Challenges in the Transition from Waterfall to Scrum – a Case Study at Portbase

Sander Bannink
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
s.n.bannink@student.utwente.nl

ABSTRACT
An upcoming trend in software development is the Scrum framework. Old sequential methods like Waterfall have proven ineffective. Many companies are making the switch from Waterfall to Scrum and there are a number of challenges in this transition. In this paper, known challenges are researched based on existing literature and case-study research is performed. Known challenges are identified, previously unknown challenges might surface and advice is given on how to deal with these challenges. These results are relevant to other companies making the transition to Scrum.

Keywords
Waterfall, Scrum, agile, transition

1. INTRODUCTION
In recent years, agile methods are an upcoming trend in software engineering and project management. A survey among 2252 companies in 2007 showed that 26 percent of them were using agile methodologies. In 2009 a survey confirmed that Scrum was by far the most used agile methodology [19]. A recent survey of nearly 500 companies showed that “Scrum is the overwhelmingly preferred Agile method, used by 40% of respondents.” [2]

The reason for the broad adaption of Scrum is the fact that traditional methodologies like Waterfall are considered ineffective and inefficient. Waterfall resulted in large bureaucratic and slow development processes, focusing on creating documents and reports rather than working software [7]. Because of the long development cycle and the changing needs of the customer during this process, the software product that was delivered often did not meet the customer’s expectations. Requirements simply could not be changed during the process because it caused enormous delays. In fact, changing requirements often resulted in starting the whole project from scratch. There was need for a more incremental, non-bureaucratic method that focused on delivering value and reflecting business needs: Scrum seems to be best suited to do the job.

Scrum is seen as the ‘silver bullet’ for solving organizational problems, but actually it reveals dysfunctions in an organization. Solving them demands hard work and open minds [5]. Many companies are making the transition from a Waterfall based development process to the Scrum methodology. In chapter 2 the characteristics of Waterfall and Scrum are explained. The positive and negative aspects of both methodologies are summarized. This gives us an overview of the reasons to adopt Scrum. Chapter 3 gives the problem statement and the relevance of this work. Chapter 4 defines the research approach taken. In chapter 5 literature research is performed to identify already known challenges in the transition from Waterfall to Scrum. Chapter 6 describes the case-study at Portbase, a logistics information hub, who has made the transition from Waterfall to Scrum in 2012. The reasons for adopting Scrum, changes in the development process, the effectiveness of Scrum and lessons learned in the transition will be researched. The literature research helps at identifying potential problems. In chapter 7 the results will be discussed, comparing them to findings in literature and possibly identifying previously unknown challenges. These findings are relevant for other companies transitioning from Waterfall to Scrum.

2. WATERFALL AND SCRUM
To give a clear overview of the reasons to adopt Scrum, the theory behind both methods is explained and a comparison is made between Waterfall and Scrum in the following section.

2.1 Waterfall
In the history of the software industry, as software products were getting bigger there was need for better prediction and control over the output of larger software projects. This resulted in the sequential waterfall model [10].

The waterfall-method follows a sequential order of phases. In the first stage the requirements of the customer are determined and are fixed before the design is taking place. This is a strong characteristic of the model, it flows downwards.

Each stage consists of a set of activities and deliverables that need to be finished before moving to the next stage. Requirements are fixed upfront, the design is taking place and the implementation follows. Then the software is tested before it is delivered to the customer.

The advantages of the waterfall model are [12]:

- It’s sequential, easy to implement.
- Minimal resources are required to implement it.
- Proper documentation.

The Waterfall model reduces uncertainty by clarifying requirements and producing design documents early [18]. This makes the Waterfall-approach highly predictable. In some projects that are not subject to changes in requirements, this is the best approach on software development.

Back in 1970 the Waterfall method was founded by Royce [20] but he immediately admitted this method is risky and invites failure. Testing is taking place only at the end of the cycle and
may require a redesign of the system. This is so disruptive that either the requirements must be modified or a substantial change in the design is required. In fact, the development process needs to be completely re-done and causes a 100% overrun in schedule and/or costs. A more iterative-like approach was proposed to fix this issue but because of its simplicity the Waterfall-model gained ground in the industry, against Royce’s intention.

2.2 Pitfalls of the Waterfall model
The major reasons for failure of the waterfall approach to software development are [29] [33]:

- Requirements are not fully understood before the project begins.
- User know what they want only after they see an initial version of the software.
- Requirements change often during the software construction process.
- New tools and technologies make implementation strategies unpredictable.

Many projects have faced the above issues that made the waterfall model ineffective. This predictive model cannot cope with rapidly changing business environments. There was need for an adaptive approach on software development. Agile is an adaptive approach on software development. Most software development projects cannot be predicted and planned upfront and agile is a more flexible way to cope with ever changing business demands. It uses short cycles to add maximum value for the customer.

2.3 Scrum
The origin of Scrum goes back to 1986 [34]. The need for speed and flexibility in product development resulted in a different approach compared to the traditional sequential way. The old method went from phase to phase and the functions were specialized and segmented. With the ‘rugby’ approach the product development is done by a multidisciplinary self-organizing team whose members work together from start to finish [34]. This resulted in the Scrum-framework, which is named after the term ‘scrum’ in rugby.

Scrum is the most popular Agile method and is based on the Agile Manifesto [1]:

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

The values on the left are more important than on the right. These values compensate for the shortcomings of the waterfall model. “The Agile Software Development Methods have the potential to provide higher customer satisfaction, lower bug rates, shorter development cycles, and quicker adaptation to rapidly changing business requirements.” [4]

Scrum is a framework to organize people and deliver a quality product on time. “Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques.” [24]

2.3.1 Team
The Scrum team consists of a Product Owner, Scrum Master and a Development Team. Scrum teams are self-organizing and cross-functional, they are not directed by others outside the team and have all the competencies to accomplish their work. [24]

The **Product Owner** is responsible for maximizing the value of the product and the work of the Development Team. He is the only person responsible for managing the Product Backlog. This includes prioritizing items and making sure that the backlog is clear to all team members of the team [24].

The Product Owner interfaces with management and both internal and external stakeholders, who want to influence to the Product Backlog. The Product Owner has to be convinced, if a priority of an item needs to be changed, or a new item added. No one else but the Product Owner is allowed to change the priorities on items the Development Team is working on [23].

The **Scrum Master** ensures that the theory, practices and rules of Scrum are understood and enacted. For the Development Team he serves as a coach, removes impediments and facilitates Scrum events. For the Product Owner he helps to manage the product backlog and ensures that backlog items are clear. For the organization he serves as coach for Scrum adoption within the organization, planning Scrum implementations and working with other Scrum Masters to increase the effectiveness of Scrum in the organization [24].

The **Development Team** is responsible for delivering a potentially releasable product at the end of each Sprint. These self-organizing teams must be empowered by the organization. They are cross-functional, all the skills combined together must be sufficient to deliver a quality product. The whole team together is responsible for the product. The maximum size of a Scrum team is 9 members [24].

2.3.2 Events
All events in Scrum are time-boxed, they have a maximum duration. The length of a Sprint is fixed, other events may be shorter when the purpose of the event is achieved. The Scrum meetings are a formal opportunity to inspect and adopt something, they are essential to get a high level of transparency [24].

The **Sprint** is a time-box of four weeks or less during which a potentially releasable (working) product is created. Each Sprint has a specific goal on what is going to be built. A new sprint starts immediately after the previous Sprint. Only in exceptional cases the Product Owner can cancel the Sprint, but this is rarely done because of the complications that can occur and the short nature of Sprints. The Sprint consists of the Sprint Planning, Daily Scrums, development work, Sprint Review and Retrospective [24].

The **Sprint Planning** is the first meeting in a Sprint. In this meeting the work to be performed in the Sprint is planned. This plan is created by the team and is agreed upon. The Scrum Master ensures that the planning is realistic. The first topic is what has to be done in this Sprint. Objectives are discussed and the Product Backlog items that should be completed to achieve the Sprint goal. The Development Team must decide, based on their capacity, how many Backlog Items they can accomplish over the upcoming Sprint. The second topic is the way in which the chosen work is going to be done to turn this functionality into a potentially releasable product. The Backlog Items are decomposed in work items of a day or less and added to the Sprint Backlog. These work items should be fully understandable by the team members, possibly with the help of the Product Owner [24].

The **Daily Scrum** is a meeting of 15 minutes every day. Its purpose is to synchronize activities and create a plan for the next 24 hours. The Scrum Master inspects the work done since the last meeting and the work that has to be done before the next one. These meetings are held every day at the same time and place. Every team member explains:

- What work they have done since the last meeting;
The Definition of Done is a description of when a Backlog Item can be considered done and is different per Development Team. They have a shared understanding of when work is complete to ensure transparency [24].

According to case-study research in five companies the conclusion was that “SCRUM works in any environment and can scale into programming in the large” [28]. A case study [8] proves that Scrum can result in a high productivity increase in comparison with traditional methods.

Another advantage: Scrum cuts through project complexity and brings order from chaos by enabling a team to organize itself, which allows a particularly productive order to emerge [18]. However, there are some pitfalls in Scrum implementation which are stated below.

2.4 Pitfalls of Scrum

A few pitfalls of Scrum:

- Scrum does not provide adequate design documentation necessary for future development. It may not work well with projects that require high level of innovation because its focus is on bringing order to the development process [18].
- Quality assurance and testing need to be done in a different way compared to Waterfall, there are some challenges in implementing a new testing methods [26].
- Teamwork and dedication of team members is very important [12].
- Resistance to change is a major factor in the adoption of Scrum [19].

2.5 Hybrid

Another disadvantage of Scrum is the uncertainty, there are two competing priorities: “the need for certainty by following rigid and formal engineering processes, and the need to remain agile to deal with drift in requirements and uncertainty.” [18]

Sometimes the implementation of Scrum leads to a framework where some key parts are left out, given low priority or traditional approaches are combined with Scrum. This is referred to as ‘ScrumBut’ [21]. “Surveys show that 50% of Scrum teams worldwide cannot get software tested at the feature level by the end of a Sprint violating the second principle of the Agile Manifesto” [31]. A case study at two companies [31] showed that teams under-emphasized of failed to implement critical elements of the Scrum framework.

Water-Scrum-Fall is considered the reality in most agile organizations today [36]. This results in failure to realize the business benefits of Agile: faster time-to-market, increased business value, and improved flexibility and responsiveness. Water defines the upfront work that is required by governance rules. It includes project planning and budget. Requirements are specified. Teams use Scrum to develop software which requires frequent software releases to get feedback. “However, most organizations do not have the architecture required to support dynamic, flexible releases; instead, they do infrequent releases backed by heavy process and governance.” This is the Fall part; the use of agile processes will not change the enterprise architecture [36].

3. PROBLEM STATEMENT

In 2012 Portbase switched its software development methodology over from traditional Waterfall to Scrum. One year down this road, Portbase is open for a study into the results of this major transition and to compare the outcomes with literature.
This paper aims at identifying potential problems in the transition from Waterfall to Scrum. A case study at Portbase will provide empirical findings, that can be compared with existing literature.

3.1 Relevance to the field
This study will be relevant for other companies making the transition from Waterfall to Scrum. Known challenges will be identified in existing literature and new challenges might be found.

3.2 Relevance to science
A recent study [19] states that there is a need for more empirical studies on agile development. Human and social factors in the adoption of Scrum are under-researched. The effects that arise from the change to agile methodologies should be investigated. [19].

This study might identify other possible challenges that are not previously identified.

3.3 Main research question
What are the possible challenges in the transition from Waterfall to Scrum?

3.4 Detailed research questions
- What is known in literature about the challenges in the transition from waterfall to Scrum?
- What challenges do we observe at Portbase?
- What generalizations can be drawn from the findings at Portbase?

4. RESEARCH METHOD
Literature research is done concerning the known challenges that can occur during the transition from a Waterfall-based development cycle to Scrum. This literature search and the resulting framework are performed using the ‘Grounded Theory’ method as described in [35]. Literature on the transition from Waterfall to Scrum are searched for and relevant references are investigated. Then the challenges are categorized according to different concepts that are found in these literature. This results into a framework of known challenges. This framework was the foundation of case-study research that was performed at Portbase to identify possible challenges in the adoption of Scrum. Semi-structured interviews were held with the Managing Director and members of different Scrum-teams with different roles. This gives a good overview of their transition to Scrum and the challenges they experienced. The findings are then reflected on the related work and turned into advice on how to improve the development process and organization. After that, these findings are generalized into advice for other companies willing to switch from Waterfall to Scrum.

5. LITERATURE REVIEW
According to several studies, the following known problems in the implementation of Scrum were identified and summarized:

5.1 Changing role of managers
In the traditional waterfall approach, managers were in control: they decided what work is being done and when. In Scrum there exists a lot of autonomy for the team. The manager’s role is completely different, managers have to learn how to behave differently in a Scrum environment [5]. “One of the problems is that managers have to understand the difference of managing people, as opposed to leading people. In Scrum the manager needs to listen to the team and help them remove impediments” [5]. Changing the mindset of project managers takes some time [27].

In Scrum this manager is called the Scrum Master. The task of the Master is leading the team, and solve problems (impediments) immediately. This task is very important because “If a defect or a problem is not addressed immediately after it is identified, rework will accumulate and it will be difficult to deliver a sprint with high quality and maintain a high velocity” [8].

Another role of the Scrum Master is to enforce the constraints of Scrum and improve communication between team members [30]. If this is not done properly and the team has too much freedom, the team might slowly drift back to ‘the old ways’ [14].

“The self-organized team nature of Scrum immediately surfaces negative outcomes from lack of trust, fear of conflict, lack of commitment and accountability, and inattention to results. The Scrum Master must identify and clarify these impediments and then work with the team and Management to remove them.” [30]

The transition to Scrum needs full support of upper management. Not having the full support of upper management is a great challenge. Lack of resources and training can cause unmotivated employees. Scrum creates less documentation so upper management might insist to create more documents to enforce the command and control structure they used in the old methods. This conflicts with one of the core values of Agile, less documentation and can cause extra work pressure in Scrum teams [6].

Managers are used to command and control. Team members in Scrum are able to figure out how to do their work, if they get instructions they aren’t free to do their work the best way possible. This is a significant change in the manager role [22]. This problem of the Scrum Master controlling and pushing instead of letting the team organize itself is also stated in [14].

5.2 Team autonomy
There are three types of autonomy in Scrum teams: external autonomy, internal autonomy and individual autonomy. External autonomy is the influence from outside the team on the team’s activities, internal autonomy refers to the degree in which the team members share decision authority and individual autonomy is the amount of freedom an individual has in completing his tasks [9] [17].

Internal autonomy is the most important factor. The team as a whole is responsible for the outcome of the Sprint. Team members need to be committed to the project and their tasks, there must be a high level of transparency about their activities, openness regarding impediments and daily meetings need to be taken seriously. If any of these are lacking, there is a chance that a high level of individual autonomy will occur. In that case everybody is focusing on their own work and problems are not reported, resulting in a dysfunctional team [17]. This issue of overindividualism is also stated in [14] and [30]. Team meetings should be respected and a penalty for absence can be enforced [31].

“For Scrum to work, you must have trust, openness to conflict, commitment, accountability, and attention to results on a team” [30]. Sometimes team members are individualistic because of a lack of cross-training. Team members in Scrum must be multi-skilled, a high level of specialization is not desirable in Scrum [14] [16] [30]. Specialization is also identified as main reason for problems regarding self-management. Developers used to be working independently and focusing on their own work, not on the team process. Lack of commitment results in team members giving priority to individual goals instead of team goals.
This is caused by specialization: team members don’t want to commit to other’s work and decision-making is difficult [16]. If developers are committed to multiple projects, the goals or needs can be in conflict. This has a negative impact on self-management of teams [16].

Redundancy, also known as backup behavior, is the ability of the team members to do each other’s work when needed. In practice this requires the team members to know what others are doing and having complementary skills [9]. A high level of specialization results in a lack of redundancy and thus reduces the flexibility, resulting in high individual autonomy [17].“Redundancy has been identified as an important prerequisite for self-organization” [17].

Daily Scrum meetings improve communication between team members so they are very important. If there is no communication between teams, it might be possible that duplicate work is done. Communication with the customer to get frequent feedback is also very important. Ideally, the whole Scrum-team should be located in an open-space-working-environment [4].

Concerning external autonomy: the sprint backlog is fixed, managers cannot ask the team to change direction immediately [5]. New ideas are only added to the sprint backlog if they are worth it and they will not interrupt the flow of the team [5]. Otherwise they will be moved to the backlog (for future sprints).

### 5.3 Clear backlogs

A factor that influences productivity in a negative way is the lack of a clear backlog. Developers need to fully understand what they are expected to do (what the customer wants). There needs to be a focus on preparing work to ensure that backlogs are complete and clear. To check if a backlog is complete, a checklist could be used [8]. Improperly formed Product Backlog content should be rejected [31]. “User stories should be complete, have acceptance criteria, and the usability tested before added to a sprint” [15]. User stories need to be formed properly, prioritized and ready before allowing them into a sprint [30].

If the developer doesn’t understand the purpose of what he is designing, it will seriously affect his productivity. It’s considered very important to have close contact with the customer. The backlog-items need to be in priority order, on top are the items that add the most value. There is always a risk that Product Owner or stakeholders assume that they own the backlog and prioritize backlog items that are less important considering the whole product or the business [13].

Most of the time customers do not know what they want, resulting in unclear requirements [4]. It is important to have a high communication bandwidth with the customer. Doing the project that provides the most value for the customer and provide solutions to real need benefits dedication and productivity [32].

The definition of work items should be consistent. Estimated and real time for completion of a task should be tracked to get better estimates in the future [11]. A physical Scrum Board gives a good overview of the status of a Sprint, with columns like Product Backlog, Sprint Backlog, Work in Progress and Done [31].

### 5.4 Coaching / learning

The importance of coaching is often underestimated. “Agile master or Agile coach is an essential role during agile adopting process in any organization” [6]. If a developer doesn’t know the constraints of Scrum, there is a chance of falling back to old methods [30]. A clear understanding of Scrum is essential, and this should be focused on in the implementation of Scrum. An experienced coach should be in place during the transition to propagate the core values of Scrum and to respond on questions about the process. Usually this coach has the Scrum Master role.

According to a survey [5] among Scrum Masters, the following advice is given: “Take your time, it takes time to get used to Scrum, it won’t all change overnight” and “Prepare for constant learning and do not read the manuals like a bible”. This emphasizes that every team is different and every team is implementing Scrum in a different way.

A case-study [6] also states that changing project management and programming habits is difficult and takes time. It comes with training and commitment to change. Another case-study [15] emphasizes the importance of training Product Owners and getting outside coaches involved early. The transition needs guidance of an agile coach [27].

Over-enthusiasm for agile and fast adoption can lead to problems. Teams often think Agile can solve their problems immediately. A drop in productivity can occur when adopting agile methods takes more time than expected. This can cause the team to be less optimistic and fall back to old Waterfall methods to meet the deadlines [6] [13].

Another important factor is feedback inside the team. Team members need to give continuous feedback to each other so they can improve their way of working. This requires openness and transparency [3] [13]. Often problems are not reported because team-members perceive the problem as personal. Another reason is lack of trust in the team or Scrum Master. Also, if problems are not handled, the team could stop reporting them [16].

This continuous learning process takes time and if work pressure is high this has a negative impact on learning [3].

### 5.5 Productivity

Each sprint in Scrum is 2 to 4 weeks. During this sprint, the teams should stick to the sprint backlog. Only in exceptional cases (when productivity is not affected) work can be added to the backlog. During sprints there is a total inflexibility in what to work on, during the planning meeting backlogs are added and the sprint backlog is fixed. This sometimes causes a perception of inflexibility, because direction cannot be changed immediately. Managers thus could be skeptical about this inflexible way of working during sprints [5], but it is one of the most important constraints of Scrum. Not respecting this will cause a drop in productivity.

Measurements of productivity should be carefully chosen, it influences the way people work [27].

Over-commitment is a bad habit. If someone is pressured to commit to an outcome that is not realistic, this can cause serious problems in productivity. Work pressure is high and quality might be at risk [22]. On the other side, if maintenance and bug fixing is taking too much time the productivity might also be at risk [14].

### 5.6 Resistance to change

A study on the acceptance of Scrum [19] shows that resistance to change is a major factor in de adoption of Scrum. This also comes forward in another study [17]: “The problems are mainly caused by resistance to, or over-enthusiasm for, agile practices within a software development team”. The abrupt change in the development structure can cause resistance [25].

Testing is an important factor in Scrum, so testers might resist because of the extra work that has to be done. Managers might
feel uncomfortable with less documentation. Costs and planning cannot be foreseen in detail because requirements can always change. Projects are open-ended so it is harder to accept this technique as it creates more uncertainty [25].

In a waterfall-based environment the introduction of Scrum might feel risky. It creates a feeling of uncertainty compared to waterfall, requirements are not fully stated upfront. Functional specialization is a waterfall habit but in Scrum members need to be cross-functional. These differences can create resistance to the Scrum-method [22].

5.7 Quality and testing
The goal of short iterations is to create the most business value for the customer. This seems to be in conflict with the need for long-term quality. Quality requirements are often forgotten in the need of keeping the time schedule. If the Definition of Done is not strictly enforced, there will be quality problems because for example testing is not done properly [16].

Testing in agile should be done continuously. The developers should design unit tests while coding, this is critical in agile. Automated build and deploy systems are a critical part of agile and should be used [26]. Practices like test driven development, pair programming and continuous integration are difficult to implement [27].

Unhealthy technical debt is a trick used by developers to solve a problem to the prejudice of quality [11]. Defects in software delay the release, make it difficult to maintain and rigid in the face of changing business needs. This is caused by a bad estimation of the time needed to complete work items.

Over-commitment can lead to bad quality. If the completing product backlog is taking more time than expected, teams might drop testing and refactoring [22] or it can cause overtime which is not permitted [14].

The lack of documentation also can be an issue. Often work needs to be handed over to another developer. In that case the documentation needs to be clear, otherwise it will raise questions which negatively impact productivity [4].

6. CASE-STUDY
Case-study research is performed at Portbase. Section 6.1 gives a short introduction to Portbase, more information about Portbase is gathered during the interviews.

The main topics of this case-study are:
- What were the reasons for Portbase to adopt Scrum?
- In which way Scrum has changed the development process at Portbase?
- Were the perceived objectives achieved?
- What lessons can be learned from the transition at Portbase from Waterfall to Scrum?
- What improvements can be made to the process at Portbase?

The aim of this case-study is the identification of challenges in the transition to Scrum at Portbase on the basis of the framework resulting from literature research. Semi-structured interviews are held with the Managing Director (COO), Strategy and Business Development Consultant (StrategyConsultant) who used to be a Project Manager and is working here for 10 years, a Product Owner (ProductOwner) who also used to be a Project Manager and is working here for 10 years, two Scrum Masters (ScrumMaster1, ScrumMaster2) who are externally hired for the introduction of Scrum one year ago and have much experience with Scrum, and two Developers: Developer1 has been employed for 7 years and Developer2 who was externally hired in 2004 and now internal for 6 years.

6.1 Context
Portbase is the neutral and reliable hub for all logistics information through the Dutch ports. Via Portbase’s port-transcending Port Community System, companies can benefit from a multitude of intelligent services for simple and efficient information exchange, both between companies and between the public and private sector. This enables all the participants to optimize their logistics processes, thereby improving their own competitive position and that of the ports. Portbase belongs to and serves the port community and is a non-profit organization.

6.2 What were the reasons for Portbase to adopt Scrum?
The first interview with the COO gives us some background information of the context in which this case-study is performed:

Portbase has a customer base of 2300 companies with 10.000 end-users. The system provides 42 different services. There was pressure to reduce the time-to-market of their software. In the past their development process was partly Scrum but due to the developing complexity there was a sequential approach to software development. The system serves as a logistics information hub of the ports of Rotterdam and Amsterdam. There is an existing system that was designed for Rotterdam and a new system on a new platform which uses SOA (Service-oriented architecture) to come to a modular system which focuses on reuse of functionality. To get the alignment between business and IT right, Scrum was needed. The sequential waterfall model gives the perception of control. Design takes too much time and communication between different phases is poor. They brought the analysts to the business side and with Scrum they reduced the specification upfront. In this way they could respond quickly on changes in the business environment. Team members have a shared responsibility for the outcome and customer satisfaction goes up due to the fact that they are seeing a working product more often, based on their needs.

6.3 In which way Scrum has changed the development process at Portbase?
All teams in the development process have made the transition to Scrum in a time span of one to two months. For the past years there was a transition to professionalization and SOA. The upcoming years there will be a focus on product innovation and intelligent use of data. The new ‘Maasvlakte 2’ area in the harbor of Rotterdam causes lots of work. Maasvlakte 2 uses a new way in which for example containers are not stacked but are directly delivered to their destination (e.g. truck). This requires a whole new system design which also has to comply to specifications of the government concerning customs. The use of IT in logistics requires a whole new way of working and plays an important role in the future. Currently the need is clearly transaction-driven, but in the future there will be more intelligent data-analysis in for example maps, forecasting, flow of goods, historical data, measurement of throughput time, track and trace and planning with all the data combined. This requires openness of participating companies which can be an issue.

The way in which Scrum has changed the development process according to the COO: “there is less specification upfront and more during the process. The old situation caused strong deviations and there was less teamwork. There is better communication with the customers, the business and IT are together responsible for the outcome. Whether or not the velocity of development has increased is hard to say. Nevertheless there are
at this point enough advantages to be satisfied with the transition to scrum.”

StrategyConsultant states the old Waterfall-situation caused late delivery, high costs and not meeting customer demands. The users are more satisfied because of many demo’s and feedback.

Developer1 says that time estimation is better because of short sprints and there are less deadlines. Developer2 sees the transition to Scrum as an improvement: “every two weeks there is a new product and the organization is far more involved. Customer contact is better: in the old situation we could build for six months, without contact with the customer, and eventually deliver something the customer doesn’t want. In the new situation there is more feedback.” This confirms the view of the COO and StrategyConsultant.

6.4 Were the perceived objectives achieved?

According to the COO: “The transition to scrum takes time, demands full support and needs to be implemented carefully and consequently. It is a learning process which requires a whole new mindset. The old situation was based on control rather than the new teamwork-oriented approach of Scrum. This requires also an organizational change. Only the software development process has switched to Scrum, the intention is to get the whole organization over to scrum. Scrums get the alignment between the business and IT right, by reacting quick on changing environments. Design upfront takes less time and there is a shared responsibility for the outcome. The quality is higher and customers are more satisfied because they are more involved in the process, also they see results and provide feedback more often.”

The design upfront is traditional according to ScrumMaster2. The Product Owner gets the demands of the customer, then a Business Analyst makes specifications in a traditional Waterfall way. Then this is placed on the team’s Backlog. This specification upfront is needed because not all the specification can be done within a Sprint. It’s better to have a Business Analyst within a team and work in an agile way, right now the Business Analyst is placed in the team.

ProductOwner confirms that not the whole organization is working agile. The releases for example are time-intensive. Dependencies in the whole system have to be tested before it can go to production. This takes about 4 to 6 weeks. There are several big releases, teams that are ready on time can participate in the release. The release of patches should go faster, for example every week. StrategyConsultant, ScrumMaster1 and Developer2 state that the system is not fully modular. There are many dependencies and this is the reason that releases are taking so long. The connection between the Scrum-teams and the pre-design and releases is bad. Releases are four times a year and very big, you cannot release after every sprint but have to wait for the big release. ScrumMaster1 says that it is difficult to explain to the customer why releases are taking so long. They can see the results of their feedback in production only after the next release.

The phenomenon described above is a typical water-Scrum-fall situation: requirements and planning take place upfront, once the management is happy with the plans the development is started. The team delivers frequently but the release process runs different cadence, picking up the software at the intervals the release plan defines [36].

The quick response to changing environments is good from within the Scrum-teams, but releasing these changes takes a lot of time and is not agile. The design upfront is done by Business Analyst and follows a tradition approach. This contradicts the objectives the COO wanted to achieve.

6.5 What lessons can be learned from the transition at Portbase from Waterfall to Scrum?

In this section the findings in the interviews are reflected on the findings in the literature review. Every challenge that’s identified or other findings related to the framework are mentioned.

6.5.1 Changing role of managers

The Scrum Master has a facilitating role, instead of the steering role of a Project Manager in Waterfall. The COO states that there is less control in Scrum, budgeting and planning is more difficult with Scrum. ScrumMaster1 calls this leading from within the team. The team is in charge, they make their own decisions. ScrumMaster2 states that traditional Project Managers cannot function as Scrum Master.

The task of the Scrum Master is to remove impediments, according to Developer1. Because of a budget decision they didn’t have a Scrum Master in the past months. This task was given to the team but this didn’t work out, impediments were not solved and Scrum-constraints were not enforced. ScrumMaster1 confirms this and says the retrospective-outcomes also were not taken action on. Also organization-wide impediments are not properly taken action on. The organization doesn’t respond properly to these impediments.

Documentation-pressure is not an issue according to ScrumMaster1: “there is no pressure from the upper management to give insight or more documentation. This is arranged in the form of an acceptance sheet upfront.”

StrategyConsultant: “There still exists a layer between the Scrum-teams and the board of directors: the Project Board. This Project Board is a Prince2-structure that still exists and produces reports for the board of directors. This layer is essential to escalate if there are major problems. The project board delegates leading the team and the reporting to the project manager. The question is if you need a Project Manager and a Scrum Master. Or can they be one person? The Scrum Master can do the kind of reporting that is required for the Project Board.” This indicates that there are still command and control structures in the organization.

6.5.2 Team autonomy

The COO states it’s all about trust in the team. You have to be confident about the discipline and qualities of the team. “People are becoming more assertive because of Scrum. They have a shared responsibility for the outcome.” According to Developer1, ScrumMaster1 and ScrumMaster2 there is enough transparency and openness during daily meetings and everyone is attending.

The COO says there exists a lot of specialization in the teams, which causes bad redundancy. “Shared responsibility requires knowledge sharing, resulting in better redundancy. The new platform makes it difficult to gain knowledge, a lot of knowledge is not in-house and needs to be acquired via external engineers. The goal is to get this knowledge in-house and get homogeneous teams who have all the knowledge required to build the system. The next step is for each developer to document their knowledge and way of working to a standard format, which can be shared with other developers.” This is confirmed by the view of Developer1, ProductOwner and ScrumMaster1 who also stress the importance of knowledge sharing.

ScrumMaster1: “Taking over each other’s tasks is not needed, you need a multidisciplinary team that contains all the skills to
make the project a success. The knowledge-profile is a T-model: one part is your specialization, the remaining parts you know something about but not everything. For example a developer doesn’t need to do testing but needs to understand the way in which testers work so they can optimize and help with each other’s work.” StrategyConsultant: “In the past there were people going in and out of teams, but this didn’t work. Now we have fixed teams that are used to work with each other. They always have the same Scrum Master. The Product Owner is linked to a project backlog. Sometimes the team is put on another project but the team itself stays intact.”

The communication between teams is facilitated with the Scrum-of-Scrums. Developer1 confirms that all teams are colocated.

6.5.3 Clear backlogs
Because of the pre-specification by the Business Analyst the definition of backlog-items should be clear, Developer1 confirms that the backlog items are always clear.

ProductOwner: “There are 2300 companies which are divided in user groups. Each user group consists of representatives of that sector. Because of its expertise in this sector Portbase gives a big proposal of what the system should look like. Customers can then discuss their own ideas about it and these can be processed. It is difficult to discuss about different ideas, you cannot satisfy every demand. There is also regular contact with the government about information standards.”

6.5.4 Coaching / learning
Every Scrum master is externally hired and function as a coach for the teams. The COO has the intention to coach own employees to become Scrum master. The coaching of the teams and Scrum masters takes time. ScrumMaster2 says: “For most organizations the implementation of Scrum has a fixed budget, if they run out of budget the Scrum-coaches leave and they might fall back to old methods.” This stresses the importance of getting Scrum Masters in-house. ScrumMaster1: “The goal of this organization is to eventually have Scrum Masters in-house. At this time experienced Scrum Masters are needed who can also advise management. The whole organizational change needs to be given more attention.”

Developer1, ProductOwner and StrategyConsultant also say the rest of the organization has to get used to the agile approach of software development. At this moment they are used to the old command and control structure.

ScrumMaster2 sees a kind of mini-Waterfall: developer finishes the code before the tester is going to test, but ideally they should work together.

Developer1 in this case was used to the empowerment of the team, the other developer was expecting more of a steering role of the Scrum Master. This indicates there is a difference in their view of Scrum. Apparently the values of Scrum were not properly explained by the coach.

6.5.5 Productivity
COO: “You never know when the product is going to be finished. The Product Owner keeps an eye on priorities but the productivity is unknown. Planning is difficult but due to the short sprint there is more insight in productivity and deviation from the desired output.”

Developer1 says sometimes a Sprint failed because of lack of technical knowledge or downtime of the development environment, productivity was low when a specialist was leaving the team. Developer2 states that their Product Owner used to be a Project Manager and was pushing the team to deliver maximum value. This resulted in bad testing and failed Sprints because of technical debt.

Developer1: “Estimations of the time required for a backlog item are good because of the short Sprints. If a backlog item is taking more or less time this is corrected at the Scrum board but not recorded in a system. There is no burn-down chart, using a burn-down chart might be better for motivation.” Userstory-points for backlog items are registered in a system, according to ScrumMaster2. If the Sprint is finished, deviations from these predictions are not registered, this could be helpful to prevent over-commitment. ProductOwner: “Estimations are done using poker planning. At the end of the Sprint you can exactly see what your team’s velocity is.” StrategyConsultant states the velocity is important in making price quotes.

6.5.6 Resistance to change
COO: “Managers have a completely different role. There is less insight for the upper management and shareholders. Less control and predictability.” StrategyConsultant: “The COO wants to have certainty about what it’s going to cost and what the result will be, this is uncertain in Scrum. The final product is uncertain.” ScrumMaster2: “Management now finds out that the transition to Scrum also has consequences for them. Teams are becoming mature and are expecting more support from the management. They need to organize themselves in an agile way, if organization impediments are not solved this is not accepted by the teams. We are in the middle of an organizational change from traditional to agile. Right now only the bottom layer is agile but the rest of the organization also has to become agile.” Developer1: “There is no pressure from the management. The team is autonomous and there was no resistance to Scrum from within the team.”

In the beginning there was some resistance to Scrum. ScrumMaster1 started with a Retrospective to see what they already knew of Scrum and the way they want to work themselves and with the team. ScrumMaster1 states that he made daily standups voluntarily because of the initial resistance. If team members didn’t see these meetings as helpful they shouldn’t come. In the end team members were always attending these meetings because they saw the added value of it.

6.5.7 Quality and testing
The COO about quality: “At this time productivity is the top priority in teams, but they have to take into account the importance of quality. Automatic testing is very important.” Developer1: “In the past there were some shortcuts to meet the deadlines, but at this time testing is going well.” Developer2 experiences technical debt sometimes. “The team is pushed to deliver as much functionality as possible and this is accepted by the team itself. Then testing is not done properly because it is underestimated. At the end you have some delay in delivering the code and then the testing is under pressure.” ScrumMaster1: “There is always need for more functionality but you have to suppress that sometimes. If development is taking more time, the testing/quality can be at risk. At that time some functionality is dropped and moved to the next sprint. Shortcuts are taken seldom. If you want a high productivity, the quality also needs to be good. Otherwise you have the risk that defects are coming back, which is causing extra work.”

Developer2: “The Definition of Done has low priority at this time. The current DoD is unrealistic an long, this is the reason that it is not used. The only thing we do is working to the delivery date and bringing the software in production.” ScrumMaster2 says the Definition of Done is organization-wide. “It was composed out of all the Definitions of Done that existed in the organization which is causing a long list of demands. But the team needs the ownership of the DoD, otherwise they won’t
be committed to the DoD. One list for every team doesn’t work because of the differences between teams. Within a team the DoD expands but eventually some points can be removed because it is in the daily routine of the team. The assumption was that every team should work in the same way so they needed the same DoD, but it doesn’t work in this way. The teams need to be trusted and formulate their own DoD.”

Developer1: “There are two testers in each team. Developers build the unit or technical tests, testers focus on functionality. At the beginning of the sprint a test plan with scenarios is made, if code is delivered it is tested automatically.”

7. CONCLUSION AND FUTURE WORK

The challenges identified in this case-study are compared to findings in literature. Advice is given on how to deal with these challenges. Previously unknown challenges are identified and make a contribution to science.

7.1 Conclusion and recommendations

The transition from Waterfall to Scrum is not easy, as is confirmed by literature research. It comes with commitment to the values of Scrum and requires a mindset of all employees within an organization. The team is becoming the central part from within multi-skilled professionals work together to create a product based on customer needs.

The main challenges in this transition are the facilitating role of managers, the absence of a command and control structure, empowerment of the self-steering team, providing value to customer needs, understanding Scrum values, resistance to change and quality of the product.

In this case study, research was done concerning the identification of challenges at Portbase. These were compared to literature and resulted in new insights on this transition. The main findings are the organizational change required and the mindset required by the team members.

In this case the development teams are using Scrum but the rest of the organization doesn’t yet work in an agile way. There still exists a command and control structure with a layer of Prince2 Project Boards between the teams and the upper management. There needs to be a more facilitating and supporting role for the Project Board to the team, for example to remove organization wide impediments. There also exists a Project Manager who is in contact with the Scrum Master of the teams but it might be possible that the Scrum Master can take over his tasks. The pre-specification is done in a traditional way by Business Analysts before it’s added to the Product Backlog. Placing the Business Analysts in the team would solve the efficiency problem, because they are then working in an agile way. After the pre-specification the team delivers functionality every Sprint, but after that the release process is traditional. There are several big releases per year but a release takes about two months of testing. This slow release process exists because of the dependencies between the 42 different services, the complete system has to be tested before it can go to production. This violates the Service Oriented Architecture (SOA) principle of modularity which is an integral part of the Tibco platform Portbase has been deploying over the last two years. These services should be independent. Only in this way the time-to-market of the software can be reduced, because most of the testing can then be done within the Scrum teams.

The mindset within the team is also an issue. Managers need to behave differently in a Scrum team, instead of managing they should be facilitating the team. Monitoring the process and removing impediments are the main tasks of the Scrum Master. This company hired external Scrum Masters who also function as coaches for the team. The desire is to eventually have Scrum Masters in-house. This should be given high priority, because if experienced Scrum Masters leave there should be potential Scrum Masters in-house to take over their tasks. The absence of a good Scrum Master results in a bad functioning team where the process is not monitored and impediments are not removed. Knowledge-sharing within and between teams should be facilitated with regular meetings. These meetings should be cross-functional to share each other’s working habits. Standard ways of working should be established and documented to cope with changes in human resources, knowledge needs to be preserved. In this way multi-skilled teams can emerge, who can work together in a better way. Story-points for backlog items are registered in a system to get better estimates but deviations from these points are not registered at the end of the Sprint. Reasons why some backlog items are taking more or less than the estimates should be registered and analyzed to get better estimates in the future. Over-commitment leads to technical debt and should be prevented. Pushing to deliver more functionality for the customer at the cost of quality is a bad habit. An organization wide Definition of Done is not functioning well. The list is too long and is a combined list of all DoD within the organization. This gives the perception of quality control but has no commitment because each team is functioning in a different way. Each team should formulate their own Definition of Done.

Another interesting observation is that 5 of the 7 Scrum-teams are externally hired. This comes with a risk because if a project is finished or there is a change in employees within teams, knowledge flows out of the organization. This is remarkable and also emphasizes the importance of knowledge sharing.

7.2 Future work

Portbase needs to focus on the organizational changes required to support the Scrum teams. Only then the full benefits of agile are realized. In this case study we have looked at Portbase, but in which way we can generalize these findings? According to Forrester Research this so-called ‘Water-Scrum-Fall’ is the reality for most organizations making the transition to agile [36]. More scientific research should be done concerning the organizational changes required to support Scrump teams and realize the full benefits of agile.

Another remarkable point is about the Service Oriented Architecture (SOA). In this case-study SOA was the main principle in the architecture of the system but the SOA-value of modularity was not implemented in the right way. There were dependencies between the different services and more research should be done on how to reduce these dependencies.

Traditional development teams consisted of several specialists which had their own expertise. One of the most important challenges in the transition to agile is to develop multi-skilled teams. Literature states that every team member should be able to take over the task of another team member but this case study shows this is not realistic in practice. Each team member can develop and share their own specialization but also needs to know somewhat of the other’s work to support and adopt to their way of working. How this knowledge sharing must be facilitated should be researched more.

Out of the scope of this research and highly recommended for future research is customer satisfaction, quality of the software and speed of development. The success of agile practices can be quantified in this way.

8. REFERENCES
