Information and Software Technology 105 (2019) 78–94

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations

Hiva Alahyari[a,*], Tony Gorschek[b], Richard Berntsson Svensson[b]

[a] Department of Computer Science and Engineering, Chalmers |Gothenburg University of Technology, Gothenburg, Sweden
[b] Department of Software Engineering, Blekinge Institute of Technology, Blekinge, Sweden

ARTICLE INFO

Keywords:
Waste
Non-value adding activities
Lean software development
Agile software development

ABSTRACT

Context: The principal focus of lean is the identification and elimination of waste from the process with respect to maximizing customer value. Similarly, the purpose of agile is to maximize customer value and minimize unnecessary work and time delays. In both cases the concept of waste is important. Through an empirical study, we explore how waste is approached in agile software development organizations.
Objective: This paper explores the concept of waste in agile/lean software development organizations and how it is defined, used, prioritized, reduced, or eliminated in practice
Method: The data were collected using semi-structured open-interviews. 23 practitioners from 14 embedded software development organizations were interviewed representing two core roles in each organization.
Results: Various wastes, categorized in 10 different categories, were identified by the respondents. From the mentioned wastes, not all were necessarily waste per se but could be symptoms caused by wastes. From the seven wastes of lean, Task-switching was ranked as the most important, and Extra-features, as the least important wastes according to the respondents' opinion. However, most companies do not have their own or use an established definition of waste, more importantly, very few actively identify or try to eliminate waste in their organizations beyond local initiatives on project level.
Conclusion: In order to identify, recognize and eliminate waste, a common understanding, and a joint and holistic view of the concept is needed. It is also important to optimize the whole organization and the whole product, as waste on one level can be important on another, thus sub-optimization should be avoided. Furthermore, to achieve a sustainable and effective waste handling, both the short-term and the long-term perspectives need to be considered.

1. Introduction

Many embedded software development organizations have adopted agile software development methods with the goal of handling several of the challenges associated with the development of software-intensive products – such as dynamic environments, hardware dependency, and longer lead times due to product complexity and interdependencies [1]. As software by itself has little intrinsic value and its value appears when it enables delivery of products and services (embedded), it is more useful to think of software development as part of product development [2,3,4]. Lean principles and practices are about improving the product development system in order to increase the speed and quality and deliver value to customers [2,4,5]. Thus, Lean principles can be seen as operations to guide the practice of agile software development - especially in the embedded software domain [3,4,6].

In lean principles, all activities and work products that do not contribute to customer value are considered waste (or Muda) [2,5,7]. The first principle of lean software development is to manage waste and focus on the identification and elimination of waste from the process [2]. However, despite the fact that waste reduction and elimination has been suggested as the first step towards the process improvements in lean, the concept of waste is relatively unexplored in the literature [8,9–12]. How do the software development organizations (SDO) define waste, what do they actually consider to be waste, and how do they eliminate or measure waste?

To investigate these questions, we conducted an empirical study to explore the concept of waste to see (a) how waste is defined and used in practice, (b) how it is prioritized and measured, and (c) what activities exist to reduce or eliminate waste in agile software development organizations developing embedded software. This paper presents the

* Corresponding author.
E-mail addresses: hiva@chalmers.se (H. Alahyari), tony.gorschek@bth.se (T. Gorschek), richard.berntsson.svensson@bth.se (R. Berntsson Svensson).

https://doi.org/10.1016/j.infsof.2018.08.006
Received 12 July 2017; Received in revised form 10 August 2018; Accepted 13 August 2018
Available online 31 August 2018
0950-5849/

results of an empirical study including data collected through in-depth interviews with 23 respondents from 14 different software development organizations based in Sweden.

The remainder of this paper is organized as follows. In Section 2, the background and related work are presented. The research methodology is described in Section 3, and Section 4 presents the results and relates the findings to previous studies. Section 5 holds the main conclusions. In order to distinguish between various wastes mentioned in our study, depending on if they are 1) one of the seven wastes of Lean, or 2) additional suggested wastes extracted from peer-reviewed literature, or 3) wastes and related aspects of wastes resulted from our study, we have denoted them differently and have used *Italic Fonts* for the first group, "quotation mark" for the second group, and Starting with capital letter (plus an ID showing which category they belong to) for the third group.

2. Background and related work

Since the introduction of agile manifesto in 2001 [13], agile methodologies with the promise of satisfied customers through early and continuous delivery of valuable software have attracted many practitioners. Many software development companies have readily accepted agile development methods such as extreme programming [14], SCRUM [15] and lean software development [2].

The success of many of the practices of agile software development can be explained by understanding the principles of lean software development [2]. The main principle of lean is the identification and elimination of waste from the process with respect to customer value [2]. While a substantial amount of papers (e.g. [16–23]) have been published in recent years on issues related to agile and lean software development, few studies (e.g. [6,8,10–12,24–26]) have dedicated their focus to investigate the agile and lean core concepts i.e. value creation and waste elimination.

The use of lean principles in the software development field and the term "Lean Software Development" as an agile toolkit, was introduced for the first time in 2001 [2]. Poppendieck and Poppendieck [2] believed that software development is just a subset of the overall product development process and state that "it is the product, the activity, the process in which software is embedded that is the real product under development". Thus, they suggested that "in a very real sense, we can call software development a subset of product development" [4]. In Lean Development in addition to the suggestion that companies should look for eliminating waste everywhere in their processes, a set of seven wastes that are common wastes, are suggested to be watched for, as these wastes happen and exist in most of the organizations [2,4,5]. In order to aid software development to identify and eliminate the waste easier, the seven wastes of manufacturing have been translated into seven waste of software development as follow:

1. *Partially-Done-Work* refers to those works that are not implemented, integrated or tested and tends to become obsolete, and the main problem with it is that you might not know if it will eventually work.
2. *Extra-Processes* are those processes that are not necessary (such as paperwork) or those that can be delayed to the end of a process.
3. *Extra-Features* might seem harmless and it may seem even a good idea to put some extra features into a system just in case they are needed, but on the contrary, it is a serious waste.
4. *Task-Switching* is a source of waste and every time that software developers switch between tasks, a significant switching time is incurred as they get their thoughts into the flow of the new task.
5. *Waiting* is one of the biggest wastes in software development and is usually waiting for things to happen e.g. delays in starting a project, delays in reviews and approvals, delays in testing or deployment.
6. *Motion* is about the amount of effort and motion takes a developer to find out the answer to their question/need. It could be a need to solve a technical problem or the need to access customer or customer representatives or the need to see the results of a test.
7. *Defects* are a self-evident source of waste in software development processes. The amount of waste caused by a defect is the result of the defect impact and the time it remains undetected [2].

Later, Poppendieck and Poppendieck [4] revisited the seven wastes of lean software development, made few changes and introduced "Relearning" and "Handoffs". "Relearning" is defined as "rediscovering something we once knew and have forgotten it" and "Handoffs" are the loss of tacit knowledge when work is handed off to others [4].

Lean development can also be approached from the "flow" point of view, as described by Toyota [27]. In this approach, the focus is on improving the "flow" or a smooth way of working, which is the elimination of unevenness [27,28]. Mandic et al. [29] look at the "flow" as a collection of interconnected value creation points in order to aid in understanding the source of the waste. They suggest two super-classes of waste sources that are "interconnections of the value creation points" and "inability to inject value in a value creation point". It is suggested that the use of the two suggested super-classes can help in the decision-making processes and the decision flow. This is done through unveiling new sources of waste, and four of these sources which are "avoiding decision-making", "limited access to information", "noise or information distortion" and "uncertainty" are proposed [29].

Although the software industry can learn a lot from the manufacturing domain, agile teams and organizations are better understood as self-organizing complex adaptive "systems", which have emergent properties [12]. Hence, waste can have different meanings in such systems and there are more aspects to be considered. Thus, Power and Conboy [12] suggest calling wastes "Impediments to flow" instead. They propose nine impediments, of which six are equal to six of the wastes identified in lean [2]. One of the three additional suggested impediments is called "failure demand" and it partly (if not fully) overlaps with the waste of *Defects* [2]. The other two additional suggested impediments are "handovers" and "unmet human potential" [12]. "Handovers" occur when incomplete work is handed over from one person/team to another. They impede the flow of work by adding delays, requiring more people, or the loss of knowledge during the handover from a person/team to another. Though, this waste could be seen as part of or related to the waste of *Task-Switching* in lean software development. "Unmet human potential" is the waste of not using, or promoting people's skills/abilities to their full potential, which may lead to not having the workflow as efficient or as effective as possible [12].

Looking at the product development field in general, three additional wastes have been suggested to the set of seven wastes of lean by Pessoa et al. [30]. The three additional wastes are: (1) "correcting", (2) "wishful thinking", and (3) "happening". "Correcting" is the redoing or scrapping due to feedback, which the authors separate it from the waste of *Defects*. "Wishful thinking", means making decisions without the needed input/data, or operating based on incorrect or insufficient controls that do not guarantee the value delivery. "Happening" includes all reactions to unexpected happenings in the environment, which could result from changes of the internal environment such as structures and/or rules, or from failing to forecast the external changes such as business, customer and market changes [30].

Studies focusing on waste in software development companies, only a few are empirical studies that have been conducted in the real industrial setting. One of the studies reports a set of wastes that have been identified through an observational single-case study [33] and compares the set of identified wastes with the set of seven wastes in [4]. Although, not all of the identified wastes are necessarily waste and may better be explained within e.g. productivity issues. According to the study [33], four of their identified wastes are not described in [4], which are called "Unnecessary complex solutions", "Extraneous cognitive load", "Psychological distress" and "Ineffective communication". The identified root causes for "unnecessary complex solutions" are feature complexity, technical complexity and/or a lack of reuse [33].

“Extraneous cognitive load” is based on a theory that suggests that the working memory is limited, and if overloaded, it may inhibit us from learning and problem solving [33]. Thus, the cognitive loads, unnecessarily added by the task environment or the way the task is presented, creates waste [33]. "Psychological distress" refers to job-related psychological distress, which may cause absenteeism, burnout, lower productivity and several health problems [33]. “Ineffective communication” refers to incomplete, incorrect, misleading or absent communication, and imbalance communication, large team sizes, and inefficient meetings reduced productivity [33].

In [11], Kanban (one of the lean development tools [28]) was used in the case company to address the seven wastes of lean. The results showed that Kanban was useful in visualizing and organizing the current work, though, unable to prevent the waste from happening [11]. In another case study conducted at a telecom company [34], they looked into waste detecting indicators and identified four indicators. The indicators are 1) the flow of maintenance request through the maintenance process with regard to continuous value creation and high throughput, 2) the inflow of maintenance requests, 3) the analysis of lead-times, and 4) the analysis of workload.

In order to eliminate waste, it is important to understand *value*, as value and waste are two concepts that go hand in hand. A relevant practice in this regard is a practice known as Value Stream Mapping (VSM) [31]. VSM is one of the lean practices recommended for process improvement purposes in lean organizations, to optimize the process of value creation and eliminating waste [2,4]. Through visualization of the current state map of a process/product, VSM assists to identify bottlenecks in a process that impedes it from flowing at its optimum [31], and it is suggested as a good method for discovering waste, evaluating, and improving the software development processes [2]. A study conducted in a telecom company to identify waste-related problems in a software product customization process using VSM, showed that the identified solution using VSM was useful for improving the company's software development process [8]. In a similar study [10] VSM was integrated with a more deliberate approach towards value aspects, using the Software Value Map [25]. The results showed that using the software value map was useful both for general improvements and also waste elimination, as well as enabling practitioners to defined and qualify value in the context of waste removal. That is a deeper understanding and more complete definition of value also allowed practitioners to better identify activities that did not produce value. However, due to its time-consuming aspect and the difficulties that come with performing a VSM [32], it has not seen a wide use. Furthermore, to evaluate the value creation process, there are not many suggested metrics, except for some to measure time and costs [31], which are project focused and regard short-term value considerations only. Long-term value and waste considerations require predictions and estimates on a more advanced level [31].

One of the few studies focused on the concept of value is our previous study reported in [26], which investigates how value is defined and approached amongst various software development organizations as perceived by the companies themselves. Therefore, unlike a VSM evaluation that looks into a particular company's processes and products, in [26], as an empirical base we studied the companies’ perception of value, to get an understanding of state-of-practice and to identify common ground and perhaps the most important value aspects and perspectives amongst various organizations in various domains. We asked our study participants to state what they consider the value and then they were asked to prioritize the value aspects. All the stated value aspects were then sorted into categories or the so-called value perspectives and aspects, using an existing software value map [25]. Our result showed that Customer related aspects were the most important value perspective amongst all various types of organizations, and delivery time was ranked as the most important aspect of it. Perceived quality was another important aspect that most of the organizations agreed upon.

However, our study [26] did not investigate the concept of waste at all as studying value was complex enough. Although value and waste are related, studying both concepts in the same study would require the introduction of two concepts as well as the elicitation of the definition, use, and understanding of both concepts and the relationships between waste types and value types. We judged that any organization mature enough to have well-defined waste and value concepts would volunteer this information during the current study, but we opted to focus on waste and only be observant of if value became a part of the discussion. As the result indicate (see Section 4) we can see that value was only brought up on a very superficial level in relation to waste, as an example a practitioner stating “the concept of value needs to be explored". When pressed on the issue the view and use of value were very much as confirmed in our previous study [26]. Delving deeper into the details on the "value" in this waste study would have prevented us from delving deeper into the waste concept itself. This was a deliberate trade-off, albeit not optimal in any way, necessary due to time and resource limitations.

Overall there are only a few studies [8,10,11,33,34] looking into waste in software development organizations. Even among these, none try to elicit the actual practical and complete view on waste, rather touch on parts of waste and often study the phenomenon indirectly. This study on waste is considered meaningful as it tries to catch practitioner's views, definitions, and handling (measurement, removal) of waste.

3. Research methodology

We used a qualitative research method in this study [35–37]. The qualitative research fits the purpose of our study as it is concerned with studying matters in their natural setting, discovering causes remarked by respondents and understanding their view of the problem [35,36,38]. The purpose of this study was to increase the understanding of the concept of waste in practice as well as which wastes are considered the most important in agile software development organizations utilizing agile/lean development methods. The data collection method used in this study was semi-structured interviews with open-ended questions. Since the concept of waste may be defined and approached differently amongst the Software Development Organizations (SDO) and be contextually dependent, we chose interviews over other data collection methods. The interactions between the interviewer and the interviewee could aid in clarifying potential ambiguities when necessary, and it helped to uncover the different definitions and approaches associated with wastes. The following research questions provide the focus for the investigation.

RQ1: How is the concept of waste seen and defined in SDOs using agile/lean approaches?
RQ1.1: What wastes are considered most important by SDOs using agile/lean approaches?
RQ2: How is waste approached, eliminated, and/or measured in SDOs using agile/lean approaches?

3.1. Research design and data collection

Planning/Selection. We aimed at interviewing two roles within each software development organization. The first role was responsible for the overall development process and had different titles such as Project/Process Manager (PM), agile expert/driver, scrum master, which are all abbreviated as PM, as it can be seen in Table 1. The second role decides on selection and prioritization of the requirements for implementation purposes, often known as Product Owner (PO), but also in a few cases were called technical expert and business consultant. However, all the roles had the same responsibility in essence and are abbreviated as PO in Table 1.

Table 1
Software development organization (SDO) and the study respondents.

SDO	Domain	Company	Role	Years involved in this role	# of employees (in SDO)	# of agile practices used	Prioritized the seven wastes of Lean
A	Telecom	A provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators.	1.Agile driver (PM)	5	~300	30	Jointly
			2.Product owner (PO)	1		33	
B	Telecom	A provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators.	1.Product owner (PO)	1,5	~300	25	Jointly
			2.Project manager (PM)	1		29	
C	Telecom	A provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators.	1.Product owner (PO)	0,5	~300	11	Independently
			2.Project manager (PM)	1		24	
D	Telecom	A provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators.	1.Program responsible (PM)	5	~55	46	Independently
			2.Product owner & System architect (PO)	5		42	
E	Telecom	A provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network operators.	1.Customer project manager (PM)	8	~28	20	Independently
F	Automotive	Automobile manufacturer and supplier of transport solutions for commercial use.	1.Process manager (PM)	1	~25 (Only team)	15	Independently
			2.Product owner &Technical expert (PO)	2		15	
			3. Architect & Requirments (PO)	8		15	
G	Automotive	Automobile manufacturer and supplier of transport solutions for commercial use.	1.Product owner (PO)	3	~15 (Only team)	21	Independently
			2.Scrum master (PM)	3		27	
H	Automotive	An equipment manufacturer developing and selling a variety of products within the embedded systems domain with the main focus on automotive industry.	1.Project manager (PM)	5	~100	9	Independently
			2.Product owner (PO)	0,5		26	
I	Automotive	Software and system supplier for automotive industry.	1.Project manager &Developer (PM)	3	N/A	22	Independently
J	Defence industry	Developing systems for military defense and civil security, focused on surveillance, threat detection, force protection and avionics systems.	1.Scrum master (PM)	1	~200	42	Independently
			2.Product owner & System architect (PO)	0,5		31	
K	IT-Consultancy	Consulting company in IT and communications technology with clients mainly in telecoms,commerce and media sectors.	1.Scrum master (PM)	3	~110	39	Independently
L	IT & Management Consultancy	IT and management consultancy firm with various areas of expertise including telecommunications, mobility, simulation technologies, critical business systems, games, media and entertainment and IT in vehicles.	1.Scrum master & Verification responsible (PM)	3	~180	39	Independently
M	Consultancy	Consultancy company developing solutions through software, and also developing people and businesses.Their clients are mainly in automotive and telecom domains.	1.Consultant & Agile coach (PM)	1,5	N/A	45	Independently
N	Consultancy	Providing both Information technology products and services in personal and enterprise computing for variety of clients in various domains.	1.Business consultant (PO)	1	~6 (Only team)	21	Independently

The sampling strategy was a combination of maximum variation sampling and convenience sampling [37]. For each company, we contacted a “gate-keeper” to identify the two respondents that would best fit the defined target roles. Another advantage associated with the "gate-keepers" is that they could help in keeping the selection process free from any personal interests and/or biased influences. Depending on the size of the organization and the definition of areas of responsibility, the selected roles could be represented either by two respondents or only one. In total, 23 respondents (13 PMs and 10 POs), at 14 Software Development Organizations (SDO) from nine companies participated in our study which resulted in 23 data points (see Table 1 for details).

All 14 SDOs used agile methods and developed embedded systems, however, varied in respect to size, type of products, and application domain. A characterization of the SDOs (following the guidelines of [39]) can be seen in Table 1.

Data Collection. The data was collected through semi-structured interviews with open-ended questions [35]. All interviews were conducted in English and varied in length from 60 to 90 min. All the interviews were conducted face-to-face. In order to facilitate and improve the analysis process, in addition to the written extensive notes and paper sheets that were marked during the interviews, all the interviews were recorded. The questions were answered from the perspective of the respondent's role and SDO.

The interviews were divided into three distinct parts. In part one, the respondents talked about their role and their responsibility/activities within their role, as well as information about their agile development environment and processes. To this end, the respondents were given a list of 58 agile practices [41] and were asked to mark and identify which practices were utilized in their organization.

The second part of the interview focused on Value, in which respondents were asked to identify and prioritize Value Aspects that are important for their SDO. The results from this part are reported in [26].

The third part of the interview focused on waste, what the respondents consider as waste, how they deal with it, and how they eliminate and measure it. Finally, the respondents were asked to prioritize the seven wastes of lean [2]. This prioritization was performed using a paper sheet with three columns. The first two columns included the seven wastes of lean and their respective definition. The last column was left empty for the prioritization of the seven wastes in relation to the importance of controlling/eliminating the wastes. The prioritization ranged between 1 and 7 where 1 indicated the most important and 7 indicated the least important waste to be controlled/eliminated.

In some of the interviews, one respondent and one interviewer attended, while in others two respondents and one interviewer were present. The purpose of the researchers was that independent from the interview setup if it was conducted jointly or separately, all the respondents would identify wastes and prioritize the seven wastes of lean independently. However, few respondents preferred to prioritize the seven wastes of lean jointly. If the respondents prioritized the seven wastes of lean independently or jointly, is shown in Table 1, under the column identified as "Prioritized the seven wastes of lean".

The interview guideline for this study can be found in the provided link at the bottom of this page.[1]

Analysis. All the collected data, the recordings and the written extensive notes, were analyzed using content analysis [35,36]. In content analysis, the focus is to generate findings from the collected information, thus one of the essential aspects of content analysis is the use of categories [36]. In this study, first, we entered all of the collected information, both from the recordings and the written extensive notes, into spreadsheets in order to facilitate the data analysis. Then, once the collected data were read and coded, all of the codes were organized into higher abstraction level codes, and finally assigned to a category. Each of the identified categories (See the categories in Table 3 (W1–W10), Table 6 (A1–A9 and P1–P4), and Table 7 (M1-M5)) were then discussed and refined among the authors. The categories originated either from prior literature [2,4] or from the collected data. The analysis of data was an iterative and interpretative process where the authors had the opportunity to get back to respondents in case of uncertainties. The results from the analysis are found in Section 4.

4. Results and analysis

4.1. Cases studied and agile practices used

To enable a better understanding of the agile development environment and the SDO's use of agile practices the respondents were asked to identify the practices utilized in their SDOs using a list of 58 practices [41]. How many practices are used per SDO is shown in results visible in Table 2. In addition, the frequency of each utilized agile practice is shown in the last column. It is worth to mention that according to Agile Alliance list of practices and mapping the practices into various agile-tribes/ areas of concern [42], practices number 32 (Lead time) and number 47 (Kanban Board) in Table 2, are considered Lean practices, and practice number 43 (Definition of Done) it is a shared practice between Lean and Scrum.

In Table 2, respondents with the same letter ID belong to the same SDO, and the IDs correspond to the SDO and the respective roles, as shown in Table 1. For example, respondents A1 and A2 are both from SDO A and have the roles of Agile Driver (A1) and Product Owner (A2). The last two rows in Table 2 show how many agile practices were used per respondent and per SDO.

As can be observed in Table 2, there are discrepancies between the statements from respondents within the same organization as to which agile practices are used, e.g. see SDO H. During the interviews, the respondents individually (except for the respondents from SDO F, identified with F1, F2, and F3) selected agile practices used, hence discrepancies were expected to some extent since "using" a practice is a matter of judgment. In addition, as the roles of the respondents were different, each respondent may not come into contact with all or the same practices. The focus of the study was, however, to study waste explicitly, and not what agile or lean practices were used. The mapping of the used practices should be seen as background data to be potentially used as part of the analysis (as can be seen in Section 4 and its subsections).

The average number of used agile practices was 26. With regards to the domain and the size of the SDO, we could not find any statistically significant correlation between use of agile practices and the domain, or the size of the SDO. The only difference between the domains and used agile practices was the number of used practices. The average number of used agile practices in the defense and consultancy industry was 37 and 36 respectively, while the telecom industry had an average of 26, with the automotive industry coming in at 19.

4.2. Waste and industrial use (RQ1)

As waste elimination is important in both lean and agile, we asked the respondents to first define waste, provide examples of waste(s) found in their development processes, and characterize the identified waste(s) and the importance and impact it may have in their practical application of agile. This was done without giving them any sort of account of waste types up-front as we did not want to influence them at this stage.

The responses varied amongst the respondents and a few respondents could not define waste at all. The reasons for this could be that the respondents had not been exposed to these types of questions before, or they had not thought about waste in terms of what it really means and implies. On the other hand, some respondents answered by providing examples from the development environment that could be

[1] https://gubox.box.com/s/jjq82crqd0uugr8lw1xe62v32hjlzvzx.

Table 2
Use of agile practices amongst the study respondents.

		Respondents																						
#	List of Agile Practices	A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	How many respondents use the AP
1	Niko-Niko Calendar																								0
2	Project Chartering																								0
3	CRC (Class, Responsibilities, Collaborators) Cards																	x							1
4	Given - When - Then																		x						1
5	INVEST		x																				x		2
6	Ubiquitous Language								x														x		2
7	Rules of Simplicity								x										x	x					3
8	Story Mapping		x					x									x								3
9	Three C's (Card, Conversation, Confirmation)								x											x		x			3
10	BDD (Behavior Driven Development)																x				x		x	x	4
11	Mock Objects		x							x							x						x		4
12	Definition of Ready							x												x	x	x	x		5
13	Personas							x	x							x	x		x						5
14	Milestone Retrospective	x															x	x	x	x			x		6
15	Role-Feature-Reason							x	x								x		x			x	x		6
16	Quick Design Session							x	x		x	x	x						x			x			7
17	Simple Design	x						x							x				x	x	x		x		7
18	Three Questions							x	x								x		x	x		x	x		7
19	Collective Ownership							x	x									x	x		x	x	x	x	8
20	Information Radiators		x	x	x												x		x	x	x		x		8
21	Relative Estimation		x				x	x	x					x	x				x			x			8
22	Story Splitting		x			x		x											x	x	x	x	x		8
23	Sustainable Pace						x	x	x										x	x	x	x	x		8
24	ATDD (Analogous to Test Driven Development)							x	x					x	x			x		x	x	x	x		9
25	Continuous Deployment	x						x	x	x				x	x	x					x			x	9
26	Facilitation				x			x						x	x		x				x	x	x	x	9
27	Heartbeat Retrospective					x		x	x	x				x	x							x	x	x	9
28	TDD (Test Driven Development)	x					x	x	x						x			x			x	x	x		9
29	Usability Testing		x		x		x	x	x								x		x				x	x	9
30	Sign up for Tasks		x		x		x	x	x						x				x		x	x	x		10
31	Pair Programming			x	x			x			x	x	x					x	x		x	x	x		11
32	Lead Time	x	x	x	x	x		x	x		x	x	x								x		x		12
33	Refactoring	x	x	x	x		x	x	x										x	x	x	x	x		12
34	Scrum of Scrums	x	x		x		x	x	x	x									x	x	x	x	x		12
35	Story Points	x	x	x	x		x	x	x	x									x		x	x		x	12
36	Team Room	x	x	x	x			x	x								x		x	x	x	x	x		12
37	Time-box	x	x	x	x	x	x	x	x	x									x			x	x		12
38	Planning Poker		x	x	x		x	x	x					x	x				x	x	x	x	x	x	14
39	Velocity	x	x	x	x	x	x	x	x						x		x		x		x	x	x		14
40	Frequent Releases	x		x		x		x	x		x	x	x	x	x		x	x	x		x		x		15
41	Task Board	x	x		x			x	x		x	x	x		x		x	x	x		x	x	x		15
42	Automated Build	x	x	x	x			x	x	x	x	x	x					x	x	x	x	x	x		16
43	Definition of Done	x	x	x	x		x	x	x	x				x	x			x	x	x	x	x	x		16
44	Incremental Development	x	x	x				x	x	x				x	x	x		x	x	x	x	x	x	x	16
45	Iterative Development	x	x	x	x		x	x	x					x	x		x		x	x	x	x	x	x	16
46	Continuous Integration	x	x	x	x			x		x	x	x	x	x	x			x	x	x	x	x		x	17
47	Kanban Board	x	x	x	x	x	x	x	x	x	x	x	x		x				x	x	x		x		17
48	User Stories	x	x	x	x	x	x	x	x	x						x	x		x	x	x	x	x	x	17
49	Burn-down Chart	x	x	x	x		x	x	x	x				x	x		x	x	x	x	x	x	x	x	18
50	Iteration	x		x	x		x	x	x	x				x	x	x	x	x	x	x	x	x	x	x	18
51	Acceptance Testing	x	x	x				x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	19
52	Daily/Scrum Meeting	x	x	x	x		x	x	x	x				x	x	x	x	x	x	x	x	x	x	x	19
53	Integration	x	x	x	x		x	x	x		x	x	x	x	x		x	x	x	x	x	x	x		19
54	Unit Testing or Developer Testing	x	x		x		x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	20
55	Backlog Grooming	x	x	x	x	x	x	x	x		x	x	x	x	x		x	x	x	x	x	x	x	x	21
56	Estimation	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	22
57	Agile Team	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	22
58	Backlog	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	23
How many agile practices used per respondent		30	33	25	29	11	24	46	42	20	15	15	15	21	27	9	26	22	42	31	39	39	45	21	
How many agile practices used per SDO		39		32		28		49		20	15			27		27		22	45		39	39	45	21	

Table 3
Wastes identified by our respondents.

Wastes identified by our respondents																								
ID	Summary of the interview results categorized to ten groups	Respondents																						Frequency
		A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	
W1	Processes: Long processes or too long to do it ; Autosar (a process tool) ; Useless reporting (as part of processes)							x	x		x	x	x				x					x			7
W2	Requirements and Design: Developing wrong requirements (req); Change in road map; not well-defined req for design team; Heavy useless architecture							x	x				x						x	x					5
W3	Management and Organizational aspects: Lack of Trust and request for monitoring every thing; lots of meetings/presentations; Non-homogeneous ways of working; Formal-approval of different phases														x	x			x	x					4
W4	No/or hard access to customer info w.r.t. feature usage; No access to hardware ("we discover things late")			x	x									x	x										4
W5	Pre-studies & Proof of concept which doesn't lead to development; pre-development activities/studies; Activities that do not make it to release such as systematization and analysis								x	x			x								x				4
W6	Third party, External suppliers (not easy to communicate or ask/force them to deliver what we need)													x	x									x	3
W7	Partially-done-work	x	x																						2
W8	Trouble-reports					x	x																		2
W9	Re-works; Re-inventing the wheels; No common design patterns												x				x								2
W10	Right feature but late; Late to market-window				x																				1

considered as waste. In general, not all the definitions of waste nor the provided examples/types of waste provided by the respondents in this study were the same or overlapping with the definitions and/or types of wastes found in state-of-art in literature, e.g. in [2,12,29,30]. For example, one respondent defined waste as *"anything that doesn't make it to the release, is waste"*, implying that product features/qualities not delivered were a waste of time to investigate, and/or to develop. Another respondent defined it as *"non-efficient way of working"*, focusing on the development process characteristics and not the product artifacts. On this line, one respondent defined waste simply as "bottlenecks", while other definitions included *"functions or features laying in the queue"*, and *"unused features which are the result of the Lack of enough knowledge/information about the customer and the products use"*. One respondent defined waste similar to the definition in [12], that is, waste is *"if something happens against the flow"*.

A summary of the identified wastes can be seen in Table 3, categorized into 10 categories (W1-W10). In Table 3, an "X" in a column means that this waste category was identified by the respondent, e.g. W10 was identified by respondent B2 only. The last column, "Frequency", shows the total number of respondents who identified each category of waste, e.g. seven respondents identified W1 independently.

Wastes related to processes (W1) were identified by more than 30% (7 out of 23) of the respondents. Examples of statements related to W1 were "*hard to learn the processes*" or "*unnecessary processes*" such as *"useless reporting"*. One reason why several respondents pointed out wastes related to processes may be related to the status of the studied SDOs. That is, the majority of the SDOs either stand in the phase of fully adopting agile processes or scaling it to the whole organization, further discussed below. Comparing the wastes mentioned under category W1 to the findings in [26], "processes" and "way of working" were mentioned and ranked amongst the important value aspects for the SDOs. This indicates that processes and way of working play an important role in the SDOs. However, it is vital to find the right balance and an optimized way of working, as processes can create more value if organized properly, but they can also generate Waste, if otherwise.

The second most mentioned waste category was related to requirements and design (W2), identified by five respondents. One

respondent stated, *"complex architecture highly oversized, they are built to support changes but with the first change they fall apart"*, while another respondent stated that *"we have the waste of developing wrong products… we need to design the right thing and just enough"*. This is similar to the experiences reported by Reinertsen [43], which showed that 95% of the participated managers admitted that they begin the design before all requirements are known, and the average product developer begins to design with 50% of the requirements. It was also found that managers normally avoid asking if the next phase of development activities have started, and developers do not mention that they have already moved on, and so this way of working might continue to happen over and over again [43]. Moreover, in a case study [33] comparable wastes were observed. One of the wastes is called "Building the wrong feature or product" which addresses the cost of creating the wrong feature/product. The other waste is called "Unnecessarily complex solutions" which is similar to the identified waste by our respondents (W2), caused by unnecessarily heavy architecture/design. The findings in [26] showed that "customer perceived quality", "product architectural value" and "functionality" were ranked amongst the most important value aspects for the SDOs in the study. This indicates that the SDOs are aware of the value of a good architecture and design, as well as the effect it has on overall product quality and customer satisfaction. However, the remaining and most important challenge is to find ways to minimize the waste and increase the efficiency through better and optimized designs/architecture.

Moreover, W1 and W2 (see Table 3) are issues that have the potential to produce various types of wastes by doing re-works instead of value-adding activities. Not well-defined requirements (W2) might reveal their waste-producing side faster than other factors such as Inefficient architecture design (W2), which may have long-term side effects. Authors in [12] put forward an impediment called "failure demand". "Failure demand" refers to a demand put on the resources of a system such as the organization and teams caused by its own failure, and leads to an impediment to flow through consuming time and effort, which could have been instead used on value-adding activities. Inefficient design (W2) has relation to what has been defined and identified as "failure demand".

Looking at the sources of waste identified in [29], we find correlations between the waste called "noise or information distortion" and our identified wastes in W1 and W2. "Noise or information distortion" can appear in two formations of time and space. The "time distortion" happens when information is not updated, recorded or remembered. If considering the wrong requirements to be a result of not updated or recorded information, we can identify similarities between the wastes in W2 and the waste described as "time distortion". "Space distortion" happens when actors are distributed across different levels or units of an organization [29]. If looking at processes as a connecting tool that passes the information from one context (e.g. developer) to another (e.g. manager), then this source of waste corresponds with the wastes identified in W1.

Management and organizational aspects (W3), No/hard-to-get-access-to customer information (W4), and Pre-studies and Proof of concept (W5) were all identified as waste, and each group identified by four respondents (Table 3). Amongst the aspects that were identified as non-value adding/waste by our respondents in W3, two activities of Unnecessary documentation or presentations, and Formal approval of different phases/processes were mentioned. However, there are activities, or regulatory mandates (e.g. compliance documentation or formal approval/approval processes), that are not directly value adding, but still necessary [44]. As if regulations are not adhered to, it might incur even more waste, or loss of opportunity for the SDO as reverse engineering or liability of broken contracts might be a factor [45]. Insights into the context of an organization are central to avoid sub-optimization pertaining to waste identification and elimination [10].

Another aspect mentioned in W3 was Non-homogeneous ways of working. This identified aspect may indicate challenges related to unsuccessful organizational management with regards to adopting new development methods. The responses from the respondents imply that it does not work well when an organization is not homogeneous in terms of way-of-working, or when some parts of the organization/teams follow agile methods while others do not. One respondent elaborated on this by stating that *"they ask us if the specification is ready for the subsystem and we answer yes, it is done for the sprint… But it is not ready by the end of the last sprint, because it is developed iteratively… the actual design review template has the question in it because it is designed based on a waterfall way of working and doesn't match the agile way of working…"*. To some extent, this may be attributed to agile transitional issues [55], but also to the fact that eventually, organizations end up using a hybrid development methodology (agile in combination with plan-driven) [46]. Looking at the findings in regards to important values for SDOs [26], "processes" and "way of working" were amongst the highest ranked value categories. This indicates that waste caused by Non-homogeneous ways of working, regardless of its root- cause, whether it is organizational management or the processes itself, needs to be addressed.

From a waste perspective, the overall goal of the agile implementation is important to communicate as it dictates what could and should be categorized as waste. Several respondents stated that some of the meetings and presentation sessions (related to W3), which are organized/requested by higher management, are considered as too many meetings, not very useful, or lacking clear agendas. One respondent explained, *"we have too many meetings… and an excess of people in the meetings, not all of the people is necessary to be there… maybe if we could sort people better and those who need to be there…"*. Many meetings or meetings running over time, has been identified and discussed in previous studies too, that might vary in different projects and organizations based on the cultural differences [33,47]. However, meetings are not necessarily waste, as long as they are well prepared, right people attending, and the right and needed knowledge is shared [48].

Lack of trust (W3) between management and development teams, or between different teams, was another waste expressed by the respondents. The respondents believe that the need to *"monitor everything"* is the result of a Lack of trust and it introduces waste and unnecessary activities by itself. One respondent explained why trust is important, *"trust is important, we need to trust each other and it is not a top-down process, [especially] when transforming towards agile/lean… there are constant challenges in change processes, roles are constantly changing"*. Trust is one of the critical factors for the success of agile methodologies [49], and the complexity of moving to a trust instead of control focus is a common problem in agile transformations [50]. The trust related statements in this study indicate that despite the emphasis on trust in agile methodologies, the SDOs have had problems in trusting their own organizations while adopting agile methods. However, they have come to this realization through experiences in changing how they work. Trust is linked to productivity and quality of the product – where monitoring was also not an enabler of trust [51]. Formal approval of the processes (W3) could also fall under the trust related issues, if the request for Formal approvals appears due to a need to control people, rather than regulatory mandates and other formal requirements. "Uncertainty" is one of the sources of waste identified in [29], which is also related to the identified aspects of W3. If people are not respected, the quality of decisions and the level of knowledge creation will be poor, as people, knowledge creation/utilization, and decision making are closely related [29].

No/hard-access-to-customer information (W4) was also observed as a source of waste by the respondents. The development team has either to rely on guessing and assumptions or to wait for information that leads to late deliveries. This could lead to the known wastes of *Extra-Features* (L7) or *Waiting* (L3) or even *Partially-Done-Work* (L6) (see Table 4), depending on the circumstances and decisions made. In [33], a waste called "Building the wrong feature" was identified and one of the observed causes of this waste was either not involving business

Table 4
Prioritization of the seven wastes of Lean.

Prioritization of the Seven Wastes of Lean																										
ID	The Seven Wastes Numbers in brackets (X,Y,Z): X = how many people chose it as most important waste Y= how many people chose it as second most important waste Z= how many people chose it as third most important waste	Respondents																								Rank
		A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	Average number (The lower the number, the more important the waste)
L1	Task-Switching (9,4,2)	5	5	1	1	4	1	1	1	3	1	5	2	2	2	2	6	4	1	1	7	5	1	3	2,8
L2	Defects (4,4,3)	3	3	2	2	6	5	4	4	1	2	1	3	4	4	6	7	1	7	2	5	4	4	1	3,5
L3	Waiting (2,4,6)	4	4	5	5	2	2	3	3	5	7	2	6	1	1	4	2	3	3	5	6	3	3	6	3,7
L4	Motion (1,5,5)	6	6	7	7	3	6	2	2	2	4	3	5	3	3	5	3	2	2	4	1	6	6	5	4,0
L5	Extra-Processes (4,2,1)	2	2	6	6	1	7	6	5	4	5	6	4	6	6	1	1	7	4	3	4	1	7	4	4,3
L6	Partially-Done-Work (2,1,4)	1	1	4	4	5	3	5	7	6	3	4	7	7	7	3	5	6	6	6	3	7	2	7	4,7
L7	Extra-Features (1,3,2)	7	7	3	3	7	4	7	6	7	6	7	1	5	5	7	4	5	5	7	2	2	5	2	5,0

stakeholders or slow stakeholder feedbacks. The identified waste category W4 is similar to the “wishful thinking” waste as suggested by [30]. "Wishful thinking" waste implies making decisions without the needed input/data or operating according to incorrect or insufficient controls that do not guarantee value delivery [30]. Another suggested source of waste, which also corresponds with our identified category of waste (W4), is “limited access to information” [29] and points out the fact that in order to make decisions, actors need to have access to the relevant information sources. So, no-access/hard-access to information might end to create other forms of wastes, that are the consequences of decisions that were either never made, or were made poorly due to insufficient information. Comparing the wastes mentioned above to the expressed value aspects of the study [26], “Customer relationships” and “Knowledge of feature value for customer” were mentioned amongst the important values. This indicates that practitioners are aware of how important it is to understand the customer needs, how the product is going to be used, which features are the most important, etc. However, facilitating effective first-hand access to this kind of information is still an important challenge which needs to be addressed.

Pre-studies and Proof of concept (W5) for items that do not make it to the final product were also considered as waste (Table 3); however, these could be unavoidable in some cases. Moreover, if seen from a revenue point of view, customers may pay for Pre-studies (W5). Thus, in addition to other benefits such as gaining new knowledge, Pre-studies can also bring revenue to the SDO and thus be value adding rather than just waste. Furthermore, a pre-study might be a pre-requisite for a non-inclusion decision of, e.g. a feature, that is developing the feature without a pre-study might result in waste, potentially more waste than incurred by the pre-study itself. In fact, this waste-creation potential has been mentioned in a study and described as “wishful thinking” [30]. “Wishful thinking” refers to making decisions without the needed input/data or operating based on incorrect or insufficient controls that do not guarantee the value delivery.

As shown in Table 3, three respondents identified wastes related to third party and external suppliers (W6) which were mainly related to the communication difficulties with external parties, or within distributed/global development. There were other studies too [29,33] that have mentioned comparable wastes to what is identified in W6. The source of waste described in [29] is “space distortion” and it happens “when actors are distributed across different levels or units of an organization” [29]. Waste identified in [33] is called “ineffective communication”, and similarly, one of its causes was observed to be “distributed stakeholders”. Lack of communication/efficient communication may not be a waste per se, but rather it may lead to other wastes such as *Waiting* (L3 in Table 4), *Motion* (L4 in Table 4), or *Extra-Features* (L7 in Table 4). Communication can be seen as overhead of sorts, but necessary overhead in some organizations (by overhead we mean those extra efforts that are put on a task which can be avoided by improving how the task is performed [52,53], for example improving the readability of a document e.g. by ordering it alphabetically, or changing a communication tool or ways of communication to a better one, etc.). After all, it is important to remember that “bad or insufficient communication” is what actually incurs waste [54].

In addition to the six above discussed wastes (W1-W6), four more categories of waste (W7-W10) were identified by either one or two respondents (see Table 3). W7 it is the same as *Partially Done Work* from the seven wastes of lean software development [2], which later in our study, we asked our respondents to prioritize amongst the seven original wastes of lean software development and the result of that prioritization (including Partially Done Work) is presented and discussed in Section 4.3.

The identified wastes W8-W10 (Table 3), such as Trouble-reports (W8), Re-works, Re-inventing the wheels and No common design patterns (W9), Right features but late, and Late to market-window (W10) could be symptoms of waste rather than wastes in themselves. For example, W9 and W10 are signs and warnings of ineffective knowledge management [63] and project management [64]. Mandic et al. [29] identified avoiding decision-making to be one of the sources of waste and associated it with the inability to inject value when there are no decisions made to transfer adequate knowledge to the software product or process. This is related to the identified waste category W10. W8, on the other hand, could indicate an excess of *Defects*, insufficient system tests, or other quality related issues. Hence, W8 would fall in the waste category of “correcting”, that is the redoing or scrapping due to a feedback [30]. Wastes stated in W9, such as Re-works and Re-inventing the wheels are related to the descriptions of the waste of Relearning [4] and also the identified wastes of “Knowledge loss” [33]. “Relearning” refers to the waste of “rediscovering something we once knew” [4], and “Knowledge Loss” is “the cost of members rolling off the team” [33], which both are related to the aspects identified in W9, such as “Re-inventing the wheels” as mentioned by one of our respondents. A forced Rework (W9), could be related to the waste of “Failure Demand” [12]

also. “Failure demand” refers to a demand put on the resources of a system such as the organization and teams caused by its own failure, and leads to an impediment to flow through consuming time and effort.

4.3. Prioritization of the seven wastes of lean (RQ1.1)

To aid agile/lean software development in finding and eliminating waste, Poppendieck and Poppendieck translate the seven manufacturing wastes to seven software development wastes – called the seven wastes of lean [2]. We introduced the respondents to the waste concept according to [2] and asked them to prioritize the seven wastes in relation to how important they are to control and eliminate. We opted to use the seven wastes of Lean as a basis for this as it is the most established explicit waste concept in software-intensive product development. We also wanted to compare the companies and respondents’ priorities and motivations as to their priorities. Using one base for the prioritization was important in this respect.

A scale between 1 and 7 was used, where 1 indicates the most important waste to control/eliminate and 7 the least important one. The results from the prioritization of the seven wastes (L1–L7) are shown in Table 4 (the column with each respondent's ID (A1–N) shows the individual prioritization of the seven wastes). To not to influence the respondents’ assessments before they had a chance to think about waste themselves, this was done after the respondents could contribute their own views and definitions of waste (as described previously).

The average score shows that Task-Switching (L1) is seen as the most important waste to control/eliminate. One of the respondents explained, “you are in the middle of developing a feature and then you have to take care of a defect, or because of a team member's special competent ". Another respondent associated Task-Switching with limited resources, many requirements and being constantly over-scoped. The consequence of the problem was expressed as: “this means that you sometimes might make decisions that are short-sighted". Another respondent added that even if it could be detrimental, Task-Switching might have some positive aspects as well, such as “if the person learns something”. On the other hand, 26% of the respondents did not consider Task-Switching among the most important waste to control/eliminate, as they ranked it between the fifth and seventh place (i.e. the least important one). We examined the prioritization results based on the respondents’ roles (PM and PO), and the domains. We could not find any correlation between the prioritized wastes and respondent role. However, looking at the domains, the respondents from the consultancy domain did not prioritize Task-Switching among the most important waste (on average), rather they ranked it as the third most important. Respondents who had the highest number (+ 40) of utilized agile practices in their processes (see Table 2), prioritized Task-Switching as the most important waste to control/eliminate. One possible interpretation could be that development organizations employing the most agile practices in their development processes have more issues with Task-Switching. Either in the form of noticing the issues more, or as a part of their process while using more agile practices - that is less planning and more prone to change, and so adapt to the changing environment. One might argue that this could lead to more reactions resulting in more switching, however, this is the matter of trade-off and to choose amongst the options with the highest priority/benefit. Though, this is not always an easy task to see or predict all the aspects, as it needs to take both the short-term and long-term effects of e.g. Task-switching into account, and to consider various aspects of it such as if it influences on developers’ productivity, customer satisfaction, availability of different skills within the teams, and so on. In a study that investigated the seven wastes of lean in a Kanban software development project [11], the result showed that Task-Switching was time-consuming and created some wastes. However, on the positive side, Task-Switching could also be used as an option to fill Waiting time.

After *Task-Switching* (L1), *Defects* (L2) was considered the second most important waste to control/eliminate, and only 26% of the respondents did not prioritize *Defects* amongst the first four important wastes. The reason could be that *Defects*, especially in the embedded-systems context, play an important role. Defects result in significant cost, but can also cause various levels of consequences depending on the product and the interdependencies between components and the interface between hardware and software [1]. One of the respondents elaborated on his/her given priority by stating that *“I don't see it [Defects] as waste in general, we have defects, we expect to have defects, and we have to have ways to take care of it.”* In lean, defects are seen as waste; however, one could argue that it can be seen as overhead and not waste. If the cost of finding/fixing a *Defects* is higher than the cost of not finding/fixing it, a *Defects* can be considered overhead rather than waste. Actually, from a lean perspective, hunting and fixing these kinds of *Defects* is the waste, not the *Defects* itself [10,52]. Thus, it is important to differentiate in relation to the relative cost of fixing and finding *Defects*, but also when and how in the development process a specific type of *Defects* is most cost-effective to identify [56,57].

Waiting (L3) was considered the third most important waste as seen in Table 4. *Waiting*, as a waste concerning lead-time reduction, has been seen in other SDOs too [8]. Similarly, in the part of our investigation reported in Section 4.2, (see Table 3, category W3) the time that was spent on *Waiting* for the formal approval, is what produced the waste in fact, and not the approval itself, according to the respondents. Delays, (as a result of *Waiting* (L3)), can also lead to increased costs. While *Waiting* has been observed as an inseparable part of the projects [11], and although some *Waiting* can be tolerated, the losses caused by *Waiting* should be eliminated if seeking for an efficient progress.

With regards to prioritization of the remaining wastes, *Motion* (L4) was ranked as the fourth most important waste to control/eliminate, followed by *Extra-Processes* (L5) and *Partially-Done-Work* (L6), as shown in Table 4. Although not many studies have been conducted on waste and their relative importance, however in [8], *Motion* and *Extra-Processes* were identified as wastes associated with lead-time issues. Waste of *Motion* (L4) could be due to collecting customer information that is needed for software development purposes, and as we could already see in Section 4.2, that No/hard access to customer information (W4) was amongst the first four categories of wastes identified by our respondents. Similar to the identified problem by our respondents, Ward [58] states that the most frequent waste in development is a waste of knowledge. Ward [58] divides knowledge waste into three categories which one is called “scatter”. “Scatter” is described as actions that disrupt the flow of knowledge, which could be due to communication barriers for example [58]. Seeking for the needed knowledge, regardless if it is due to communication barriers or any other reason, and if it is related to customer information or other internal information, can lead to waste of *Motion* (L4).

In regards to *Extra-Processes* (L5), we can see a relation and partial overlap between the wastes related to processes (W1) (Table 3) that was identified as number one category of wastes (in terms of the frequency of respondents who mentioned it) and the L5. However, when we later asked our participants to rank amongst the seven wastes, *Extra-Processes* (L5) is ranked as number 5, based on the average number. One explanation could be that our respondents could not see these two wastes as related to each other, and observed *Extra-Processes* only when they appear in more obvious forms such as unnecessary paper works or documentation.

Partially-Done-Work (L6) on the other hand, was identified and mentioned by our respondents in the exact same term, as we can see in Table 3. There too, Partially Done Work (W7) was not amongst the frequently mentioned wastes and was identified by only two of our respondents. It is suggested that Partially-Done-Work tends to become obsolete and you might not know if it eventually works [2]. However, the agile approach, on the other hand, suggests that you select few initial functions, start implementing them and you might leave them behind if the result is not satisfactory for customers [59]. If our respondents looked at Partially-Done-Work from such a viewpoint, it is

Table 5
Mapping of the wastes identified in this study with wastes from literature/other studies.

Wastes identified in this study / Seven lean wastes or other wastes from literature		Wastes identified in this study									
		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
		Processes: Long processes or too long to do it ; Autosar (a process tool) ; Useless reporting (as part of processes)	Requirements and Design: Developing wrong requirements ; Change in road map; not well-defined req for design team; Heavy useless architecture	Management and Organizational aspects: Lack of Trust and request for monitoring every thing; lots of meetings/presentations; Non-homogeneous ways of working; Formal-approval of different phases	No/or hard access to customer info w.r.t. feature usage; No access to hardware (" we discover things late")	Pre-studies & Proof of concept which doesn't lead to development; pre-development activities/studies; Activities that do not make it to release such as systematization and analysis	Third party, External suppliers (not easy to communicate or ask/force them to deliver what we need)	Partially done work	Trouble-reports	Re-works; Reinventing the wheels; No common design patterns	Right feature but late; Late to market-window
The seven original wastes of lean software development [2]	Task Switching (L1)										
	Defects (L2)								Partly related		
	Waiting (L3)				Partly related						
	Motion (L4)				Partly related						
	Extra Processes (L5)	Related				Partly related					
	Partially Done Work (L6)							Overlap			
	Extra Features (L7)		Partly related								
Other defined or identified wastes from literature that were related to identified wastes from this study	Relearning [4]									Related	
	Wishful thinking [29]				Partly related						
	Correcting [29]								Partly related		
	Uncertainty [28]			Partly related							
	Limited access to information[28]				Related						
	Space distortion [28]	Partly related					Related				
	Time distortion [28]		Partly related								
	Avoiding decision making [28]										Related
	Failure demand [11]						Related			Related	
	Unnecessary complex solutions [32]		Partly related								
	Building the right feature or product [32]		Partly related		Partly related						
	Ineffective communication [32]						Related				
	Mismanaging the backlog [32]							Partly related			
	Knowledge loss [32]									Related	

not surprising that they did not prioritize it amongst the most important wastes.

A similar waste, which has relation to Partially Done Work (W7/L6), was observed in another study [33]. There, it was called "Mismanaging the backlog", however, we found some similarities between some of the mentioned root causes of their observed waste and W7/L6, such as "not enough ready stories" and "delaying testing".

As seen in Table 4, *Extra-Features* (L7) was seen as the least important waste, based on the ranking by respondents of our study, to control and eliminate. One of the respondents reflected on this, *"we have always a problem to get to the level that was requested…and if that, I would not see it as a waste I would see it as a bonus"*. On the other hand, 26% of the respondents prioritized *Extra-Features* among the top three most important wastes to control/eliminate.

We examined the data for possible correlations between the prioritization of *Extra-Features* and roles and/or domains. Although we did not find any correlation in regards to roles, an interesting result was seen when looking at the domains. The consultancy domain, unlike the other three domains, prioritized *Extra-Features* as the most important waste. The respondents from the consultancy domain argued that *Extra-Features* were those features that were added without a proven need or valid hypothesis. Thus, adding *Extra-Features* significantly slows down both the feedback and the revenue generation. Many organizations fail to see the hidden economic costs of *Extra-Features* [12]. The highly prioritized *Extra-Features* may indicate a higher level of responsibility towards revenue and cost aspects amongst the consultancy organizations and thus has resulted towards a better understanding of the importance of *Extra-Features* [60]. It may also show that consultancy-based organizations have more sensitivity towards agile concept and practices such as time-boxing. Time-boxing in agile methods helps towards focusing on the customer and it reduces gold-plating [61]. Despite the available knowledge about excessive software development practices (and extra features as a part of it) that are risky and might have a variety of negative consequences on the project schedule, quality and costs, the causes of excessive practices are diverse in nature. Various factors play a role, e.g. which stakeholder is at fault, in which project phase it takes place, and whether it can be avoided or not [60]. In some cases, *Extra-Features* are not considered waste as reported by Ikonen et.al [11], instead, were seen as usability improvements. Even if the *Extra-Features* could increase the chance of defect/malfunction and may consume time, the *Extra-Features* were beneficial for the customer of the project [11]. Avoiding gold-plating extra-features, but weighting this towards increased value for users, or for that matter, internal aspects such as maintenance is a difficult trade-off to a fast-paced good-enough release of agile environments.

4.4. Overall reflection and recap

Wastes and some of the other waste-producing activities identified by our respondents, directly or indirectly, relate to the organization and the way of organizing processes. Looking at W1 and W3 (see Table 3), half of the respondents (48%) expressed wastes merely related to processes or organizational aspects of the processes. Since most of the SDOs in this study are still in the processes of fully adopting agile and scaling it to the whole company/organization, this could be the result of a non-homogeneous organization in terms of ways-of-working [62]. Agile methodologies emphasize collaboration and are based on value-focused strategies [13]. Thus, for the agile teams, avoiding non-value-adding activities and removing obstacles towards value, echoes a natural part of the adoption. However, based on the results from the investigated SDOs in this study, to differentiate value-adding from waste does not appear as a completely clear or easy process.

As discussed in Section 4.2, we asked the respondents what they considered waste, and we did not steer the respondents by giving them the traditional seven wastes of lean. The stated different types of wastes, difficulties, and matters leading to waste, and also indirect waste creating activities, resulted in a total of 10 categories of wastes (W1-W10). From the 10 categories, only one waste, Partially-done-work (W7), is exactly as *Partially-Done-Work* (L6 in Table 4) from the seven original wastes of lean software development. However, there are of course other partial overlaps between the wastes stated by respondents and the seven wastes of lean, e.g. W1 is related and partly covered by definition of *Extra-Processes (L5)*. Aspects discussed in W2 are somewhat related to *Extra-Features(L7),* W4 is somewhat related to *Waiting (L3)*, and parts of description matches with the definition of *Motion(L4)*. Parts of aspects mentioned in W5 are related to *Extra-Processes(L5),* and the same applies to Trouble-reports (W8) which is partly related to *Defects (L2)*.

A summary of the relations between the wastes identified and presented in our study with the seven wastes of lean [2] is shown in Table 5. Also, the noticed similarities and relations between the wastes identified in our study and other wastes observed/suggested in other studies can be seen in Table 5. If our identified wastes were related to either one of the seven lean wastes or wastes from other literature, the crossing cell between the two wastes states "Related". If not all the aspects, but part of the aspects discussed in our identified waste category was related to the seven wastes of lean or other literature, the crossing cell shows "Partly related". If the two stated wastes were exactly the same, it is written "Overlap".

4.5. Waste reduction and elimination (RQ2)

The majority of the respondents rely mainly on traditional process assessment and improvement activities for waste reduction. More than half of the respondents (13 out of 23) stated activities that are categorized under Processes (A1), as illustrated in Table 6, Part 1. Looking at Table 6, "X" denotes a waste reduction category mentioned by a respondent, e.g. A9 was mentioned by respondent D2. The last column "Frequency" shows the total number of respondents who identified a category, e.g. 13 respondents identified A1.

The waste reduction activities (Table 6) were mostly related to various practices or methods of agile or lean processes. One of the respondents stated *"Lean transformation"* as an activity towards waste reduction and elimination. Thus, assuming that changes in the process and applying certain practices would reduce waste, while actually identifying and targeting waste as a concept directly, was not applied as the main activity. Two of the respondents stated the use of tools (CASE tools) as waste reduction activities. Another respondent identified Kanban as a good "tool" for identifying waste, stating *"Kanban is a good thing for identifying the waste… those that are blocking the flow, process"*. Similarly, results from other studies show that Kanban can be an effective method in visualizing and organizing current work, and also perhaps in striving for minimizing the non-value-adding work. However, Kanban does not prevent waste from creeping in [11]. There exist other methods such as Value Stream Mapping (VSM) [4] that have shown utility in waste identification [8,10]. VSM methods are centered on value-adding and non-value-adding elements and the separation between the two. In recent years, due to its value-focused policy, the capacity to analyze and design flows at the system level and across multiple processes; VSM has attracted great interests amongst software engineering community [10]. Thus, for all the SDOs that have value creation and waste reduction as their goal, VSM may offer a targeted way to identify waste.

In addition, the results in this study show that, except for a few stated activities such as Test-improvements (A6) and Risk reduction activities (A8), which are initiated at project management level, the majority of the waste reduction activities are happening at the team or individual level. A supporting, and interesting statement came from two respondents from the same SDO. The respondents stated that their waste reduction activity is *"to make the wastes visible to the upper managers"*. So, expressing that waste identification is up to the individual team members, however, in order to take proactive waste identification

Table 6
Waste reduction activities and suggestions.

Waste reduction activities and suggestions mentioned by our respondents

Part 1

ID	Waste reduction activities mentioned by our respondents (Summary of the interview results)	Respondents																					Frequency
		A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	
A1	Processes: Working more in agile way; Lean-transformation; Kanban; Retrospectives; Time-planning; Sprint-planning; Backlog prioritization through communication with customer; Visibility; Team-planning; Definition -of-done that comes with the agile way of working; Optimizing the processes	x	x				x	x	x		x	x				x		x	x	x			x	x	13
A2	Tools: Lean tools which helps to see and reduce waste; other documentation tools																		x					x	2
A3	Hundred and small initiatives that you do all the time to reduce waste and define impediment	x	x																						2
A4	Make it visible to managers										x	x													2
A5	Communicate with teams and force them to deliver to reduce the waiting time or to get what is needed													x											1
A6	Test-improvements																					x			1
A7	The concept of team itself ("Working as teams makes it easier to ask for what we want/need")																			x					1
A8	Project management activities: Risk reduction activities; Time-planning; Sprint-planning																							x	1
A9	Reducing the starting time								x																1

Part 2

ID	Suggested awareness or future plans, mentioned by our respondents (Summary of the interview results)	Respondents																					Frequency
		A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	
P1	Talking about waste or saying that every one should be aware of it	x	x				x											x			x				5
P2	To become better at considering what is value (and for customer) and use it as a guide line to reduce waste			x	x																				2
P3	Knowing relative value of performing tasks							x																	1
P4	Using similar-terminologies and common-language for awareness purposes and reduction activities					x																			1

and elimination action, explicit methods could largely benefit SDOs as well as enabling waste to become visible to managers [10]. In lean, the active identification and elimination (or avoidance) of waste on all levels (from individuals developing to top management) is central. To avoid waste and sub-optimization, waste identification and elimination should be coordinated both locally and centrally, as well as top-down and bottom-up.

We also asked the respondents how they could improve or change the way waste was identified and handled in their SDO currently (see Table 6, Part 2). Out of nine respondents who expressed their ideas, five identified a need to Talk about waste and increase the awareness of waste (P1). To facilitate waste awareness, one respondent suggested using Similar-terminologies and a Common-language (P4). Moreover, two respondents stated that "the concept of value" and "what is value for customer" needed to be explored in order to be used as a guideline for waste reduction activities (P2). Overall, the concept of value and

communicating and establishing a joint understanding of value was discussed as a prerequisite for waste identification, and "*putting a price on tasks*" and the relative value of performing a task (P3) was also suggested in the same regard. Looking into value as a concept, various aspects from internal and external values need to be taken into consideration to avoid sub-optimization [10]. For example, removing a documentation activity in a project can be seen as waste removal (as it may not be needed by the project), but a sub-optimization from a product perspective as e.g. product managers may need this documentation activity to provide decision support for future planning. In this case, something that is considered as waste on the project level, if removed as waste, generates a sub-optimization as all value aspects of the activities were not seen in advance [65,66]. Thus, there is a need to separate waste from overhead [10]. Furthermore, it has been suggested that it is likely that most of the seven wastes will become apparent in all software development projects if process assessment uses them as a focus for analysis [11]. Thus, what remains important is that the focus of analysis might be revised and adjusted continuously, either based on different time and stages of the development processes or different weight or effect of wastes, or other relevant consideration and priorities such as company/domain specific aspects.

4.6. Waste measurement (RQ2)

To determine which parts of the SDO, or the development process, that have the potential for improvement/waste reduction, it is important to have measurements or evaluations depicting of the current status. Accordingly, we asked the respondents if, and how, they measure waste in their SDO respectively. The majority (61%) of the respondents do not have any measurement in place or do not measure waste, as shown in Table 7. As there are metrics already used and could be relevant as indicators of waste measurement [4], e.g. cycle time or defects, the result shows that the importance of waste measurement and the effect of waste on the overall productivity of the organizations are not fully recognized. Alternatively, that waste as a concept is hard to grasp thus hard to measure or apply current measures to, especially that some waste is potentially recognized already. One of the respondents elaborated on the difficulties of measurement by stating that, "*it (measurement) is hard because we need a baseline to measure. You have to come up with some sort of KPI (Key Performance Indicator) in order to do it*" . However, in order to create a "baseline", organizations need to start to measure at some point and waiting for a baseline that does not exist and it is dependent to a measurement itself, only introduces a double-blind and the so-called catch-22 [66]. Moreover, it has been suggested that companies, in particular within the embedded development domain, can build their own history database with baselines for estimation purposes and quality planning [1], and similarly, the use of some custom metrics amongst the agile teams has been reported [67]. It is essential to actually define what an organization intends to measure and why, in order to define useful metrics and avoid introducing unnecessary measurement cost [68].

In addition, we asked the respondents to express any type of indicators that they think could aid in seeing the effects of waste and ultimately aid in the reduction or elimination of waste. According to one of the respondents, the *"team's failure to deliver in time"* could be seen as an indicator. The respondent further explained, *"when somebody has been dragged from our team, we can show at the end of the sprint that this is the amount of work that has not been done, because of this person who was dragged out to do other things"*.

As shown in Table 7, Defects (M1), Lead-time (M2), Team-velocity (M4), and Trouble-reports (and the reduced cost of it) (M5) are the main measurements or indicators stated by the respondents as potential waste indicators. In previous research, some of the identified indicators such as lead-time [34] and workload analysis [34], which is similar to team-velocity [69], are in line with what has been identified by our respondents.

The result of a systematic literature review study on metrics in agile and lean software development organizations [67] shows that requirements engineering, specification, and design are seldom measured in such organizations and only as part of "the whole development cycle". As discussed in [67], the explanation could be that these processes are not considered important in agile software development, or there is no need for a separate measurement and only as part of the whole development cycle. However, the result of our study shows that these assumptions might not be always true, as we can see in Table 3 under the second most frequently stated category of wastes called W2. Our respondents have observed and considered requirements and design as important aspects and leading towards creating more waste if not taken care of in the right time and correct way.

Some finding reveals that in agile organizations the target of measurements are the product and process, but not the people [67]. Similarly, we did not receive any response indicating the use of such measurements, or as part of their suggested future plans. One explanation could be that agile software development relies on the assumption that teams are capable enough in such ways that the team members can improve themselves without any metrics. However, measuring and taking human aspects into account, in terms of different personalities, collaboration, communication, and other human behavioral factors, has been recognized as central in terms of impact on efficiency and productivity [33,49,67,70,71]. Moreover, in [70] it has been highlighted that some of the common agile practices, in addition to their benefits, can also cause some frustration and apprehension amongst team members, and so from the management perspective it is important to be aware of that, and address it where needed. Human factors become even more important when the product and teams are large [67].

Overall, the result and the statements by the respondents, emphasize the need to establish better and more holistic measurements and indicators for waste. The usefulness of measurements [9] and the need

Table 7
Waste measurements.

Measurements mentioned by our respondents																									
ID	Measurements (Summary of the interview results)	Respondents																							Frequency
		A1	A2	B1	B2	C1	C2	D1	D2	E	F1	F2	F3	G1	G2	H1	H2	I	J1	J2	K	L	M	N	
M1	Defects	x	x																				x		3
M2	Lead-time							x	x																2
M3	Team's failure to deliver in time (due to task switching) can be seen as a measurement itself																		x	x					2
M4	Team-velocity																					x			1
M5	Trouble-reports (and reduced cost of it)						x																		1

to provide better estimates in embedded software development to enable adequate benchmarking data has been raised in previous studies [1]. People have difficulties in understanding what an impediment (waste) is, how to see it, how to measure and quantify their impact, and how to reduce waste in general [12]. Yet, how to measure waste in such a way that all the various influential factors are considered, what the best indicators are, and how to create metrics that can aid SDOs to measure both the effect of wastes and also the effect of waste elimination activities, requires further investigation and research.

5. Threats to validity

Like any other studies, there exist threats to validity that are worth discussing. To discuss the possible threats, we have followed the four perspectives of validity threats presented in [38].

Construct validity. The construct validity reflects on to what extent the studied operational measures correlate with the constructs of its research questions [38]. The variables in our research were measured through open-ended interview questions where the respondents were asked to express their own thoughts. In order to minimize the risk that the respondents may have misunderstood the researchers and/or the questions asked, due to e.g. the use of different terms and references, peer debriefing was used [38]. Peer debriefing suggests carrying out the research by a group of researchers instead of one, which may lower the risk of being biased towards one researcher's views and perceptions. Thus, peer debriefing helps in asking the questions in the right way and using proper terms that were understood by the respondents. Moreover, we involved research colleagues, who had the experience of working in a similar research study, for their feedback during the research design process. We have also used triangulation for the data collection or to be more precise for the selection process of the interviewees. Triangulation is when the researchers take different perspectives on an issue under study [36]. Data triangulation was implemented by incorporating subjects from various software development organizations with different domains (see Table 1) and by combining the data from two groups of the respondents with respect to their roles. This widens the perspective of the collected data to some extent and reduces the risk of acquiring only one specific view [36]. Another threat is related to the presence of a researcher during the interviews. The respondents may have felt exposed to express their views, which may have influenced their answers. To minimize this threat, the respondents were guaranteed of anonymity and were assured that the answers from the interviews were only to be used by the researchers. A third threat to construct validity is related to the selection process of the companies and the respondents. We selected the companies within our industrial network as it would provide us with the necessary trust of the companies. Since the respondents were not fully randomly sampled, there is a threat to selection bias that the proposed respondents were only those who had an interest or positive attitude towards lean and waste concepts. However, the respondents were chosen based on their roles and through a “gate-keeper” at all the companies.

Internal validity. Internal validity is seeking to establish a causal relationship whereby certain conditions are believed to lead to other conditions [38]. If the effect of a treatment on an outcome has been wrongly concluded, without considering a third factor (which is the cause of the effect), the study has a low degree of internal validity [40]. According to Yin [40], internal validity is mostly for explanatory and causal studies rather than descriptive or exploratory studies. However, during the data analysis, triangulation was used to increase the internal validity of this study, through the use of literature, in order to confirm or refute the findings from the case study.

External validity. The external validity is concerned with the ability to generalize the results, and to what extent the findings can be transferred to other cases [38]. However, the objective of qualitative studies is seldom to generalize beyond the actual setting and its focus is rather on explaining and understanding the investigated phenomena. While there are researchers, e.g. Hagg and Hedelund [72] who argue against the generalizability of case studies, Runeson and Host [38] suggest that the intention of case studies is to enable generalization, where the findings extend to cases with common characteristics, and therefore the outcomes are relevant. With respect to the common characteristics, all the SDO in this study were using agile software development process and developed embedded systems. Also, understanding the investigated phenomena in one situation may help in understanding other situations. However, in order for the findings to be considered generalizable, it must be possible to compare and relate the context and characteristics of the case companies in this study to other situations and the context of interest. To help the reader to understand the relevance of the study, and perhaps to compare with other settings, the characteristics of each company and each interviewee in this study are presented in Table 1 and the use of agile and lean practices per each SDO are presented in Table 2.

Reliability. These threats examine the degree to which the analysis is dependent on the specific researcher, e.g. if the interview questions or the coding process are not clear [38]. In order to improve reliability, an interview guideline was created to make sure that all relevant aspects were covered in all interviews. Moreover, in order to decrease the risk of wrong interpretations while taking the notes, all interviews were recorded. In addition, a member checking technique was used during the interview session for the prioritization of the seven wastes of lean. Also, the paper sheets with the written definitions of each waste helped in assuring that the respondents prioritized the seven wastes with the same understanding and consistent definitions of each waste.

6. Conclusions

We studied the concept of waste and probed professionals for definitions on how they viewed, identified, and ultimately worked towards minimizing waste. Overall, waste is a concept that is always relevant, and waste-minimization is a part of the core “philosophy” of both lean and agile. In more “traditional” plan-focused development methodologies, the main tool for removing waste/issues is process assessment and improvement. In agile/lean this is still true, but the major quality assurance of “way-of-working” is the learning activities continuously achieved, the awareness, and the identification and elimination of waste, using activities such as retrospectives [15]. Thus, it is important to not only consider both bottom-up and top-down waste reduction activities, but also to “optimize the whole”, the whole organization and the whole product the whole time [3], including but not limited to all stakeholders, managers, teams and individuals, as well as design and development, test and maintenance parts. Since people are the main component of the software development processes, in order to investigate waste and its root causes and to establish waste reduction and improvement strategies, it is important not to neglect people and human-related factors [12,67]. Furthermore, for waste reduction strategies, both the short-term and the long-term perspectives need to be considered to achieve a sustainable return on investment [10].

Waste is a relative concept, and one organization's waste can be another organization's necessity. However, awareness of, and the definition, discussion, and joint understanding within an SDO is a pre-requisite for waste identification and reduction. In our study, we see that practitioners mention many types of “waste”, but there is little agreement; waste types and what constitutes waste identified in state-of-the-art is not used as a base. This is made worse by the fact that many organizations that are agile/lean, work in a hybrid manner in fact, where islands of agile exist nestled in a traditional plan-driven organization. This generates a mix of waste concepts. For example, parts of the organization want to “know the requirements” before design, or have a “good” design before development starts. This seems to be at odds with the “fail fast”, “fail often” and “learn as you go” idea of agile/lean, and in itself introduces waste, as the interface between the agile and non-agile parts do not work. Artifacts like pre-studies that are not

developed in time by non-agile parts of the organization, are delivered too late for a sprint and thus are never used, in essence, the effort and the artifact itself *becomes* waste. There is also a lack of trust that creates (for the agile parts) overhead in terms of meetings, coordination, and reporting, as well as monitoring to strive for control. In this type of environment, it is not hard to understand why it is difficult to share an understanding of waste, not to mention the ability to measure and remove it.

It is also possible to see it the other way around. A pre-study that is performed can result in a product or feature not being developed. This might save the development organization significant costs as the "wrong" thing is avoided, to begin with. The pre-study thus is an investment in avoiding waste and should thus not be seen as waste initself. Seeing the pre-study as waste can be seen as a sub-optimization in thinking [66]. However, if the pre-study of this example is done resulting in no action taken (even if the pre-study says no you do it) then the pre-study is most certainly waste and so are the subsequent mistakes, that could have been avoided. Another example of overhead is coordination. Coordination between teams, for example, might not lead to direct value addition, but not coordinating might lead to defects and misunderstandings, which in turn can lead to even more waste.

What exactly constitutes waste, and how to balance activities and differentiate between waste and overhead is a very complex undertaking that has to be performed across organizational levels to avoid sub-optimization [53,66]. However, a base starting point for waste identification and removal is to discuss and create a common understanding of what constitutes waste in an organization. Even if not reaching an agreement, it can be still considered a success, as the concept is discussed and awareness is raised. Following this, monitoring and measurement and subsequent pre-emptive measures can be taken to avoid the generation of waste – which should be a core goal of any learning activity of agile and non-agile alike. The multiple case study presented in this paper was motivated by ten years of actively working with many agile and agile-hybrid companies, and the insight that waste, and organizations' understanding, and handling of waste was unclear and often non-existent – this knowledge was, however, a "gut feeling" and not substantiated. This study shows that waste is a concept recognized and considered important by practitioners themselves, yet very few explicit activities are performed to discuss and reach a common understanding of waste. Further, almost none of the organizations have any measurements or focus on identifying waste.

6.1. Future work

As agile/lean is reaching a status of de-facto industry standard, the need for research into waste as a concept, and methods and principles for handling waste, in both agile/lean and hybrid organizations are needed. Practitioners in the industry can use lean's definitions or the ones by e.g. Poppendieck and Poppendieck [2,4], as a base to start the discussion. Software engineering research, on the other hand, has to continue and research the area and develop methods, practices, and principles that enable the measurement, identification, and effective and efficient removal of waste as a part of agile/lean product development methodology. One area that needs further research is the difference between waste and overhead, especially when it comes to identification and measurements of wastes. In addition, another area for future research is the identification of domain-specific vs. common wastes. This could be done by conducting in-depth investigations within different domains, as well as considering different types of products/services and/or various customers within each domain.

References

[1] C. Ebert, C. Jones, Embedded software: facts, figures, and future, Computer 42 (4) (2009) 42–52. Apr..

[2] M. Poppendieck, T. Poppendieck, Lean Software Development: An Agile Toolkit, Mass.: Addison-Wesley Professional, Boston, 2003.

[3] A. Shalloway, G. Beaver, J.R. Trott, Lean-Agile Software Development: Achieving Enterprise Agility, 1 ed., NJ: Addison-Wesley Professional, Upper Saddle River, 2009.

[4] M. Poppendieck, T. Poppendieck, Implementing Lean Software Development: From Concept to Cash, Pearson Education, 2007.

[5] J. Morgan, J.K. Liker, The Toyota Product Development System: Integrating People, Process, and Technology, Taylor & Francis, 2006.

[6] X. Wang, K. Conboy, O. Cawley, 'Leagile' software development: an experience report analysis of the application of lean approaches in agile software development, J. Syst. Softw. 85 (6) (2012) 1287–1299 Jun.

[7] J.P. Womack, D.T. Jones, Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Revised and Updated, 2nd ed., Productivity Press, New York, 2003.

[8] S. Mujtaba, R. Feldt, K. Petersen, Waste and lead time reduction in a software product customization process with value stream maps, in Proceedings of the Australian Software Engineering Conference (ASWEC), 2010, pp. 139–148.

[9] K. Petersen, C. Wohlin, Measuring the flow in lean software development, Softw. Pract. Exp. vol. 41, (9) (Aug. 2011) 975–996.

[10] M. Khurum, K. Petersen, T. Gorschek, Extending value stream mapping through waste definition beyond customer perspective, J. Softw. Evol. Process. 26 (12) (2014) 1074–1105.

[11] M. Ikonen, P. Kettunen, N. Oza, P. Abrahamsson, Exploring the sources of waste in Kanban software development projects, Proceedings of the Software Engineering and Advanced Applications (SEAA), 2010, pp. 376–381.

[12] K. Power, K. Conboy, Impediments to flow: rethinking the lean concept of 'waste' in modern software development, Agile Processes in Software Engineering and Extreme Programming, Springer International Publishing, 2014, pp. 203–217.

[13] A. Manifesto, accessed December http://agilemanifesto.org, (2013) accessed December.

[14] M. Lindvall, et al., Agile software development in large organizations, Computer 37 (12) (2004) 26–34 Dec.

[15] M. Cohn, Succeeding with Agile: Software Development Using Scrum, 1st ed., Addison-Wesley Professional, 2009.

[16] K. Conboy, L. Morgan, Beyond the customer: opening the agile systems development process, Inf. Softw. Technol. vol. 53, (5) (2011) 535–542.

[17] M. Korkala, P. Abrahamsson, Communication in distributed agile development: a case study", Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, 2007, pp. 203–210.

[18] D. Mishra, A. Mishra, Complex Software Project Development: Agile Methods Adoption, J. Softw. Maint. Evol. Res. Pract. 23 (8) (2011) 549–564.

[19] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, J. Syst. Softw. 82 (9) (2009) 1479–1490.

[20] J. Iivari, N. Iivari, The relationship between organizational culture and the deployment of agile methods, Inf. Softw. Technol. vol. 53, (5) (2011) 509–520.

[21] R. Hoda, J. Noble, S. Marshall, Developing a Grounded Theory to Explain the Practices of Self-organizing Agile Teams, Empirical Softw. Engg. 17 (6) (2012) 609–639.

[22] S.W. Ambler, Scaling agile software development through lean governance, Proceedings of the 2009 ICSE Workshop on Software Development Governance, Washington, DC, USA, 2009, pp. 1–2.

[23] K. Petersen, C. Wohlin, Software process improvement through the Lean Measurement (SPI-LEAM) method, J. Syst. Softw. 83 (7) (Jul. 2010) 1275–1287.

[24] Z. Racheva, M. Daneva, K. Sikkel, Value creation by agile projects: methodology or mystery? Proceedings of the Product-Focused Software Process Improvement, Springer, 2009, pp. 141–155.

[25] M. Khurum, T. Gorschek, M. Wilson, The software value map — an exhaustive collection of value aspects for the development of software intensive products, J. Softw. Evolut. Process. 25 (7) (2013) 711–741 Jul..

[26] H. Alahyari, R. Berntsson Svensson, T. Gorschek, A study of value in agile software development organizations, J. Syst. Softw. 125 (2017) 271–288 Mar.

[27] J.K. Liker, D. Meier, The Toyota Way Field Book, McGraw Hill Professional, 2005.

[28] D.J. Anderson, D.G. Reinertsen, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010.

[29] V. Mandić, M. Oivo, P. Rodríguez, P. Kuvaja, H. Kaikkonen, B. Turhan, What is flowing in lean software development? Lean Enterprise Software and Systems, Springer, Berlin, Heidelberg, 2010, pp. 72–84.

[30] M.V.P. Pessôa, W. Seering, E. Rebentisch, C. Bauch, Understanding the waste net: a method for waste elimination prioritization in product development, in: S.-Y. Chou, A. Trappey, J. Pokojski, S. Smith (Eds.), Global Perspective for Competitive Enterprise, Economy and Ecology, London: Springer London, 2009, pp. 233–242.

[31] H.L. McManus, Product development value stream mapping (PDVSM) manual, release 1,0, Massachusetts Institute Of Technology, 2005 The Lean Aerospace Initiative.

[32] M. Braglia, M. Frosolini, F. Zammori, Uncertainty in value stream mapping analysis, International Journal of Logistics Research and Applications vol. 12, (6) (Dec. 2009) 435–453.

[33] T. Sedano, P. Ralph, C. Péraire, Software development waste, *Proceedings of the 39th International Conference on Software Engineering*, Piscataway, NJ, USA, 2017, pp. 130–140.

[34] K. Petersen, A palette of lean indicators to detect waste in software maintenance: a case study, Agile Processes in Software Engineering and Extreme Programming, Springer, 2012, pp. 108–122.

[35] C. Robson, Real World Research, Blackwell, 2002.

[36] U. Flick, An Introduction to Qualitative Research. SAGE 2009.

[37] M.Q. Patton, Qualitative Research & Evaluation Methods, SAGE, 2002 N.B. Moe, T.

Dingsøyr, and T. Dybå, A teamwork model for understanding an agile team: a case study of a Scrum project Inf. Softw. Technol., vol. 52, no. 5, pp. 480–491, May 2010.
[38] P Runeson, M. Host, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (Apr. 2009) 131–164.
[39] M Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empir. Softw. Eng. 16 (2011) 365–395.
[40] R.K. Yin, Case Study Research: Design and Methods, 5th edition, SAGE Publications, Inc, Los Angeles, 2013.
[41] Agile Glossary, accessed December https://www.agilealliance.org/agile101/agile-glossary/, (2013) accessed December.
[42] Subway Map to Agile Practices, accessed December https://www.agilealliance.org/agile101/subway-map-to-agile-practices/, (2013) accessed December.
[43] D.G. Reinertsen, The Principles of Product Development Flow: Second Generation Lean Product Development 62 Celeritas Redondo Beach, 2009.
[44] H.-L. Abdelwahab, Regulatory Compliance and its Impact on Software Development, Software Compliance Research Group Department of Electrical and Computer Engineering, 2015.
[45] V. Jéssyka, et al., Integration between requirements engineering and safety analysis: a systematic literature review, J. Syst. Softw. 125 (2017) 68–92.
[46] R. Turner, A. Jain, Agile Meets CMMI: Culture Clash or Common Cause? Proceedings of the Extreme Programming and Agile Methods — XP/Agile Universe 2002, 2002, pp. 153–165.
[47] R. Muller, K. Spang, S. Ozcan, Cultural differences in decision making in project teams, Int. J. Manag. Proj. Bus. 2 (2009) 70–93.
[48] H. Gustavsson, J. Axelsson, Improving the system architecting process through the use of lean tools, 2010, PICMET '10 - Portland International Center for Management of Engineering and Technology.
[49] A. Cockburn, J. Highsmith, Agile software development, the people factor, Computer 34 (11) (2001) 131–133 Nov.
[50] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of Migrating to Agile Methodologies, Commun. ACM 48 (5) (May 2005) 72–78.
[51] N.B. Moe, D. Smite, Understanding a lack of trust in global software teams: a multiple-case study, Softw. Process 13 (3) (2008) 217–231 May.
[52] L.-O. Damm, L. Lundberg, C. Wohlin, Faults-slip-through—a concept for measuring the efficiency of the test process, Softw. Process Improv. Pract. 11 (1) (2006) 47–59 Jan.
[53] C.R. Sánchez, A. Lamersdorf, A.F.V. Torre, J. Münch, D. Rombach, Estimating the effort overhead in global software development, Proceedings of the 2010 Fifth IEEE International Conference Global Software Engineering (ICGSE 2010), Los Alamitos, CA, USA, 2010, pp. 267–276.
[54] J. Pernstål, R. Feldt, T. Gorschek, The Lean Gap: A Review of Lean Approaches to Large-scale Software Systems Development, J. Syst. Softw. 86 (11) (Nov. 2013) 2797–2821.
[55] T.J. Gandomani, H. Zulzalil, A.A.A. Ghani, A.B.M. Sultan, M.Z. Nafchi, Obstacles in moving to agile software development methods; at a glance, J. Comput. Sci. 9 (5) (2013) 620.
[56] D. Lars-Ola, L. Lundberg, C. Wohlin, Faults-slip-through-a concept for measuring the efficiency of the test process, Softw. Process Improv. Pract. 11 (1) (2006) 47–59.
[57] A. Wasif, et al., Search-based prediction of fault-slip-through in large software projects, Proceedings of the 2010 Second International Symposium on Search Based Software Engineering (SSBSE), IEEE, 2010.
[58] A.C. Ward, Lean Product and Process Development, Mass.: Lean Enterprise Institute, Cambridge, 2007.
[59] B. Meyer, Agile! The Good, The Hype and The Ugly, Springer Science & Business Media, 2014.
[60] O. Shmueli, B. Ronen, Excessive software development: practices and penalties, Int. J. Proj. Manag. vol. 35, (1) (Jan. 2017) 13–27.
[61] M. Coram, S. Bohner, The impact of agile methods on software project management, Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 2005, pp. 363–370.
[62] B. Boehm, R. Turner, Management challenges to implementing agile processes in traditional development organizations, Softw. IEEE 22 (5) (2005) 30–39.
[63] I. Rus, M. Lindvall, Knowledge management in software engineering, IEEE Softw. 19 (3) (Jun. 2002) 26–38 Los Alamitos.
[64] T. Cooke-Davies, The 'real' success factors on projects, Int. J. Proj. Manag. 20 (3) (Apr. 2002) 185–190.
[65] K. Marjo, et al., From feature development to customer value creation, Requirements Engineering Conference, 2009. RE'09. 17th IEEE International. IEEE, 2009.
[66] T. Gorschek, A. Davis, Requirements Engineering: In Search of the Dependent Variables, Inf. Softw. Technol. 50 (1-2) (2008) 67–75.
[67] E. Kupiainen, M.V. Mäntylä, J. Itkonen, Using metrics in agile and lean software development – a systematic literature review of industrial studies, Inf. Softw. Technol. 62 (2015) 143–163 Jun.
[68] N.E. Fenton, M. Neil, Software metrics: roadmap, Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, 2000, pp. 357–370.
[69] D. Hartmann, R. Dymond, Appropriate agile measurement: using metrics and diagnostics to deliver business value, Proceedings of the Conference on AGILE 2006, Washington, DC, USA, 2006, pp. 126–134.
[70] K. Conboy, M. Lang, O. McHugh, Motivating agile teams: a case study of teams in Ireland and Sweden, Proceedings of the International Research Workshop on IT Project Management 2010, Jan. 2010.
[71] T. Dingsoyr, T. Dyba, Team effectiveness in software development: human and cooperative aspects in team effectiveness models and priorities for future studies, Proceedings of the 2012 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2012, pp. 27–29.
[72] I. Hagg, G. Hedlund, "Case studies" in accounting research," accounting, Organ. Soc. 4 (1/2) (1979) 135–143.