident:

➢ communication, manager
➢ Develop skills outside our current individual expertise Let go of the fear of change
  Share tasks more often
➢ More people and less BAU
➢ Speed of decision making
➢ we would need to get rid of BAU stuff and tasks which need no involvement of any other team at all.

5.11 Correlations among categories 1,2,3

In this section we present the correlations among categories 1,2,3. The correlation analysis was performed to determine the relationship between the variables for each individual sample separately. The results of the correlation analysis are presented in the tables below, where there appear to be several linear correlations between the variables. The purpose of this is to come to conclusions about the relations among these categories and finally to understand how the satisfaction acquired by using scrum affects the team.
Table 30: Correlation among description, organization and distraction of team members

<table>
<thead>
<tr>
<th>On a scale from 1-5 how easily can you be distracted?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>On a scale from 1-5 how introvert would you describe yourself?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>On a scale from 1-5 how organised would you say you are?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td>,439*</td>
<td>,013</td>
<td>31</td>
<td>-,497**</td>
</tr>
</tbody>
</table>

* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

Table 31: Correlation among communication of ideas and lead of the process

<table>
<thead>
<tr>
<th>On a scale from 1-5 how easy is it for you to be in a lead of a process?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>How often do you feel that you can communicate your ideas easily?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td>,510**</td>
</tr>
</tbody>
</table>

Table 32: Correlation among communication of ideas, lead of the process and retrospectives

<table>
<thead>
<tr>
<th>How often do you feel that you can communicate your ideas easily?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>On a scale 1-5 how valuable do you find the sprint retrospectives for the team?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td>,401*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On a scale from 1-5 how easy is it for you to be in a lead of a process?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td>,444**</td>
</tr>
</tbody>
</table>
### Table 33: Correlation among creativity and other characteristics of team members

<table>
<thead>
<tr>
<th>Question</th>
<th>Pearson Correlation Sig. (2-tailed)</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how easy is it for you to explain something in detail?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how creative would you say you are?</td>
<td>-.366*</td>
<td>31</td>
</tr>
<tr>
<td>how informed are you regarding the projects and progress of other teams in the company?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how valuable do you find the sprint retrospectives for the team?</td>
<td>-.404*</td>
<td>30</td>
</tr>
<tr>
<td>how easy is it for you to be in a lead of a process?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how aware is your team of what your daily tasks are?</td>
<td>.474**</td>
<td>30</td>
</tr>
<tr>
<td>how organised would you say you are?</td>
<td>.385*</td>
<td>31</td>
</tr>
<tr>
<td>how creative would you say you are?</td>
<td>.375*</td>
<td>31</td>
</tr>
</tbody>
</table>

### Table 34: Correlations among daily tasks and other variables

<table>
<thead>
<tr>
<th>Question</th>
<th>Pearson Correlation Sig. (2-tailed)</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how aware is your team of what your daily tasks are?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>On a scale from 1-5 how organised would you say you are?</td>
<td>.400*</td>
<td>31</td>
</tr>
<tr>
<td>On a scale from 1-5 how creative would you say</td>
<td>.375*</td>
<td>31</td>
</tr>
</tbody>
</table>
We can see from the table below that according to pearson correlation there is a strong enough positive correlation (0.717) between the satisfaction by the collaboration achieved among team members and the level of matureness of the team. This colleration is also statistical significant at the level of statistical significance 0.01 (sig. 0.000<0.01).
### Table 35: Correlations among workflow and collaboration among team members

<table>
<thead>
<tr>
<th>Question</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>N</th>
<th>How often do you know what other team members are working on?</th>
<th>How satisfied are you by the collaboration achieved among your team members?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected?</td>
<td>-.432*</td>
<td>.015</td>
<td>31</td>
<td>-.235</td>
<td>-.366*</td>
</tr>
<tr>
<td>How often do unexpected cases pop up?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On a scale 1-5 how mature do you believe that you are as a team?</td>
<td>,696**</td>
<td>.000</td>
<td>31</td>
<td>,628**</td>
<td>,717**</td>
</tr>
<tr>
<td>On a scale 1-5 how much knowledge sharing do you feel that you do in your team?</td>
<td>,578**</td>
<td>.001</td>
<td>31</td>
<td>,504**</td>
<td>,297**</td>
</tr>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected?</td>
<td>1*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How often do you know what other team members are working on?</td>
<td>,647**</td>
<td>1*</td>
<td>31</td>
<td>,778**</td>
<td></td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members?</td>
<td>,625*</td>
<td>.000</td>
<td>31</td>
<td>,778**</td>
<td>1</td>
</tr>
<tr>
<td>On a scale from 1-5 how close to the term &quot;Scrum Guy&quot; do you believe that your Scrum Master is?</td>
<td>-.401</td>
<td>.064</td>
<td>22</td>
<td>-594**</td>
<td>-681**</td>
</tr>
</tbody>
</table>
Table 36: Correlations among maturity of the team and knowledge sharing

<table>
<thead>
<tr>
<th></th>
<th>How often do unexpected cases pop up?</th>
<th>On a scale 1-5 how mature do you believe that you are as a team?</th>
<th>On a scale 1-5 how much knowledge sharing do you feel that you do in your team?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>On a scale 1-5 how mature do you believe that you are as a team?</strong></td>
<td>Pearson Correlation -0.362*</td>
<td>1</td>
<td>0.545**</td>
</tr>
<tr>
<td><strong>On a scale 1-5 how much knowledge sharing do you feel that you do in your team?</strong></td>
<td>Pearson Correlation -0.148</td>
<td>0.545**</td>
<td>1</td>
</tr>
<tr>
<td><strong>Do you feel that all team members follow the workflow of the team as expected?</strong></td>
<td>Pearson Correlation -0.432*</td>
<td>0.696**</td>
<td>0.578**</td>
</tr>
<tr>
<td><strong>How satisfied are you by the collaboration achieved among your team members?</strong></td>
<td>Pearson Correlation -0.366*</td>
<td>0.717**</td>
<td>0.297</td>
</tr>
</tbody>
</table>

Table 37: Correlations among "True ScrumMaster" and other variables

<table>
<thead>
<tr>
<th></th>
<th>On a scale from 1-5 how close to the term &quot;True ScrumMaster&quot; do you believe that your Scrum Master is?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How often do unexpected cases pop up?</strong></td>
<td>Pearson Correlation -0.524*</td>
</tr>
<tr>
<td><strong>On a scale 1-5 how mature do you believe that you are as a team?</strong></td>
<td>Pearson Correlation -0.427*</td>
</tr>
<tr>
<td>Question</td>
<td>Pearson Correlation</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected?</td>
<td>0.519*</td>
</tr>
<tr>
<td>How often do you know what other team members are working on?</td>
<td>0.583**</td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members?</td>
<td>0.704**</td>
</tr>
<tr>
<td>On a scale from 1-5 how close to the term &quot;Scrum Guy&quot; do you believe that your Scrum Master is?</td>
<td>-0.718**</td>
</tr>
</tbody>
</table>

Table 38: Correlation among satisfaction by scrum and collaboration of the team

<table>
<thead>
<tr>
<th>How satisfied are you with scrum?</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how aware is your team of what your daily tasks are?</td>
</tr>
<tr>
<td>How often do you know what other team members are working on?</td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members?</td>
</tr>
<tr>
<td>On a scale 1-5 how valuable do you find the sprint retrospectives for the team?</td>
</tr>
</tbody>
</table>

* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).
From all the above tables, we find that most variables, concerning categories 1 to 3, are positively correlated, suggesting that when the one variable is changed, the statistically significant associated variables will also be changed.

5.12 Correlations among categories 4, 5, 6

In this section the correlations between categories 4, 5, 6 are presented. Initially, some features of the personality of the respondents are correlated. In Table 5.41 we observe the negative correlation between the idea that a person (who has used scrum in the past) has for his or herself and the skill of organization in the team. The correlation is not intense (0.440) and is statistically significant at a significance level of 0.05 (sig 0.046). Instead, there is a positive correlation between the creativity of individuals and the idea they have for themselves.

Table 39: Correlations among personal characteristics

<table>
<thead>
<tr>
<th>On a scale from 1-5 how introvert would you describe yourself?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>N</th>
<th>On a scale from 1-5 how organised would you say you are?</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how organised would you say you are?</td>
<td>-0.440*</td>
<td>0.046</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On a scale from 1-5 how creative would you say you are?</td>
<td>0.476*</td>
<td>0.029</td>
<td>21</td>
<td>0.075</td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.42 shows the correlations between the level of competencies and skills that respondents feel they have. More specifically, there is a positive correlation between the ability of the individual to describe something in detail and his ability to lead a group. That is, as the ability to explain a project with clarity and detail increases, the ability to lead the
group increases too. Also, as the person's ability to estimate the time needed for daily tasks increases, the ability to explain something in detail increases. Both correlations are statistically significant at a statistical significance level of 0.05.

Table 40: Correlations among abilities and skills

<table>
<thead>
<tr>
<th></th>
<th>On a scale from 1-5 how easy is it for you to explain something in detail?</th>
<th>On a scale from 1-5 how easy is it for you to estimate the time needed for your daily tasks?</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how easy is it for you to estimate the time</td>
<td>Pearson Correlation</td>
<td>0.441*</td>
</tr>
<tr>
<td>needed for your daily tasks?</td>
<td>Sig. (2-tailed)</td>
<td>0.046</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>21</td>
</tr>
<tr>
<td>On a scale from 1-5 how easy is it for you to be in a lead of</td>
<td>Pearson Correlation</td>
<td>0.465*</td>
</tr>
<tr>
<td>a process?</td>
<td>Sig. (2-tailed)</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 41: Correlation among ability of communication and this of explanation in detail

<table>
<thead>
<tr>
<th></th>
<th>How often do you feel that you can communicate your ideas easily?</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale from 1-5 how easy is it for you to explain something</td>
<td>Pearson Correlation</td>
</tr>
<tr>
<td>in detail?</td>
<td>Sig. (2-tailed)</td>
</tr>
<tr>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>
Table 42: Correlation among daily tasks, matureness and workflow of the team

| On a scale 1-5 how mature do you believe that you are as a team? | Pearson Correlation | 0.627** |
| Do you feel that all team members follow the workflow of the team as expected? | Pearson Correlation | 0.534* |

We can see from the table below that there is a positive correlation between the workflow of the team and the matureness of the team. The correlation is quite high (0.764) and statistically significant at a statistical significance level $\alpha = 0.01$ (sig=0.000).

Table 43: Correlation between the workflow and the matureness of the team

| Do you feel that all team members follow the workflow of the team as expected? | Pearson Correlation | 0.764** |
| How satisfied are you by the collaboration achieved among your team members? | Pearson Correlation | 0.581** |
Table 44: Correlations among workflow and collaboration of the team

<table>
<thead>
<tr>
<th></th>
<th>Do you feel that all team members follow the workflow of the team as expected?</th>
<th>How often do you know what other team members are working on?</th>
<th>How satisfied are you by the collaboration achieved among your team members?</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a scale 1-5 how mature do you believe that you are as a team?</td>
<td>Pearson Correlation: 0.764**</td>
<td>0.327</td>
<td>0.581**</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed): 0.000</td>
<td>0.148</td>
<td>0.006</td>
</tr>
<tr>
<td></td>
<td>N: 21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>On a scale 1-5 how much knowledge sharing do you feel that you do in your team?</td>
<td>Pearson Correlation: 0.406</td>
<td>0.447</td>
<td>0.497</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed): 0.068</td>
<td>0.042</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>N: 21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Do you feel that all team members follow the workflow of the team as expected?</td>
<td>Pearson Correlation: 1</td>
<td>0.447*</td>
<td>0.776**</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed): 0.042</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N: 21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>How often do you know what other team members are working on?</td>
<td>Pearson Correlation: 0.447*</td>
<td>1</td>
<td>0.316</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed): 0.042</td>
<td>0.163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N: 21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>How satisfied are you by the collaboration achieved among your team members?</td>
<td>Pearson Correlation: 0.776**</td>
<td>0.316</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed): 0.000</td>
<td>0.163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N: 21</td>
<td>21</td>
<td>21</td>
</tr>
</tbody>
</table>

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

5.13 Correlations among categories 7,8

The results of the correlation analysis for categories 7 and 8 are presented in the tables below, where there appear to be several linear correlations between the variables. There is
a negative correlation among knowledge sharing in the team and distraction which is logical enough because as knowledge sharing increases, distraction among team members declines. This is a statistical significant correlation at a level of significance 0,05 (sig=0.026<0,05).

**Table 45: Correlation among Knowledge sharing and team distraction**

<table>
<thead>
<tr>
<th>On a scale from 1-5 how easily can you be distracted?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 45: Correlation among Knowledge sharing and team distraction

**Table 46: Correlation among abilities of communication-explanation in detail**

<table>
<thead>
<tr>
<th>How often do you feel that you can communicate your ideas easily?</th>
<th>Pearson Correlation</th>
<th>Sig. (2-tailed)</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 46: Correlation among abilities of communication-explanation in detail

**Table 47: Correlation among abilities of creativity-collaboration in the team**

<table>
<thead>
<tr>
<th>How satisfied are you by the 1-5 how creative would you say you are?</th>
<th>Pearson Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 47: Correlation among abilities of creativity-collaboration in the team
collaboration achieved among your team members? 

| Sig. (2-tailed) | N | ,006 | 16 |

Table 48: Correlation among abilities of daily tasks-matureness in the team

| On a scale 1-5 how mature do you believe that you are as a team? | Pearson Correlation | ,658** | Sig. (2-tailed) | ,006 | N | 16 |

Table 49: Correlation among abilities of workflow-collaboration and matureness in the team

| Do you feel that all team members follow the workflow of the team as expected? | Pearson Correlation | ,801** | Sig. (2-tailed) | ,000 | N | 16 |
| How satisfied are you by the collaboration achieved among your team members? | Pearson Correlation | ,588* | Sig. (2-tailed) | ,017 | N | 16 |

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

From the above, we find that most variables are positively correlated, suggesting that when the one variable is changed, the statistically significant associated variables will also be changed.
Conclusion

The purpose of this diploma thesis is to examine demographic and personal attributes of employees, the projects which they work with and the characteristics of their team to identify the factors that will help to recognize that a group is ready to make optimal use of Scrum methodology or ways with which we can help them to get there.

After comparing the profiles of people using the Scrum methodology now - with those who replaced it - with those who have used Agile but not Scrum and those who have no experience we identified the basic differences between these groups and by correlating these variables with the final performance / satisfaction / achievement of goals, we have reached to conclusions about which of these attributes promote team productivity and flexible working patterns and which create problems and delays.

The survey’s results reveal that while the demographic and personal characteristics of the members of a group (such as being creative with ability to explain in detail and remain focused on the objectives and daily obligations) may affect the results, 88% of the success depends on the composition of the group. By recognizing and respecting these different attributes as well as finding ways for optimal interaction between the team members. Concentrating on the dynamics of the group and the way they deal with the problems that occur is also important and can make a big difference.

We need to try to reduce dependencies from other groups by gathering the right people with the appropriate knowledge and skills within the group (autonomous, cross functional groups). This automatically means more effort and research for more optimal ways to achieve knowledge sharing even among people with different knowledge backgrounds and skillsets as well as knowing and understanding the bigger picture(ex roadmap)along with what we want from the group. The team must feel free, but not chaotic. Then it is important to try to create clear tasks in a way that all team members understand their complexity, to record unexpected tasks (frequency and complexity analysis) and also to attempt to come up with effective plans for the BAU(Automation, Forecasting).

We need to create a stable and predictable workflow and ensured that it is followed from all team members, but it is important to keep this as a “Live” document-frequently modified until it meets the group’s needs.
But the most important factor remains people. And we need to focus there:

I. We need to create a strong team. Develop team building activities. Eliminate the feeling of fear of change to support it and provide help to the team in order for them to mature together. The team's sense of responsibility for success or failure of projects must be developed and responsibility transferred from the managers or the individual people working on each piece of the project. Have a sense of stability and each member feels comfortable to express their opinion (even if the group is led into conflict first).

II. We must have a Product owner that respects the role of Scrum Master, feels responsible for its "product" and strives to develop a constructive relationship with stakeholders.

III. For the role of Scrum Master, we need people with social skills. They should be able to understand the needs of the group and support it without simply applying the theory. They should be able to teach the team the value of each meeting, and be open to dialogue and readjustment. The goal is that the team should not feel that there is not enough time for quality work and concentration as this will lead to negative feelings for the meetings in general, and ultimately for the methodology. They need to be able to lead the group, but should not behave like a manager or team leaders. They should earn the trust of the team and find ways to elicit honest and quality responses from retrospectives.

Finally, we should always keep in mind that the most important investment is people and to their satisfaction, development and collaboration we should mostly invest in.
« Always try to make the team feel, as a group of volunteers that work on something they are really passionate about »
Bibliography

English bibliography

Cockburn, A. (s.d.). Crystal Clear, A Human-Powered Methodology for Small Teams .
*Design Patterns and Refactoring*. (s.d.).


**Websites**


http://leadinganswers.typepad.com/photos/uncategorized/2007/06/28/1_lifecycle_2.jpg(figure 2)

(figure 3)

http://www.pmconsult.it/dynamic-system-development-method.html(figure 10)

**Greek bibliography**


Σταθακόπουλος, (2005). Μέθοδοι έρευνας αγοράς. Σταμούλη Α.Ε.
Appendix A : 1. Questionnaire :

https://docs.google.com/forms/d/1dMaUbppK8hPWG4vAOHKzPbeOrqUlqxE6g8kuMUivLSQ/edit