Continuous Improvement in Agile Development Practice

Marta Kristín Lárusdóttir, Åsa Cajander, Michael Simader

Continuous Improvement in Agile Development Practice

The Case of Value and Non-Value Adding Activities

Marta Kristín Lárusdóttir[1], Åsa Cajander[2], Michael Simader[3]

[1] Reykjavik University, Menntavegur 1, Reykjavik, Iceland
marta@ru.is
[2] Uppsala University, Lägerhyddsvägen 2, 751 05 Uppsala, Sweden
Asa.Cajander@it.uu.se
[3] Celum America Inc, 70 West Madison St., Suite, 1447, Chicago, IL 60602, USA
michael@simader.me

Abstract. Agile development has positive attitudes towards continuously improving work practices of IT professionals and the quality of the software. This study focuses on value adding activities such as user involvement and gathering metrics and non-value adding activities, such as correcting defects. Interviews were conducted with 10 IT professionals working with agile development in Iceland. Results show that IT professionals emphasise communication with users both through direct contact and using email, but they rarely use metrics to make improvements measurable. The most serious non-value adding activities are: partially done work, delays and defects. The core reason is that long lists of defects in the projects exist, which means that the software is partially done and the defects cause delays in the process. There are efforts to reduce non-value adding activities in the process, but IT professionals are still confronted with problems attributed to miscommunication and the impediments by the external environment.

1 Introduction

Software development is a complex task related to human, social and organizational factors, as well as technical factors [1]. It requires both plans as well as situated action [2] to be successful, and far too often the software built is hard to use [3]. There are several competing normative processes describing how software development should be done and these processes have changed as time goes by. In the 1990‘s more emphasis in software development was on delivering parts of the software to customers iteratively and incrementally than before, so time to market would be shorter [4]. Additionally, the software was getting more interactive with emergent requirements. In incremental software development the software requirements are divided into parts, which are implemented and a deliverable version of that part of the software is made [5]. The basic idea is that the IT professionals use the knowledge they gained in previous increments to improve the software being developed, but in the iterative development not much emphasis is on improving the process of the development.

adfa, p. 1, 2011.

Agile software development has recently emerged as a popular iterative software development process and it focuses on communication and independent teamwork, as is stated in the principles behind the Agile manifesto [6]: "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done". Moreover, it is stated in the manifesto that continuous improvement of the process is recommended, as in this principle: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." In the agile software development process Scrum this is done at the retrospective meetings, where the whole team discuss how to improve their work processes for the next weeks.

The practice of continuous improvement has been not much studied in research even though software developer's work with continuous improvement reveals what practitioners see as important in their practice and what kind of problems they address. Such research would also help researchers understand practitioners understanding of usability work as only one aspect of continuous improvement in software development. Hence, this paper describes an interview study identifying how continuous improvement is done in agile software development. We have analysed how the IT professionals involve customers and users to improve the software under development, what activities the IT professionals conduct for continuously improve their software development process and what measurements are made to measure the status of the improvements. Additionally we have analysed what non-value adding activities they describe according to seven categories of waste defined in Lean software development according to Poppendieck and Poppendieck [7]. Finally we discuss the main results from the interviews on continuous improvement in agile software development.

2 Background

In this section we first present agile software development (hereafter called Agile), and continuous improvement in Lean management as this has been used in order to elucidate the work with continuous improvement in Agile. We have chosen the concept waste from Lean since Agile has its roots in this process.

2.1 Agile Software Development

To improve the process of software development many processes have been suggested by IT professionals and researchers in the area [1]. In 2001 the Agile Manifesto was written by advocates of software development processes like eXtreme Programming, Scrum, Crystal and Feature Driven Development. Through the manifesto the term Agile Software Development was born [8]. In short the manifesto includes the following four key values: 1) Individuals and interactions over processes and tools; 2) Working software over comprehensive documentation; 3) Customer collaboration over contract negotiation and 4) Responding to change over following a plan.

Moreover, individuals and interpersonal communication between stakeholders are in the center of attention and the agile software development processes are iterative and incremental, hence it is possible to adapt to a changing environment and circumstances [6].

2.2 Continuous Improvement: Value Adding Activities

Lean management (hereafter referred to as Lean) is a holistic approach aiming at providing the right product in the correct amount of time at the right place with the right quality [9]. Lean is not a process but can be seen as as a set of principles [10]. In Lean management the focus is on continuously improving the process of working by defining and improving value adding activities such as customer involvement and minimising non-value activities called waste [11] in order for projects to become more efficient.

The main focus of customer involvement is to add value to the customers through making the software more usable for its users [12]. In theory customers or users could be involved in four human centred design activities: When understanding and specifying the context of use for the software, when specify the user requirements, when produce design solutions to meet these requirements and when evaluating the designs against requirements [12]. In practice customers are often involved when evaluating design solutions, but it is not as common to involve users when specifying requirements [13].

Typically thorough usability evaluation is conducted as seldom as twice a year, often by contracting an external usability expert despite of being highly rated by IT professionals as a value adding activity [14]. The main reason why thorough user evaluation is not conducted is lack of time in the Agile projects [14,15]. However, some user involvement activities, such as workshops, are typically used at least twice a year in Agile projects [14]. These activities are informal and therefore these fit better to the fundamental principles of Agile, which are speed and communication. Additionally producing incremental deliverables in short project periods is another popular and important Agile feature. One challenge for IT professionals working in Agile projects was maintaining the overall vision of the user perspective, despite of the Scrum tradition of slicing projects in smaller parts [16].

The main value adding factor reported in a paper describing a study on customer involvement activities was that activity should be planned, conducted and the results analysed [17]. The activities need to be conducted and the outcomes iterated for the usability of the software to be reasonable. The participants reported that the main constraints for choosing a particular activity were the stage of the project, availability of IT professionals and time. In one of the projects where the usability of the developed software was poor, the only customer involvement activity was evaluation. One of the main conclusions in the paper is that customers should be involved in all the four human-centred design activities [12].

2.3 Continuous Improvement: Non-Value Adding Activities

Shingo has studied Lean for manufacturing and identified seven categories of waste in manufacturing: In-Progress Inventory; Over-Production; Extra Processing; Transportation; Motion; Waiting and Defects [18]. The work of Poppendieck and Poppendieck [7] presents a mapping of waste categories from Lean in manufacturing to waste in software development, and define the following seven categories of waste:

1. *Partially Done Work* is present when chunks of code are impeded somewhere in the process from the start of the work before they reach the state of being integrated, tested, documented and deployable. Partially done work can be diminished by dividing work into smaller chunks or iterations [7, p.74]. Middleton, Flaxel, and Cookson [19] describe that Partially Done Work delay the product from being deployed, therefore a continuous flow should be pursued. A high amount of requirements put into the system is mentioned as a typical example of a problems. Further examples for Partially Done Work are uncoded documentation, unsynchronized code, untested code, undocumented code and undeployed code [7, p.74].
2. *Extra Features* are features in the software that have no clear value for the customer, or features that do not support the accomplishment of the customer's current job [7, p.75]. Extra features are similar to over-production, which is deemed by Ohno [11] as the worst of the seven wastes. If the feature has no clear value it should not be developed.
3. *Relearning* is rediscovering forgotten knowledge. Therefore it is essential to create and preserve knowledge as part of a learning process. Further, it is crucial to utilize existing knowledge and experiences from employees [7, p.76].
4. *Handoffs* occur when knowledge is transferred from one colleague to another. With every single handoff some tacit knowledge is lost, because it is difficult to make tacit knowledge explicit. Poppendieck and Poppendieck [7] state that only 6% of the original knowledge is left after a chain of 4 handoffs. Therefore it is necessary to reduce handoffs in order to reduce waste [7, p.77].
5. *Task Switching* requires knowledge workers to reset their mind after each switch. This resetting is time consuming and therefore seen as waste. It is considered, that working on tasks ought to be kept to a minimum in order to reduce task switches [7, p.78].
6. *Delays* occur in many different situations. One of the most important types of delay is waiting for people in different areas. Developers make critical decision about every 15 minutes. These decisions can only be made, if the required information is present. Colocated teams with short iterations and regular feedback can provide developers with the information they need to make decisions without delay. Hence, knowledge needs to be available when and where it is needed [7, p.79].
7. *Defects* within the software cause numerous problems and may lead to customer dissatisfaction. The target should be to deploy or deliver the product with the lowest possible defect rate. Mistake proofing tests and the discovery of defects in early stages of the software development [7, p.80].

These seven types of waste were used as a basis for interviewing IT professionals in our study about non-value adding activities in their work practice.

3 Method

The qualitative study included 10 semi-structured interviews, all following the same structure asking for the experience and context of the informants within they organization. The introductory part was focussing on the experience of the informants and the applied processes to understand the context. Questions to the informants were directed to how the informants improve their software development process, how the involve customers and what non-value adding activities they conduct in relation to waste.

The interviews were conducted mostly on site of the informants' organizations. Two of the interviews were conducted at Reykjavik University. All interviews lasted for about 45 minutes and were all conducted in English. The researcher also took notes and recorded the interviews. All recorded interviews were transcribed verbatim, with slight modifications to make the text more readable, when filler words or phrases disrupted the structure of a sentence. Informants are presented as males despite their actual gender in this paper.

The companies were chosen by analysing data from the Statistics Iceland office [20] and the Icelandic chamber of commerce [21]. The focus was on companies selling business to business software, both bespoke software and off-the-shelf software. All the chosen companies use agile software development processes, and many used Scrum. Some of the companies have adapted the Scrum process to their own needs and/or also apply Lean Software Development principles.

The informants were found through recommendations within the chosen companies. All of which have several years of experience in the application of Agile, especially in Scrum, and worked also as members of the development teams. The roles of the informants vary from company to company. The informants were categorised in three roles: Director of development (3 informants), Head of development (4 informants) and Scrum managers (Product Owner and Scrum Master), (3 informants).

The data from the interviews were compiled, analyzed and assigned to categories. Interpretative phenomenological analysis as described in Silverman and Rapley [22, p.274] was applied to identify and generate themes and sub-themes. This was an iterative approach and the themes were refined with every transcript of the interviews, which results in the final themes and sub-themes. Notes and the assigned themes were directly put down on the printed transcripts. First themes were generated, which lead to an initial list of themes. The themes in this list were clustered, which resulted in a list of themes of connected areas. As the last step these themes were organized in a table consisting of themes and sub-themes.

4 Results on Value Adding Activities

The following results show how customers are involved in the process of software development to eliminate waste in the development, how organizations have implemented a continuous improvement process, how they utilize metrics to support this process and how they work with value adding activities.

4.1 Customer Involvement

Most of the informants shared a common desire for a high degree of customer involvement. The term customer refers to different stakeholders on the customer side including users. The informants describe that a close relationship can prevent from misunderstandings about the requirements. Most of the interviewees prefer direct communication on the phone or through emails to prevent misunderstandings and it is "more convenient to just pick up the phone". One informant reported about a loose relationship with the customer, which led to misunderstandings and a higher amount of later improvements, once the software had been delivered. This experience has made the company value a closer relationship with the customer.

Several informants reported that it is important that the customers formally agree to participate in the software development and that Scrum is integrated as a part of their organisational culture, as in this quote: "The customer needs to fit the Agile process in their environment. They must learn it and it is hard in the beginning." However, as stated in the quote the integration of Agile in the organisational cultures is not always easy. Some customers are not interested in working according to Scrum but prefer to use the waterfall way of thinking: "some customers want to stay with the plain and old-fashioned waterfall model, and this is reflected in the nature of the tenders". One result from the interviews is that it is difficult to work according the Agile if the customer organisation use another way of working: "It is difficult to be Agile in a non-Agile environment.".

Sometimes the development team have problems with receiving feedback from the customers, as the organisational culture was not based on feedback and continuous improvement work. The pace in Scrum teams is perceived to be quick compared to the customer organisation, and this becomes a problem: "We ask for feedback, but the answer comes a lot later."

One informant experience that using Agile actually was seen by the customer organisation as decreasing the contact with the customer since Scrum prescribes that the team meets the customer at the end of every sprint, and not during the sprint work. In Scrum this is prescribed in order to reduce distraction. However, it is interesting to note that the system developer interviewed did not agree with this, but saw Agile as a way of increasing the customer involvement in the process: "but in fact the service level is actually increasing".

One argument used for a good user contact was that the need for education becomes less if the customer is much involved in the systems development process. "Interesting side effects also are, after developing a big piece the training was basically nonexistent", because of the high customer involvement.

Results indicate that it might be easier to use Agile if the customer organisation is relatively small and has a flexible way of working. One informant complained about the process and the interaction with the customer was less Agile due to the acquisition by a large corporation, “We are less Agile today than before the acquisition”. He said it was not possible anymore to involve the customers as much as before, even though that when “the customer invests the time, then this is really beneficial when it comes to waste” he commented.

4.2 Process for Value Adding Activities

Few of the informants reported that they have a well defined process for continuous improvement and that they follow this process strictly and most define continuous improvement work very loosely within the organization. This indicates the lack of awareness or even denial for problems within the development process in many organisation.

Several of the informants reported that they had not found a functioning process that supported their work with continuous improvement. Previous attempts to address working with continuous improvement was too tedious and time consuming: “it’s a tedious process and the same old stories are addressed again and again. This lowered the motivation of the employees. This needs to be changed”. The same happened in other organizations were continuous improvement used to be a part of the daily routine, “and nothing really changed, so the team members were not interested anymore. But we are trying to develop it again”. One should note that the same informant reported about major problems when it comes to a mutual understanding of the requirements.

In one of the organizations there are different colors for different types of tasks, and among these tasks are improvement tasks “It’s about improving the development process, solving root cause problems, if we are getting the same issue again and again”. Some organisations had a very functioning routine for working with continuous improvement: “Every Friday after lunch, we only work on our company, we do not work for the customers, we work on improving our business”. In one organisation a special improvement group maintains an improvement backlog, and everyone is invited to participate in this process. In one company there are retrospectives conducted on team level, but also on corporate level. There is a dedicated team, that tries to streamline those improvements, so teams can work together cross-functional.

4.3 Metrics

Results from our interviews show that metrics are rarely used to make improvements measurable. Making improvements measurable requires defined metrics to compare the state in the beginning and end. The respondents mostly have no metrics at all defined. This again indicates a certain lack of awareness for improvements. Those who have established metrics utilize them only on a high level, such as task traversing time as this is seen as the most important, as in this quote: “The most valuable metric,

for instance in this Kanban thinking, is the lead time. How fast things flow through the pipeline". The ultimate goal, according to several informants, is to deliver code as quickly as possible.

They also pointed out that the definition of metrics is very difficult and to use numbers might be tricky as well. Some of the informants pointed to difficulties in defining the correct metrics, because "We are measuring finished story points in a sprint, but when a story with 15 points is not finished, it is passed to the next sprint, although most of its work happened in the current sprint. So the statistics are skewed". They also stated the problem of finding a correct definition of meaningful and useful metrics.

5 Analyzing Non-Value Adding Activities

In this section we describe the results from the interviews which are analyzed according to the seven types of waste from Poppendieck and Poppendieck [7] since these denote non-value adding activities in a structured way.

5.1 Partially Done Work

It seemed as though the informants were not familiar with the concept of Partially Done Work, as they needed further explanations on the term in order to reply to the question. This indicates that they do not experience this type of waste regularly. A possible explanation can be found in the utilization of Agile, that inherently aim on reducing Partially Done Work, in order to deliver value fast by dividing the work load into smaller chunks to tackle this problem. Still, some informants mentioned that unfinished user stories were moved to the next sprint, but they did not seem to categorize that as partially done work.

Nine out of ten informants reported that unfinished features or non-fulfillment of requirements are the most common examples in the category Partially Done Work. Six informants described that testing is conducted by another person separately, therefore these informants finish the features partly, whereas four also test the software and thereby finish the development of that feature. All of the informants stated, that they use KANBAN like boards or status walls for better visibility of the status the work to be able to check if it is finished or not, e.g. one informant says: "We do visual management, so we have every ticket up on the wall".

Although the informants do not recognize a defect backlog as partially done work, 8 out of 10 maintain an inventory of defects, which is considered to be waste in the terms of partially done work by Poppendick and Poppendick [7]. One informant is aware of this type of non-value adding activities when he stated "It's an absolute waste to collect huge backlogs of defects that you review every 2 months and it's only the top 10% that's going to get ever implemented". He explained further, "We have a zero bug policy, although this is utopian, but we try at least."

It is interesting to note that even though, undocumented, untested or unfinished code is categorized as Partially Done Work in the theory from Poppendieck and Pop-

pendieck [7], the informants do not find these things problematic. Undocumented code is accepted since the Agile processes do not focus on documentation. Unfinished and undocumented code is also accepted according to the informants. Most developers choose a task or a user story, and make a running version of that requirement ready for testing.

5.2 Occurrence of Extra Features

Extra features occur in the informants' organizations. Some interviewees stated that they knew of this issue, whereas others claimed to exactly develop what is demanded by the customers. Only one informant stated to take the effort to analyze which features add value for the user in the productive system. They systematically remove them in succession, if there is no use for a particular functionality. For the rest, no real efforts were reported to estimate the value of a feature for the customer.

One informant did not see a problem with extra features, because even though they cannot charge this extra effort, "in the long run, we do not lose money with it, because of repeated business" and they want to live up to the expectations and even exceed them. Another informant appraised this issue similarly, "We learn from it and we profit from the knowledge we gain from it".

Reasons for extra features can often be found in miscommunication, and hence not knowing exactly what the customer demands. One informant has the problem of a "non-mutual understanding of what the users and what the business customers are actually asking for". In order to "keep the solution user friendly" one informant stated that they removed features and streamlined the solution. The same happens in another informant's organization. He explained: "it's a work system, there are people in there and there is no value having this feature in there when it's not used".

5.3 Relearning

Relearning is actually discovering forgotten knowledge, so the results are analyzed according to if the informants experience loss of knowledge and how the knowledge is preserved by documentation and communication in the team.

None of the informants responded that they would have a problem with the loss of knowledge within the development process. One informant stated, that about half of the development team left the company and this did not cause many problems, because "the application is tested, everything is tested and we use a high level language, so it's fairly easy to navigate through and also because everyone is telling everyone about everything."

The utilization of heavy-weight documentation is mainly used for contractual reasons, because it is requested by the customer. As one informant stated their customers would demand detailed documentation about the project, furthermore "The process is too partitioned; hence, documentation is needed, because there are so many people involved in the whole project development life-cycle". Some of the respondents maintain wiki pages and issue tracking systems for general documentation purposes, but

they also emphasized that most of the knowledge is shared in an open communication process.

All of the respondents utilize daily stand-up meetings and encourage team members to communicate face-to-face to share knowledge within the team. Most of the informants facilitate KANBAN walls, e.g. one informant explains the advantages of this tool as follows, "So we have all the tickets up on the wall, and this makes all the tasks visible for everyone". In general the interviewees emphasize the open and respectful communication within the teams. One informant even reports about a social contract that is concluded among the team members as follows "Everyone is open minded about asking questions and giving feedback". As teams in Scrum are self-organizing, the teams decide what is important and needs to be written down and what not. The informant also highlights the level of respect and cooperativeness within the software development team when it comes to integrating new team members, "Everybody would be so helpful, it's really good to pick up speed with a new team member".

Some of the interviewees reported about rotation within the teams to distribute knowledge, like one informant stated, "it is helpful to be at least knowledgeable of each part of the code". One informant declined the involvement of experts within the team and reported that they tried to share knowledge as much as possible, by using e.g. pair-programming or team member rotation, whereas some informants think that experts are important and argue that the high complexity of products makes experts necessary. One informant states that experts might be distracted by other team-members as they are a valuable source of information and the single contact point.

5.4 Handoffs

The results show that there is a certain pattern regarding handoffs from one IT professional to another identifiable. The activities of the IT professionals could be grouped in three categories: a) Requirement elicitation; b) Development and c) Testing and Release. These three categories of activities occur in all of the informants' organizations and there are typically handoffs between these within the organizations. Typically the development process is described as one of our informants phrased it: 'One developer works on one task, for bigger tasks the work is divided into smaller tasks'.

The IT professionals developing bespoke software typically elicited requirement in collaboration with customer. The IT professionals developing off-the-shelf software maintain wish lists, which are compiled from requests by the customers. In general the user stories are developed by the developers themselves for describing the requirements and also tested by the same developers. One informant described the problem of losing knowledge when the requirements in form of user stories are handed to the developers, "There is probably a loss of information going on". He explained further this effect when saying: "because we sit in a really open space, so there are a lot of discussions going on". Another informant described handoffs as problematic in the requirements elicitation process and the delivery process, because the development team is embedded in a large corporation, which affects the communication with the customers, like he explained: "The customer is fairly isolated from the development". Additionally, he stated: "For me I think this is too partitioned, too waterfall.

Especially, a lot of information is lost, when you have a totally separated team talking to the customer. So what is developed may have lost some context". Many informants stated that close relationship to the customer is beneficial when it comes to preserving knowledge.

One informant tries to tackle the problem of handoffs by delivering small parts of the software continuously: "It's a thinking of minimizing these handoffs and the costs of handoffs. So we have a deployment pipe all the way into operation". He also reported that having the same team working all the way from the kick-off meeting of the project to the operation eliminated a lot of waste. So a vital communication process within the team could counteract the loss of knowledge. The informants reported that there are basically no handoffs within the development activities, but there is an extra handoff when a separated testing team is involved. Some of the respondents also apply code-reviews to maintain a higher quality and support the communication process. This is technically not to be considered as a handoff, since knowledge is shared rather than transferred.

5.5 Task Switching

Because all informants use Scrum, an adapted Scrum process or Kanban, they all choose themselves which tasks to work on. One informant described this by saying: "The developer picks the task and then puts a sticky note on the board". Another informant describes this in a similar way: "It's the Agile and Lean principle of a pull system over push system, so the team pulls the tasks". The third informant describes how this is done in his team: "This happens in the daily stand-ups. Team members are self-organized and choose their tasks. We do not assign tasks. You commit to a task". One informant explained what all the other informants have in common, that is, that tasks are chosen accordingly to the priority by saying: "If you are a developer and you see the accepted tickets and there are three highly prioritized, so you pick one out of those three".

The informants agreed to that the limit to the number of tasks they are working on is just one task at a time, but this is not always possible due to many different reasons. Incoming defect reports can interrupt the development process, because defects need to be given a higher priority for some of the informants. Hence, when developers are working on defects they would have more task switching in their work. One informant explained this as follows: "Developers should work on one task at a time, but on the daily stand-ups they can choose more, because the planning period is 24 hours and they can finish more tasks within that period. Except if there are some defects coming in, that need attention, then the developer has to switch to that task". Two informants also explained that they needed to switch to solving a higher prioritized defect, if that was reported. In one organization a special role was introduced that is assigned to a particular team member every sprint cycle, which only handles defect reports. One informant explained that urgent customization requests from customers could interrupt a current task. One informant explained that this is disturbing for him, by saying: "If a developer needs to work on another ticket, the first task needs to be put on hold and then you can see how the hold queue is piling up. I think it's cumbersome to have

many tickets or working on many things at the same time". Another informant explained that developers only work on one task at a time, but the method of pairing is used, so there might be more tickets assigned to one developer than he actually works on. The third informant pictured the situation in his organization as follows: "Ideally (we work on) one task, but in reality things tend to get a lot worse than that. So context switching is a problem, especially for certain team members that are very knowledgeable".

5.6 Delays

The informants reported different reasons for delays, but there are two main themes identifiable. First of all there is a blocking in the development process due to missing actions by an outer stakeholder. One informant explains it as follows: "We are not synchronized enough with them. They have to do something and then we have to wait and cannot move and then they have something done and then we have to go on. But within the department things run quite smoothly, I think". Miscommunication and a lack of clear responsibilities can be seen as a reason for these delays. One informant had a concise answer to prevent this from happening. He explains: "One of our tag lines is: Responsibility all the way. We are responsible for the value and this story has no value until this blocking is resolved, we have to finish it. So we are offering our help, call them, and make sure things happen".

One informant explains that poor software design and the resulting complexity delays the development process and it takes a while until the developer understands the problem and structure of the software. This leads to the next common problem, fixing defects. Many informants consider this as the main reason for delays. Also the testing process can hinder the release of a feature, like one of the informants explains: "In our case the main reason for delays was the testing process. The testing process took too long and then defects were found, which postponed the process". The informant suggested that more testers were involved in the development.

5.7 Defects

The informants handle defects similarly. In most of the organizations the defects are registered and prioritized in a backlog, and depending on the severity level, the defects are fixed right away or taken care of later. In some of the organizations the backlogs are assessed from time to time to evaluate the status of defect backlog, so if the defects had become invalid, they would be removed from the backlog. One informant described that there are three different levels of defects: "It depends where a defect is found in the cycle. When it's in development we do not log it and take care of it immediately. In the release stage we log the defect and fix it. If the defect was delivered to the customer, there is a strict change process implemented." Only one informant stated that they have a zero-bug policy, which is supported by automated testing and the continuous delivery approach. He also adds that zero bugs are utopian, but it reflects the attitude towards defects. Furthermore, issues grouped as defects could also be failure demands, which is something that needs attention and could have been

avoided by setting the right actions in the very beginning. Failure demands evolve over the project period and need to be tracked and eradicated. Another informant explained that the high complexity within the system caused that it is more defect prone. So many defects were caused by a poor design in the very beginning. Informants also argued that defects interrupt their working pace.

In some organizations there was a special role introduced to handle defects. This role is reassigned to new developers every sprint, because this is a tedious work for developers, and it's positive for the individual's motivation to be only responsible for defects every couple of months. One informant explained: "Then the rest of the developers can focus only on stories and new features".

6. Discussion

In the following we discuss our results related to how IT professionals can continuously improve their way of working through value adding activities such as user involvement, better processes and metrics. Additionally we discuss our results on the non-value activities that IT professionals describe.

6.1 Discussions on Value Adding Activities

Our results show that many of the Agile projects have a close customer contact which concurs with the basic values in Agile [6]. The degree of contact is however perceived to be affected both by the size and the organisational culture of the customer organisation. According to the interviews the customer organisation needs to be flexible and able to deliver feedback and decisions at the same pace as the development project works, and this is not always the case.

One interesting finding is that several of the informants argue that formal agreements with the customer organisation regarding customer involvement are necessary. Since Agile is much based on motivation and close feedback [6], this does not concur with the basic value of the process.

It is interesting to note that Agile, as many other development processes, does not distinguish between the customer and the real user of the system. This blurs the aspects of real user involvement and makes them less visible.

Most companies in the interview study do not have a process for including value adding activities in their work, despite the fact that Agile has this explicit focus. Some explain this lack as caused by the tedious way in which previous attempt to include value adding activities. Others say that they do have a process, and that it is lightweight and adapted to the circumstances of Agile.

Agile does not give much detailed support, but is more a framework for systems development, and this results in lack of coherent methods such as methods for user involvement [23] and continuous improvement.

Our results show that metrics are rarely used for measuring in Agile. The only measurement mentioned in the interviews are measurements of speed and time. There were no measurements made of any quality aspects such as usability or user

involvement. This concurs with other research on usability evaluations [13,14]. This situation might be caused by the basic value of speed in Scrum [6], but it might also be caused by the fact that it is difficult to measure other aspects than time related measurements [24]

One can wonder if one avenue forward to include usability work in organisations is measurement? Previous research on measuring usability and user experience have shown that such measurements have little or no influence on forthcoming decisions [25]. However, it is very possible that these results reflect the trends of that time and that the introduction of usability measurements would be more successful when included in systems development processes based on Lean values since measurement as such are a core value there.

When introducing measurements in organisation it is crucial that there are agreed success criteria, and it is difficult to establish evaluation criteria for social elements that are affected by the introduction of an IT system [1]. However, attempts have been made to use metrics in order to measure the establishment of user centred systems design [26] in organisations such as for example by Gulliksen et al. [27].

6.2 Discussion on Non-Value Adding Activities

Our results show that the IT professionals are well aware of waste in their organizations and it seems as if they do not interfere with their ability to deliver fast. Yet, there are some types of waste that impede the development processes. Two categories were most dominant, defects in the software that often result in a partially done work and delays in the IT professionals' work due to lack of communication.

Most of the informants maintain defect backlogs, however, the informants do not perceive this collection of defects as a problem. This clearly contradicts the Lean Software Development principle "Build Quality In" [7], where they state that it is better to avoid defects in the beginning, than to test quality into the product in late stages. Maintaining a defect backlog leads to context switching, because developers are forced to choose more than one task at a time, depending on the prioritization. Most of the respondents referred to partially done work as working on defects and maintaining defect backlogs. The missing perception or negligence of this type of waste can lead to be highly ineffective. Some companies had introduced a special role to diminish these effects. It was described that a member of the development team handled these defects for a limited amount of time before this role was passed on to another person, because of its tedious character.

The lack of communication was the main factor for waste related to delays in the IT professionals work and producing functionality that the IT professionals regarded as extra features.The IT professionals needed good communication within the team and with the customer. Additionally the needed feedback from the customers or users in order to adapt to changing situations and deliver value fast. The following statement summarizes the main obstacles within the development process the best: "It is difficult to be Agile in a non-Agile environment." Even though Agile processes were applied (mostly Scrum), the customers and other stakeholders from the external environment are not used to collaborating in this Agile environment. This is mainly

attributed to the disparity in the understanding of collaboration and communication between vendor and customer. Agile teams need a close collaboration and feedback from the customer in order to adapt to changing situations and deliver value fast. It seemed that customers are still more used to the traditional processes, e.g. waterfall model, where a different degree of communication is needed throughout the process. The customer needs to be educated to collaborate in an Agile environment. This lack of communication was the main factor for delays and extra features. The negligence of this issue leads might lead to major waste within the development process.

7. Conclusion

The results in this study on the value adding activity of involving users show that the IT professionals emphasise communication with user both through direct contact and using email to add value to the software development. However, some informants describe that it is not always easy to work according to agile processes with customer organisations that are not used to that way of working. The IT professionals rarely use metrics to make improvements or value adding activities measurable. The most serious non-value adding activities are categorised as: partially done work, delays and defects. The core reason is that there are long lists of defects in the projects, which means that the software is partially done and the defects cause delays in the process. Even though there are efforts to reduce non-value adding activities in the process, these organizations are still confronted with problems attributed to miscommunication and the impediments by the external environment.

8. References

1. Baxter, G., & Sommerville, I.: Socio-technical systems: From design methods to systems engineering. Interacting with Computers, 23(1), 4-17 (2011)
2. Suchman, L.: Plans and situated actions., Cambridge University, New York (1986)
3. Eason, K.: Changing perspectives on the organizational consequences of information technology. Behaviour & information technology, 20(5), 323-328, (2001)
4. Boehm, B.: A view of 20th and 21st century software engineering. In: Proceedings of the 28th international conference on software engineering, ACM Press, Shanghai, China, (2006)
5. Basili, V. and Turner, J.: Iterative enhancement: A practical technique for software development. IEEE transactions of software engineering, Dec., 390 - 396, (1975)
6. Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., et al.: Agile manifesto. `http://agilemanifesto.org`
7. Poppendieck, M., & Poppendieck, T.: Implementing lean software development: From concept to cash (3rd ed.), Addison-Wesley Professional, New York (2007)

8. Williams, L.: What agile teams think of agile principles. Communication of the ACM, 55(4), (2012)
9. Modig, N. & Åhlström, P.: This is Lean – resolving the efficiency paradox, Bulls Graphics AB, Halmstad, (2012)
10. Kniberg, M., Henrik S.: Kanban and scrum - making the most of both. C4Media Inc., (2010)
11. Ohno, T.: Toyota production system: Beyond large scale production, Productivity Press, (1988)
12. International organisation for standardisation - ISO 9241-210:2010: Ergonomics of human-system interaction - Part 210: Human-centred design process for interactive systems, Switzerland (2010)
13. Lárusdóttir, M., Cajander, Å., & Gulliksen, J.: Informal feedback rather than performance measurements–user-centred evaluation in Scrum projects.Behaviour & Information Technology, (ahead-of-print), 1-18, (2013)
14. Jia, Y., Larusdottir, M. K., & Cajander, Å.: The usage of usability techniques in Scrum projects. In Human-Centered Software Engineering (pp. 331-341). Springer, Heidelberg, (2012)
15. Larusdottir, M. K., Bjarnadottir, E., Gulliksen, J. : The Focus on Usability in Testing Practices in Industry, Proceedings of the Human Computer Interaction Symposium at the World Computer Congress 2010, Brisbane, Australia, (2010)
16. Cajander, A, Larusdottir, M.K., Gulliksen, J.: Existing but not Explicit - The User Perspective in Scrum Projects in Practice, INTERACT 2013, Cap Town, (2013)
17. Bruno, V. and Dick, M.: Making usability work in industry: An Australian practitioner perspective. In: Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining user interfaces, ACM Press, Adelaide, Australia., (2007)
18. Shingo, S.: Study of toyoda production system from an industrial engineering viewpoint. Productivity Press, (1982)
19. Middleton, P., Flaxel, A., & Cookson, A.: Lean software management case study: Timberline inc. In H. Baumeister, M. Marchesi, & M. Holcombe (Eds.), Extreme programming and agile processes in software engineering (Vol. 3556, p. 1-9) Springer, Heidelberg, (2005)
20. Statistics Iceland office: `http://www.statice.is`
21. Icelandic chamber of commerce: `http://www.vi.is`
22. Silverman, D., & Rapley, T.: Qualitative research (Vol. 3). SAGE Publications Ltd., (2011)
23. Salah, D., Paige, R., & Cairns, P. A Systematic Literature Review on Agile Development Processes and User Centred Design Integration, (2011)
24. Jokela, T., Koivumaa, J., Pirkola, J., Salminen, P., & Kantola, N.: Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone. Personal and Ubiquitous computing, 10(6), 345-355, (2006)

25. Gulliksen, J., Cajander, Å., & Eriksson, E.: Only Figures Matter?–If Measuring Usability and User Experience in Practice is Insanity or a Necessity. In International Workshop on , 91, (2008)
26. Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., & Cajander, Å.: Key principles for user-centred systems design. Behaviour and Information Technology, 22(6), 397-409, (2003)
27. Gulliksen, J., Cajander, Å., Sandblad, B., Eriksson, E., & Kavathatzopoulos, I.: User-Centred Systems Design as Organizational Change: A Longitudinal Action Research Project to Improve Usability and the Computerized Work Environment in a Public Authority. International Journal of Technology and Human Interaction (IJTHI), 5(3), 13-53, (2009)