Waste Self-reporting for Software Development Productivity Improvement

Marc Sallin[1(✉)], Martin Kropp[1], Craig Anslow[2], and Robert Biddle[3]

[1] University of Applied Sciences Northwestern Switzerland, Windisch, Switzerland
marc.sallin@outlook.com, martin.kropp@fhnw.ch
[2] Victoria University of Wellington, Wellington, New Zealand
craig.anslow@vuw.ac.nz
[3] Carleton University, Ottawa, Canada
robert.biddle@carleton.ca

Abstract. Little research has been done on enabling software development teams to self-report waste to assist in productivity improvement. This study created a waste categorization and survey for teams to identify and quantify wasteful activities. Developers from a Swiss company used the survey for three weeks. Participants found the survey helpful for identifying waste but there was little evidence that self-reported waste correlated with improved performance.

Keywords: Empirical · Software development · Productivity · Efficiency · Waste · Lean · Case Study

1 Introduction

Improving the productivity of software development teams is a natural and perpetual goal for organizations. However, measuring the productivity of software development teams has always been seen as inherently difficult. Productivity is determined as the ratio of an achieved output and the required input [1]. In software development, the input mainly consists of the working time spent on development, expressed as the cost of developing software. With the output, it is less clear: the produced lines-of-code or number of implemented functions have shown not to be suited to measure the output. Both quantity and quality of the output are hard to define for software development and typically vary from project to project. Therefore, there is no common approach yet to measure them [1]. As a consequence, organizations often do not measure productivity at all and trust in their intuition for optimization or adopt sub-optimal measurements, which can lead to even worse decisions or wrong incentive behavior [2].

In this study, we investigate a new approach to improve software development productivity. By definition, removing waste from a process improves productivity [3]. Waste is referred to as "*any activity that consumes resources but creates no value for customers*"[4]. However, waste is often not easy to identify, since it can be hidden behind administrative tasks, multitasking, poor prioritization,

C. J. Stettina et al. (Eds.): XP 2023, LNBIP 475, pp. 50–66, 2023.
https://doi.org/10.1007/978-3-031-33976-9_4

and invisible cognitive processes [5]. This is, among other reasons, why waste identification and removal is typically approached through action research or case studies. External personnel can help a team or an organization to identify and remove waste [6–8], but this makes waste identification and quantification expensive. For that reason it is often performed as a one-time intervention, and lacking a sustainable effect on productivity improvement.

In this paper, we present a concept for a systematic yet lightweight self-reporting of waste for software development teams to guide and measure productivity improvement. We first developed a self-reporting tool for identifying and reporting waste and explored in a study whether the amount of reported wasted time is a reliable proxy to track productivity improvements in the long term. More concretely, we wanted to examine the following research questions:

RQ1: Can a software development team identify and quantify waste by using self-reporting? The self-reporting approach should present a good balance between weight and completeness to not miss important waste. The reporting itself should not be seen as waste but should help adding value in a sense to help improve productivity.

RQ2: Is self-reported amount of waste correlated with software delivery performance? To use the amount of reported waste for tracking productivity improvement we expect it to decrease as a team improves its productivity. Based on the PE-Model [1] which describes performance as a broader term including software productivity we use performance as a surrogate for productivity. To collect first evidence for this assumption we look for a relationship between delivery performance [9] and reported waste.

2 Related Work

"Waste" or *"Muda"* (Japanese for waste) has its origin in the concept of lean manufacturing. The core principle of "lean" is to eliminate non-value-adding activities, which are defined as waste. Womack & Jones proposed lean following his analysis of the Toyota Production System (TPS) [4]. TPS prioritizes waste removal by creating a culture that pursues waste identification and elimination in the entire production of a vehicle [3]. The process discerns three types of activities: activities that create value for the customer; activities that create no value for the customer but are currently necessary to manufacture the product; and activities that create no value for the customer, are unnecessary. These are considered waste and therefore should be removed immediately. Initially, the TPS characterized seven types of waste [3], which later were extended with two more waste types by Womack & Jones [4]. In their work about Lean Software Development (LSD), Mary and Tom Poppendieck adapted lean and the TPS from manufacturing to software development, and identified seven waste categories in software development [10].

Identifying waste in software development seems easy only at first glance, by potentially just observing it [5]. Some studies describe how researchers identified

and removed waste in software development teams or organizations [6–8]. All of them included external personnel which helped with the identification. Several studies have been conducted to identify and categorize waste. Sedano et al. defined a taxonomy of waste consisting of nine categories [5]. Al-Baik & Miller conducted action research to identify and eliminate waste in an IT organization and defined ten types of waste [8]. All the studies we found identified waste with a one-time intervention and applied methods with a lot of overhead, or which needed external personnel. Deshmukh and Srivastava reviewed lean methodology in software development [11], while Meyer et al. identified context switching as a universal productivity killer [12,13]. Khodawandi categorized tasks as value add, necessary non-value add, and obvious non-value add, but found that this was not granular enough to identify wasteful activities [14]. Halkos and Bousinakis discovered that stress reduces productivity and personal satisfaction increases it [15].

Studies identified waste using Value Stream Mapping (VSM) [6,16–18], open or semi-structured interviews [8,19–22], analyzed content (like retrospectives) [5], conducted observations [5] or used questionnaires [20,22,23]. Al-Baik & Miller used three questions per waste category to identify waste [8]. Ikonen et al. did the same but used the categories defined by LSD [21]. Besker et al. investigated the time wasted caused by technical debt. They let the participants fill out a survey twice a week during seven weeks [22]. The study of Alahyari et al. used open questions but offered no help for recall [19]. Several researchers used directed questions to identify and recall waste; they derived the questions based on the waste categories. Interestingly, the well-known tool to identify waste in manufacturing, VSM, is criticized for being used in the software development context. The fundamental critique is that manufacturing is fundamentally different from software engineering. When applied in practice, participants state that the results are obvious but it takes a lot of time to apply VSM [8,11,17]. While there have been several approaches to capture waste in software development, all of them suffered from being either one-time interactions or heavy-weight, which reduced their sustainability. Our concept is to implement a light-weight approach for waste self-reporting which could be done on a regular basis to improve software productivity in a more sustainable way.

In recent years, the productivity aspect in software development has gained more attraction with the adoption of DevOps [9]. Besides its organizational influence, DevOps often is seen as promising to improve delivery speed, productivity, and quality [24]. Wiedemann et al. found that only four key metrics (FKM) differentiate between low, medium, high, and elite performers: lead time for change, deployment frequency, time to restore service, and change failure rate [9]. These metrics help organizations and teams to determine whether they are improving the overall IT performance. They are strongly correlated with well-known DevOps practices and hence known as DevOps metrics. However, the productivity metrics are lagging measurements and do not directly suggest any actions. Moreover, less mature teams need guidance to be able to improve on these specific metrics.

3 Research Method

We organized the study into three phases. In the first phase we analyzed and summarized the existing literature, theories, and frameworks about waste classifications. Then, we gathered qualitative data from the research context by performing focus group interviews with the developers from the company participating in the main study. The result was a categorization of waste enriched with context-specific elements and examples. In the second phase we addressed RQ1 "Can a software development team identify and quantify waste by using self-reporting?" and we developed a self-reporting survey and running the survey, using the waste categorizations from the preparation phase. We designed the survey by analyzing recall studies and existing waste identification studies. The survey was planned as a daily self-reporting survey for three weeks, including questions about participant's experience with it. Finally, the third phase comprised the analysis phase, in which we analyzed the gathered data to answer RQ2 "Is self-reported amount of waste correlated with software delivery performance?" At the end of the reporting period, the participants were requested to fill out an additional survey, in which we asked them about their software delivery performance. The software delivery performance was determined using the productivity metrics suggested by Forsgren et al. [9]. Using statistical analysis, we investigated if there was a correlation with the self-reported waste collected during self-reporting survey.

The study was carried out in an IT department of a large company in Switzerland. The group consists of strategic divisions that operate in the core markets of the company and affiliated function units. Function units support the group's management and strategic divisions with cross-cutting-concerns. In 2019 the IT function unit had around 1200 full-time equivalent employees with about 330 software developers. The unit runs only projects for internal customers. The function unit is divided into six departments. One of the departments is the department for software development and consists of 25 teams, of which 15 are software development teams. We included the software development department in the study and looked for participants for the focus groups to collect waste examples and to fill out the reporting. The resulting pool of potential participants consisted of 164 individuals, distributed over 15 teams. For self-reporting, we invited the potential participants to attend the study by e-mail. 26 employees responded to this e-mail and agreed to participate. These people represent 10 different teams from the development department. For the self-reporting (SR), twenty-two participants were included in the data analysis. Among them were two women and sixteen men, with ages between 21 and 58. Their highest degree of education is shown in Table 3, Table 2 shows the experience, and Table 4 the employment.

The focus groups (FG) aimed to get an exhaustive number of waste examples. We decided the size of one focus group to be six people and to host one session per group. Six participants per focus group and two sessions could represent twelve of the fifteen teams. To select the candidates for the focus group, we used quota sampling to ensure half of the candidates attending will also participate in self-

reporting and half will not. Moreover, we ensured that a team is represented by at most one person. Four and five participants did finally attend. Two of them were women, and the rest were men, with an average age of 35 years. Table 1 shows their experience and Table 3 their highest degree of education.

Table 1. Experience of focus group participants.[a]

Years	Working for Swiss Post	Experience Software Engineering Agile methodologies	Experience Experience	DevOps
0–2	5 (56%)	1 (11%)	1 (11%)	5 (56%)
3–5	1 (11%)	2 (22%)	3 (33%)	3 (33%)
6–10	2 (22%)	1 (11%)	5 (56%)	0
11–15	1 (11%)	3 (33%)	0	0
≥16	0	2 (22%)	0	1 (11%)

[a]Due to rounding errors, the percentages may do not sum up to 100%.

Table 2. Experience of self-reporting participants. [a]

Years	Working for Swiss Post	Experience Software Engineering	Experience Agile methodologies	Experience DevOps
0–2	8 (36%)	4 (18%)	5 (23%)	11 (50%)
3–5	6 (27%)	3 (14%)	8 (36%)	8 (36%)
6–10	2 (9%)	4 (18%)	9 (41%)	3 (14%)
11–15	4 (18%)	5 (23%)	0	0
≥16	2 (9%)	6 (27%)	0	0

[a]Due to rounding errors, the percentages may do not sum up to 100%.

Table 3. Highest degree of education of all study participants. [a]

Degree of Education	FG	SR
Vocational Education	0	1 (6%)
Higher Education	3 (33%)	2 (11%)
Bachelor of Science/BSc	4 (44%)	10 (56%)
Master of Science/MSc	1 (11%)	3 (17%)
PhD	1 (11%)	2 (11%)

[a]Due to rounding errors, the percentages may do not sum up to 100%.

Table 4. Employment of self-reporting participants.[a]

Employment	Number
100%	10 (56%)
90%	3 (17%)
80%	4 (22%)
70%	1 (6%)

[a]Due to rounding errors, the percentages may do not sum up to 100%.

4 Classification of Software Development Waste

We first conducted a literature review to get an initial list of empirical-based waste classifications along with examples. We then combined the found classifications to get one classification list that covered all found examples. Finally,

we held two focus group sessions to collect further examples of waste from the company and verified that the classifications also covers those examples. Table 5 lists the twelve categories used to classify the waste examples from the literature review and the examples collected during the focus group sessions.

For the literature review we used the keywords "software development", "software engineering" and "waste". As suggested by Gusenbauer [25], we used three search engines to retrieve literature: ACM Digital Library, ScienceDirect, and Scopus. We found sixteen empirical and peer-reviewed studies which are about waste in software engineering according to lean thinking. We extracted 105 unique examples of waste and found three studies [5,8,19], containing a categorization of waste. Out of the 105 examples, 85 could be assigned to a category of Sedano et al. [5]. The remaining examples could be assigned to the category "Management & organizational aspect" or "processes" from Alahyari et al. [19]. We merged those two categories as the processes are an organizational aspect. We did not use the categorization of Al-Baik and Miller [8] because the study context was a whole IT organization and not just software development.

To gather waste examples from the research context we conducted two focus group sessions [26] with the topic "How do I waste my time?" As preparation, the participants were introduced to the concept of waste by reading an instruction document prepared by the research team. We assigned each example to a waste category identified in the literature review. The examples which could not be assigned to a category were analyzed to define possibly new categories.

The first focus group session generated 77 examples from which 68 did fit into existing categories. The second session generated 90 examples, 63 could be assigned directly. 28 of the not categorized examples were about doing manual work which could have been done automatically. For example, "requesting for firewall rule changes" or "requesting for new database infrastructure". The participants mentioned that those things need to be done by looking information up and sending e-mails within the organization. Hence, we defined an additional category, "manual work", as "The cost of doing work manually which could be automated". The remaining eight examples without a category were related to unreliable infrastructure and the resulting troubleshooting, the lack of responsibility and doing things that are not related to the person's job. While those activities are necessary to reach the overall goal and hence not waste at a first glance, they distract from performing the value-adding activities. For example, troubleshooting infrastructure and platforms should not be a common concern for development teams. Therefore, we introduced the custom category "other duties" as "the cost of doing work which is supposed to be done by others."

5 Waste Self-reporting Survey

The second phase comprised the development of a waste quantifiable survey questionnaire, the execution of the survey over the planned period of time, and a final retrospection survey, in which the participants reported about their experience with the waste self-reporting. To be able to quantify the reported waste,

we defined a measurement for each waste category together with its measurement unit. The identified measurements with their units are shown in Table 5 for each category and are explained in the following.

Table 5. Waste Categories incl. Measurement. WC11 & WC12 are new.

ID	Waste Category	Measurement and Unit
WC1	Building the wrong feature or product [5]	Customer confidence (Likert-Scale)
WC2	Mismanaging the backlog [5]	Time spent (h) & Delay (h)
WC3	Rework [5]	Time spent (h)
WC4	Unnecessarily complex solutions [5]	Time spent (h)
WC5	Extraneous cognitive load [5]	Time spent (h)
WC6	Psychological distress [5]	Stress (numerical rating scale)
WC7	Waiting/multitasking [5]	Delay (h) & Context Switches (count)
WC8	Knowledge loss [5]	Time spent (h)
WC9	Ineffective communication [5]	Time spent (h)
WC10	Management & organizational aspect [19]	Time spent (h) & Delay (h)
WC11	Manual work (new category)	Time spent (h) & Delay (h)
WC12	Other duties (new category)	Time spent (h)

Time Spent: The time spent is how much active time is spent on a wasteful task or activity. One example is the time spent doing rework. The time spent is measured using the unit hour.
Delay Time: Activities in some categories do not require active working time but cause delays. Delays are harmful because they increase the lead time. The delay is measured in hours. Delay typically occurs in combination with other observable measurements. An example is the waste category "Management & Organizational aspects." This category is quantifiable by time spent (e.g., filling out a form for approval) and with delay (e.g., waiting for approval until one can continue with the activity).
Stress Level: Psychological distress is subjective to people who experience it. The Cohen Perceived Stress Scale (PSS) is widely used to measure the perceived stress. But as it is time-consuming to report. Hence, we took inspiration from pain measurement and decided to use a numerical rating scale (numerical rating scale) to measure psychological distress. On the scale zero means not stressed at all and ten extremely stressed.
Customer Confidence: Creating a feature that a customer does not need is a waste of time. However, if it is already evident when the feature is created that it is not needed, why is it built? In hindsight, "Building the wrong feature or product" can be quantified by the time wasted to build something; but while working on it, this may not be that clear. Hence, we decided to not measure it with the time spent, but rather with the degree of confidence a participant has, that they work on the right thing from a customer perspective with a five-point Likert scale.

Context Switches: The category "waiting/multitasking" can be measured by "delay time" but also by counting context switches. Interruptions or waiting which causes context switches come at a high-cost.

For the question creation, we considered several factors about how recall works. Specific questions to capture any kind of waste in a certain category possibly increases the accuracy. However, the participants report daily, and a too detailed and long survey leads to reporting fatigue. To prevent reporting fatigue, the number of questions must be minimized, and answering must be fast. To cover all categories with their measurements sixteen questions would be necessary. We excluded two categories and measurements from the survey, which finally resulted in twelve survey questions related to the categories, plus two additional open questions to get extra information.

We excluded the category "mismanaging the backlog" and a part of "waiting/multitasking" from the survey. The category "mismanaging the backlog" is very broad. It goes from "duplicated work" onto "imbalance between feature work and bug fixing" to "not enough ready stories." Mismanaging the backlog will lead to inefficient and ineffective working manners, unnecessary time spent, and delays. Measuring this in a daily or weekly survey seems not an adequate approach for several reasons: a questionnaire with multiple questions would be necessary to identify a certain degree of mismanagement of the backlog; asking those questions seems more a topic for sprint retrospectives, rather than in short intervals. Waste of the category "waiting/multitasking" is observable at different scales. For example, multitasking can mean switching every few minutes between different tasks or working on another project every few days. This is similar for waiting. One can wait a few minutes for a build to complete or wait weeks to get formal approval to continue a task. Other categories already cover the aspects of the large scale. For the small scale, interruptions and context switches could simply be counted and there is promising research with a focus on automatically detecting and reducing harmful interruptions [13,27]. For those reasons, the category "waiting/multitasking" is not explicitly considered in the self-report survey.

We ended up with the following list of the questions asked in the survey. Beside the twelve waste identification questions, we added two optional questions: one to get qualitative information about stress and one to be able to discover waste which did not fit into our categorization.

1. How stressed did you feel today?
2. How much time did you spend on preventable rework?
3. How much time did you spend on manual or routine work?
4. How much delay (or expected delay) did you experience caused by missing automation/self-service or processes?
5. How much time did you lose because of ineffective communication?
6. How much time did you spend for unnecessary administrative/organizational demands?
7. How much delay (or expected) did you experience caused by administrative/organizational demands?

8. How much time did you spend on activities which are not your duties?
9. How confident are you that you worked on the right things today from a customer perspective?
10. How much time did you spend on unnecessary cognitive load?
11. How much time did you spend on unnecessarily complex solutions?
12. How much time did you spend because of knowledge which wasn't available?
13. Extra: What caused you stress?
14. Extra: Were there wasteful activities which you couldn't report because they did not belong to a question?

6 Data Collection

The data collection was conducted in three steps.

a) An initial survey gathered participants' demographics.
b) Then, we conducted the main survey, which went over three weeks to gather information about waste. In addition, before we started the self-reporting, two participants used the survey for two days and gave informal feedback.
c) We concluded the data collection with a final survey to assess the completeness of the daily reporting (participants stated the days on which they deliberately did not report), to assess the usefulness of the survey tool, and to collect the productivity metrics data (see Sect. 7 details).

The software delivery performance was measured using the productivity metrics as defined by Forsgren et al. [9] and was captured from the participants at the end of self-reporting period. We took the questions and answer options from the state of DevOps report survey conducted by DORA in 2021. The daily reported waste was grouped by measurement and aggregated over the whole reporting period. The aggregation was the daily average, and the measurements were stress, time spent, delay, and customer. To get the productivity metrics score, we assigned the answers for the productivity metrics questions to a proportionally increasing score and summed them per participant. The answer "I don't know/NA" was rated with a score of zero. The worst answer (slowest/least stable) was rated with one and so on. We excluded participants who did specify "I don't know/NA" for each productivity metrics question from the data set.

The participants were asked about their subjective experiences in the final survey. We asked about three aspects of the self-reporting tool. First, if the survey helped them to recall the encountered waste. Second, if it helped identify waste, and third if the reported waste during the three-week period was representative of their usual workdays. Additionally, we wanted to know if the participants would be ready to attend repeated measurement periods.

7 Results

All 26 participants completed the initial survey. 22 completed at least three daily surveys of the main survey. The final survey was completed by 21 participants.

In total, 229 reports of the daily main survey were filled out. 14 out of the 22 participants filled out the survey 10 times or more over the three week period. On average, a participant reported about 4.6h of wasted time per day. Rework is the category that is reported to cause the most waste. Rework is responsible for 20% of the reported waste while the next category is other duties with 15%. For delay, one participant reported about 4 h of delay per day on average. Administrative demand caused around 5× as much delay as missing automation. In 26.5% of the time, the respondents were "completely confident" to work on the right things from a customer's perspective. 44.2% of the respondents were "somewhat confident" and 21.9% "neutral." Only 7.5% of the time, the respondents were "somewhat insecure" or "completely insecure" to work on the right things. Figure 1 shows the distribution. The reported stress score had a mean of 2.3, a median of 2, a standard deviation of 2.3. The minimum was zero, and the maximum was eight. The most significant stresses mentioned in free text were meetings (9×), deadlines (8×), and interruptions caused by calls or parallel tasks (4×). To further validate the categorization we gave the participants the possibility to report examples of waste which they think did not fit into one of the given categories. We found fitting categories for all eighteen examples.

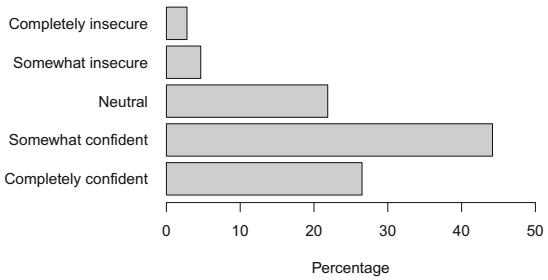

Fig. 1. Confidence to work on the right things.

RQ1: Can a software development team identify and quantify waste by using self-reporting? Figure 2 shows the analysis of the value of and experience with the self-reporting 76% of the participants found the survey helpful for recalling encountered waste (S1). Only 5% disagreed with that statement, and 19% were neutral about it. 38% agreed or strongly agreed that the survey helped to identifying waste, 48% were neutral, and 14% disagreed (S2). For both questions, nobody strongly disagreed. 81% of the participants agreed or strongly agreed that the reported waste during the three-week reporting period was representative of their usual workdays. 43% of the participants were willing to regularly do self-reporting, 33% expressed neutrality and only 24% disagreed with this statement. Overall, these results suggest some reason for optimism about the approach. The causes need exploration: perhaps the daily schedule was too much, or motivation, or lack of confidence were factors.

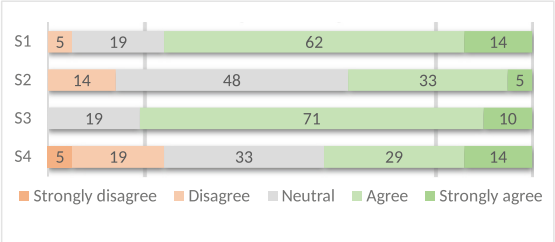

Fig. 2. Value of the waste self-reporting survey. S1: Helpful for recall, S2: Helpful in identifying waste, S3: Representative of usual experience, S4: Willingness to do regularly.

RQ2: Is self-reported amount of waste correlated with software delivery performance? 21 participants filled out the final survey with the productivity metrics questions. We excluded four of them, because they have chosen "I don't know/NA" for every of the four productivity metrics questions. We calculated the mean daily waste reported per measurement, per participant and day. Time spent and delay are both given in mean hours per day. The participants reported using radio-buttons and selected hour ranges. We used the mean of the given range. For the time spent, it was possible to report more than eight hours per day due to the survey design. We corrected for this by adjusting two from a higher value than eight to eight. The Table 6 shows the Spearman correlation coefficient of the measures time spent, delay, stress and customer together with the productivity metrics. The measurements delay, time spent and stress show a low correlation and are not statistically significant. A visual inspection of the scatter plot does also not suggest any trends. We looked at the correlation between the productivity metrics score and the twelve waste categories without aggregating them as well but found no statistically significant correlation. Between the productivity metrics and customer confidence there is a statistically significant moderate correlation. Participants with lower productivity metrics reported a higher confidence to work on the right things from a customer's perspective.

Table 6. The Spearman correlation coefficient and p-value for productivity metrics and waste.

	Time spent	Delay	Stress	Customer
ρ	−0.038	−0.27	−0.3	0.63
p	0.89	0.32	0.24	0.007

8 Discussion

RQ1: Can a software development team identify and quantify waste by using self-reporting?
The self-reporting survey usage by developers indicated that it assisted the recall and quantification of software development waste. We found the survey to be less helpful in identifying new wasteful activities for the participants. We learned that three weeks of self-reporting was enough to get a representative picture of the usually encountered waste. The participants have shown a great willingness that they would participate in self-reporting regularly.

The large majority of the participants found the survey very helpful for recall and remembering encountered waste. This finding indicates that the categories and related questions are good in creating a stimulus for recalling the already known waste but are not as good in helping participants to identify waste they are not aware of. Identification of waste was found to be difficult by other researchers already [5,28]. Another aspect we found was that almost half of the participants did not fill out the daily survey at least once. This shows that a non-negligible share of the participants did not follow the process despite granular information upfront and regular customized reminders. Further data analysis would be needed to evaluate the cause for this.

The shortcomings of identification of new waste may be solved by addressing the described question-substitution effect, by a conceptual change and by providing better tooling. Due to the way cognition works, we consider providing more explanations and examples together with each question as not a promising approach. We suggest to adapt techniques of de-biasing such as asking more detailed/specific questions, changing the wording of the same question, or using nudges. However, this must be balanced with the goal of having a lightweight tool that does not burden the participants. The conceptual change may be to reverse the approach by not asking for wasteful activities per category but by asking for all activities and identifying wasteful sub-activities. The idea is similar to what Khodawandi did [14].

Using dedicated software can improve waste reporting reliability by providing support, guidance, and motivation to users. Possible elements were reliable reminders, gamification and automation. PersonalAnalytics by Meyer et al. can be a good starting point and could integrate the context-switch/interruption category [13]. Existing tools like RescueTime and ManicTime can provide inspiration. A tool could provide a list of activities for users to choose from, like Khodawandi did, and ask specific questions about each activity to identify waste. Each activity can have multiple wasteful aspects, and the questions should be based on waste categories and examples from this study.

RQ2: Is self-reported amount of waste correlated with software delivery performance?
The overall results suggest that teams with high score in the productivity metrics do not report significantly less waste than teams that score low. A possible interpretation might be that a team that improves and reduces waste gets

more sensitive to waste, and hence the reported total numbers do not significantly change. That is because self-reporting waste remains subjective. This would make the absolute and relative amount of waste an unusable indicator for measuring productivity improvement. However, it could be that the productivity metrics score is not a good metric to represent the team performance. An alternative explanation is that a well-performing team, according to productivity metrics does not suffer from less waste, e.g., for organizational reasons. Or that the hypothesis only holds when looking at one team and not for comparing teams. Additionally, respondents who chose "I do not know/NA" for every productivity metrics question were excluded. But if they choose this option for one of the questions this is rated with zero points, which distorts the data.

Previous research somewhat contradicts our findings. Brown et al. found that a high amount of rework significantly differs between low, medium, and high performers [29]. However, they define rework as a combination of unplanned work and rework. Moreover, it is unclear how precise the estimations of their respondents were and if they are comparable with self-reporting waste at regular intervals as done in our study. In the state of DevOps report of 2018, Forsgren et al. found that higher software delivery performances were less prone to burnout [30]. This might be an indication that stress is reduced with a higher productivity metrics. However, it has to be considered that they compared with only three clusters of teams while we used a linear scale. Other researchers have already found that reducing waste can sometimes be straightforward [5] but also challenging when it is outside of the control of a team [31]; this is in line with the possibility that even a mature team with high productivity metrics suffers from organizational waste, like other teams with lower maturity.

8.1 Limitations

Construct Validity: The literature review comes with the limitation that it is unsure if we included all relevant studies. We used the term "waste" to retrieve articles. However, authors may have used another term in their studies to describe the phenomena of waste.

Participants read a waste document before the focus group, including a categorization example by Sedano et al. [5]. This could have caused bias and made it easier for participants to recall waste examples fitting into those categories. The moderator's questions could have influenced the direction of the discussion.

Besides the already mentioned threats to validity for the waste example gathering, the limitations for the categorization are rooted in subjectivity. Due to timing constraints and the kind of work, we did not do researcher triangulation.

Finally, we acknowledge a deeper issue that relates to our measures but also the lean concept of "waste": whether the categories identifies truly reflect waste in the sense of resource allocation without value. Determining this would require careful study of the decisions made by the organization and their outcomes.

Internal Validity: The amount of self-reported waste needs to be interpreted with caution because, as with most survey designs, responses may have been affected by the subjectivity of the respondents. The answer about the time was

given as radio buttons with the span of two hours. This methodological decision led to a loss of accuracy. Due to the pragmatic approach of self-reporting using a survey, it was possible for participants to report more than eight hours of time spent waste per day. They were advised to not report waste twice. Nevertheless, we found participants which did not always follow this rule. We corrected for this in the productivity metrics and waste correlation but not for the total reported waste. Participants missed or skipped some daily reportings. Nevertheless, the weeks with missing daily reports were included in the analysis because the visual assessment did not show that missing data led to a systematic overestimation or underestimation.

Though not relevant for the analysis, the following aspects have to be considered when interpreting the data. First, participants did not report during the same three weeks. Second, they did not all have the same degree of employment. Third, not all of them reported the whole three weeks but dropped out early. The self-reporting behavior and the answers about the self-report experience, especially the willingness to do self-reporting regularly, need to be interpreted with the consideration of selection bias. The invitation to participate in the study contained the information that it would be necessary to fill out a questionnaire every day and week.

External Validity: Despite reaching theoretical saturation for the categorization of waste, it is rooted in qualitative research and can not be generalized for all organizations. Another limitation lies in the low amount of data (sample size is small) and the sampling (reporting weeks not equally distributed over the year). Thus, the results of the applied statistical methods must also be interpreted with caution. Besides the methodological limitations there is a conceptual limitation. We acknowledge our approach will most likely not be able to reveil waste when it is at the core of somebody's job description but a holistic perspective is still necessary [28].

9 Conclusion and Future Work

The developed categorization of waste covered all waste encountered by the self-reporting participants, so appears to be a useful basis for future work in this area. Many study participants indicated readiness to engage in self-reports sessions regularly, though some expressed reservations, and in practice a number of missed reports suggests a need to explore causes – perhaps survey frequency and consequent fatigue. We speculate that the practicality of self-reporting could be significantly improved with dedicated tooling.

We did not find a significant relationship between the amount of waste and the productivity metrics. Hence, with this study we cannot provide evidence that the self-reported amount of waste decreases with improved software delivery performance. Nevertheless, we encourage doing further research and conducting a larger and longitudinal study. Future research should especially consider the limitations our study has illustrated and adjust the methodology accordingly.

More research is necessary to improve our approach, to validate our findings and to be able to draw robust conclusions. We suggest conducting future research regarding the following topics: 1) software development waste categorization should be validated in other organizations. 2) investigate how to reduce the self-reporting burden using dedicated tooling and automatic measurement. 3) replicate this study and the statistical analysis with more candidates, while addressing some of the limitations.

References

1. Wagner, S., Deissenboeck, F.: Defining productivity in software engineering. In: Sadowski, C., Zimmerman, T. (eds.) Rethinking Productivity in Software Engineering, Berkeley, CA, Apress (2019). ch. 4
2. Ko, A.J.: Why we should not measure productivity. In: Sadowski, C., Zimmerman, T. (eds.) Rethinking Productivity in Software Engineering, Berkeley, CA, Apress (2019). ch. 3
3. Coniam, F.: A study of the toyota production system from an industrial engineering viewpoint. Manuf. Eng. **69**(10), 14 (1990)
4. Womack, J.P., Jones, D.T.: Lean thinking-banish waste and create wealth in your corporation. J. Oper. Res. Soc. **48**(11), 1148 (1997)
5. Sedano, T., Ralph, P., Peraire, C.: Software development waste. In: Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017 (2017)
6. Mujtaba, S., Feldt, R., Petersen, K.: 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)
7. Bufon, M.T., Leal, A.G.: Method for identification of waste in the process of software development in agile teams using lean and scrum. In: Uden, L., Ting, I.-H., Corchado, J.M. (eds.) KMO 2019. CCIS, vol. 1027, pp. 466–476. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21451-7_40
8. Al-Baik, O., Miller, J.: Waste identification and elimination in information technology organizations. Empirical Softw. Engg. **19**(6), 12 (2014)
9. Forsgren, N., Smith, D., Humble, J., Frazelle, J.: State of DevOps Report 2019. Technical Report, DORA (2019)
10. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit (The Agile Software Development Series). Addison-Wesley Professional, Boston (2003)
11. Deshmukh, M., Srivastava, P.: Literature review of lean methodology and research issues for identifying and eliminating waste in software development. In: Reddy, A.N.R., Marla, D., Favorskaya, M.N., Satapathy, S.C. (eds.) Intelligent Manufacturing and Energy Sustainability. SIST, vol. 213, pp. 375–388. Springer, Singapore (2021). https://doi.org/10.1007/978-981-33-4443-3_36
12. Meyer, A.N., Fritz, T., Murphy, G.C., Zimmermann, T.: Software developers' perceptions of productivity. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, vol. 16–21, November 2014
13. Meyer, A.N., Fritz, T., Zimmermann, T.: Fitbit for developers: self-monitoring at work. In: Sadowski, C., Zimmerman, T. (eds.) Rethinking Productivity in Software Engineering, Berkeley, CA, Apress (2019). ch. 22

14. Khodawandi, D.: Separating and quantifying value and waste to improve operational performance in software development. In: Proceedings of the 1st International Symposium on Business Modeling and Software Design (2011)
15. Halkos, G., Bousinakis, D.: The effect of stress and satisfaction on productivity. Int. J. Prod. Perf. Manage. **59**(5), 6 (2010)
16. Berrahal, W., Marghoubi, R.: Lean continuous improvement to information technology service management implementation: Projection of ITIL framwork. In: 2016 International Conference on Information Technology for Organizations Development, IT4OD 2016 (2016)
17. Ali, N.B., Petersen, K., Schneider, K.: FLOW-assisted value stream mapping in the early phases of large-scale software development. J. Syst. Softw. **111**, 213–227 (2016)
18. Lehtonen, T., Kilamo, T., Suonsyrja, S., Mikkonen, T.: Continuous, lean, and wasteless: minimizing lead time from development done to production use. In: Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016 (2016)
19. Alahyari, H., Gorschek, T., Svensson, R.B.: An exploratory study of waste in software development organizations using agile or lean approaches: a multiple case study at 14 organizations. Inf. Softw. Technol. **105**, 78–94 (2019)
20. Bjarnason, E., Wnuk, K., Regnell, B.: Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering. Inf. Softw. Technol. **54**(10), 1107–1124 (2012)
21. Ikonen, M., Kettunen, P., Oza, N., Abrahamsson, P.: Exploring the sources of waste in Kanban software development projects. In: Proceedings - 36th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2010 (2010)
22. Besker, T., Martini, A., Bosch, J.: Software developer productivity loss due to technical debt-a replication and extension study examining developers' development work. J. Syst. Softw. **156**, 10 (2019)
23. Tuan, N.N., Thang, H.Q.: Combining maturity with agility - lessons learnt from a case study. In: ACM International Conference Proceeding Series (2013)
24. Lwakatare, L.E., et al.: DevOps in practice: a multiple case study of five companies. Inf. Softw. Technol. **114**, 217–230 (2019)
25. Gusenbauer, M., Haddaway, N.R.: Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. Res. Synth. Methods **11**(2), 3 (2020)
26. Kontio, J., Bragge, J., Lehtola, L.: The focus group method as an empirical tool in software engineering. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, Springer, London, pp. 93–116 (2008). https://doi.org/10.1007/978-1-84800-044-5_4
27. Züger, M., Meyer, A.N., Fritz, T., Shepherd, D.: Reducing interruptions at work with FlowLight. In: Sadowski, C., Zimmerman, T. (eds.) Rethinking Productivity in Software Engineering, Berkeley, CA, Apress (2019). ch. 23
28. Power, K., Conboy, K.: Impediments to flow: rethinking the lean concept of 'Waste' in modern software development. In: Cantone, G., Marchesi, M. (eds.) XP 2014. LNBIP, vol. 179, pp. 203–217. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06862-6_14

29. Brown, A., Forsgren, N., Humble, J., Kersten, N., Gene, K.: State of DevOps Report 2016, Puppet + DORA, Technical Report (2016)
30. Forsgren, N., Kersten, M.: DevOps metrics. Commun. ACM **61**(4), 3 (2018)
31. Rodríguez, P., Partanen, J., Kuvaja, P., Oivo, M.: Combining lean thinking and agile methods for software development a case study of a finnish provider of wireless embedded systems. In: Proceedings of the Annual Hawaii International Conference on System Sciences (2014)