Happy Coders and Reluctant Maintainers: Understanding Task Enjoyment in Software Development

Klara Borowa
klara.borowa@pw.edu.pl
Warsaw University of Technology, Institute of Control and Computation Engineering
Warsaw, Poland

Bartłomiej Rasztabiga
Warsaw University of Technology, Institute of Control and Computation Engineering
Warsaw, Poland

Hubert Soroka
Warsaw University of Technology, Institute of Control and Computation Engineering
Warsaw, Poland

Maciej Tymoftyjewicz
Warsaw University of Technology, Institute of Control and Computation Engineering
Warsaw, Poland

Rodrigo Rebouças de Almeida
rodrigor@dcx.ufpb.br
Federal University of Paraíba, Department of Exact Sciences
Rio Tinto/PB, Brazil

Abstract

Happiness and enjoyment are important factors in software practitioners' daily work, possibly enhancing motivation, productivity, and software quality. However, there exist few studies on practitioners' preferences for software development tasks. The goals of this study were: (1) to find which software development activities were the most enjoyable with a focus on practitioners' roles, (2) to find what factors impact this enjoyment, and (3) to assess whether tasks are avoided due to their dislike. To explore these, we performed a 4-phase mixed methods study using questionnaires, interviews, and focus groups with 173 practitioners of diverse backgrounds. Two ranked lists were obtained: (1) tasks depending on enjoyment level, (2) factors causing task enjoyment and dislike. Additionally, we found that in most cases, the enjoyment level did correlate with the time spent on tasks. The task most likely being avoided due to dislike is maintenance. Based on our findings, we proposed a set of improvements that practitioners may use to improve their work environment. Additionally, researchers may use our findings to prioritize future research on practitioner happiness, enjoyment, motivation, and productivity.

CCS Concepts

• **Software and its engineering** → *Programming teams.*

Keywords

Software Engineering Tasks, Practitioner preferences, Mixed-methods study

ACM Reference Format:
Klara Borowa, Bartłomiej Rasztabiga, Hubert Soroka, Maciej Tymoftyjewicz, and Rodrigo Rebouças de Almeida. 2026. Happy Coders and Reluctant Maintainers: Understanding Task Enjoyment in Software Development. In *Proceedings of Paper submitted for review (ICSE SEIP '26).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

ICSE SEIP '26, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Various human aspects are known to affect software practitioners [27], possibly impacting business performance [32]. One such factor can be happiness and enjoyment. Low happiness in developers can cause serious consequences, such as low motivation, low productivity and low code quality [10]. Task assignment seems to have a notable impact on developer happiness. For example, Masood et al. [18] found that developers are more happy when they perform work that gives them learning opportunities.

However, research on practitioner work enjoyment so far has mainly focused on: (1) coding tasks [16], (2) types of factors that may impact enjoyment without the use of any task taxonomy [26] [19], (3) solely the developer's experience without taking into account other practitioner roles [22].

The goal of this paper was to find what activities are most enjoyable & disliked by software practitioners of all roles, as well as what factors impact this enjoyment. Additionally, we explored whether task enjoyment level may impact task avoidance, i.e. making practitioners likely to not perform the task that they should be doing. These goals are represented by the following research questions:

(1) (RQ1) Which activities related to software development are liked and disliked by practitioners?
(2) (RQ2) What are the factors that influence the positive or negative attitude towards these activities?
(3) (RQ3) How does personal dislike/enjoyment of certain activities correlate with time spent on them?

In our study, we extended a pre-existing task taxonomy to cover most software development activities. Then, we strived to answer the RQs through a 4-phase mixed-methods study with 173 participants, based mainly in Poland and Brazil.

Through this study, we created a ranking of the most enjoyable software development-related tasks and factors that impact this like/dislike. We also found that there is a correlation between enjoyment and time spent on tasks. The most impacted task by this phenomenon was found to be maintenance. Based on our findings, we propose a set of improvements for practitioners to enhance their work by helping them perform key tasks that may have been unconsciously avoided by teams due to task dislike.

Additional material containing questionnaire questions, the interview plan, and focus group slides is available online [6].

2 Related Work

Software developer motivation

The idea that developer motivation is crucial is not new. 30 years ago, McConnell already noted that low motivation can significantly degrade productivity and system quality [20]. Motivation, however, is a unique factor not necessarily caused by project success. Linberg [17] found that even in failed projects, job satisfaction could be maintained if tasks aligned with the developers' intrinsic motivators.

There are numerous studies on factors impacting motivation and the consequences of either low or high developer motivation. For example, Ko et al. [12] showcased that inadequate documentation negatively leads to reduced productivity and the feeling of frustration for developers. Additionally, Kuusinen et al. [13] found that developer motivation can be impacted by the IDE they use. Another notable factor seems to be technical debt, since Besker et al. [4] found that it can negatively affect developers' morale.

Beecham et al., in their systematic review about motivation in software engineering, found that "The most commonly cited motivator is the job itself, yet we found very little work on what it is about that job that Software Engineers find motivating" [3]. This could be viewed as one of the factors impacting the creation of research on Dev-X (Deverloper Experience). This term, first introduced by Fagerholm and Munch [8], was meant to convey a variety of factors that impact how developers think and feel about their work. This research domain has since grown significantly, as showcased by a recent systematic review by Razzaq et al. [27], who found 218 relevant papers on this topic.

Tasks enjoyment, and its impact on productivity

A notable observation was made by Nichols [25] regarding productivity. He noted that it is counterproductive to score a developer's productivity without considering what specific tasks they were performing, since there seems to be a huge variance depending on different tasks. Some pre-existing work suggested that some tasks may be less enjoyable. LaToza et al. [14] emphasized that cognitive challenges, particularly the effort required to manage implicit knowledge, often make code exploration or debugging less appealing. Additionally, Selic [30] noted that developers simply may not like writing documentation because they do not view it as valuable.

Yet, Fagerholm et al. [7] found that in the case of some tasks, developers sometimes express an "Intrinsic motivation to perform" based on rewards and their own personal development goals. This in turn lead to higher productivity. Later, Murphy-hill et al. [24] found that the factor most strongly correlating with productivity was "I am enthusiastic about my job." This clearly showcases that personal enjoyment is a crucial productivity factor. As such, to increase productivity, it seems prudent to explore what software development tasks are enjoyable and provide intrinsic motivation.

The enjoyable software development tasks

Licorish et al. [16] explored support, enhancement and defect tasks through data mining projects. They found that "that teams expressed different attitudes when working on various forms of software tasks, and they were particularly emotional when working to remedy defects." This showcases that a practitioners' emotional state may depend on the task they are currently performing. However, their study was based on a very limited set of coding tasks, which is not the case in our study.

A notable study on task enjoyment was done by Meyer et al. [22], based on a large questionnaire filled by Microsoft developers. In this study the researchers focused on what tasks are part of a "good" and "bad" day. Based on this, an assumption can be made that the "good day" activities are more enjoyable than "bad day" ones. In our study, we improve the following in comparison to the work of Meyer et al.: (1) We expand the taxonomy of tasks created by Meyer et al., (2) We focus strongly on diverse roles than solely "developer", (3) We gather data in a more diverse setting (practitioners from different companies from Poland and Brazil), (4) We explore the impacts of task enjoyment on the time spent performing the task.

Mansood et al. [19] performed a study directly addressing the likes & dislikes of developers regarding tasks through an interview study with 32 practitioners analyzed *via* grounded theory. However, they explored what types of tasks were undesirable instead of using a pre-existing taxonomy of tasks. As such, in comparison to our study, many typical activities may have been omitted. Additionally, our mixed methods study on a larger sample provides more generalizable findings compared to an interview study.

Obi et al. [26] performed a large mixed-methods study with developers from Microsoft. Similar to Meyer et al. [22], the main focus of their study were "good" and "bad" days for developers. However, Obi et al. focused on the factors that make "good" & "bad" days instead of tasks. Additionally, they heavily focused on the task of coding in their analysis (they showcased that bad/good days can be recognizable by telemetry data). In contrast, our study's focus is on both developers and practitioners with different roles, and on the specific software development activities specified by our taxonomy, which is significantly larger.

3 Method

To answer the research questions regarding software practitioners' task preferences as well as obtain a deeper understanding of the origins and implications of these preferences, we performed a 4-phase mixed-methods study. The overall study method is presented in Figure 1. This Section describes each of these phases.

3.1 Phase 1: First questionnaire

The goal of this phase was to obtain: (1) software practitioners' ratings regarding all typical software development tasks, (2) information on how much time they spend on these tasks, and (3) initial data on what impacts task enjoyment.

To do so, we had to prepare a taxonomy of tasks, which was then used to make a questionnaire with both ordinal-scaled questions (on task enjoyment and time spent) and open questions (on task enjoyment and dislike factors).

3.1.1 Task taxonomy. The task taxonomy was meant to cover all typical tasks in software development on a similar level of abstraction.

We first included all activities and tasks described by Meyer et al. [22] in their paper about developers' daily activities. Since this

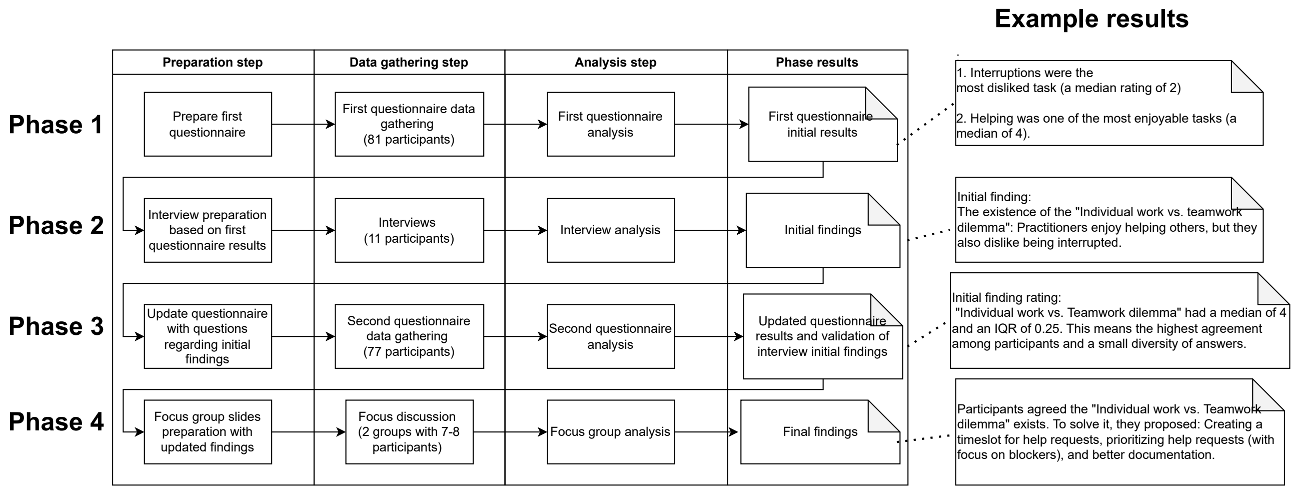

Figure 1: Study Method Overview

paper focuses on all software practitioners (not only developers), we supplemented the taxonomy with tasks on a similar abstraction level, extracted from SWEBOK [31], such as software and architecture design, planning, or infrastructure. Additionally, we included version control mentioned by Masood et al. [19]. Finally, we noticed that all existing lists do not include cases when the workday may not be filled to maximum capacity with meaningful tasks, and practitioners may be waiting for an opportunity to perform a task. We named such waiting activity "Idling" and included it in our taxonomy. The resulting taxonomy of tasks is presented in Table 1.

Resulting task set comprises all the actions a software developer can take in order to create or improve software.

3.1.2 First questionnaire preparation. Having prepared a task taxonomy, we prepared an online questionnaire by following the guidelines by Kasunic [11]. It consisted of the following:

(1) Informing participants about the study details and asking for consent.
(2) Gathering demographic data on the participants.
(3) Ordinal scale questions about the enjoyment of each task (from 1 "Strongly dislike" to 5 "Strongly enjoy").
(4) Ordinal scale questions about the time spent on each task of each task (from 1 "under 1h" to 6 "Over 20 hours").
(5) Two open questions where participants were asked what factors influence their enjoyment and dislike of the listed tasks.

For each question about a specific task, participants could choose the option "N/A" if they never performed a specific activity or did not have any opinion about it to share.

3.1.3 First questionnaire data gathering. The first four authors distributed the first questionnaire using their social media platforms. Despite the questionnaire being in English, since all of these authors are based in Poland, most of the answers came from Polish practitioners. We obtained 82 filled questionnaires from a diverse pool of participants (see Section 3.5).

3.1.4 First questionnaire analysis. **For all ordinal scale questions**, we calculated the median as the measure of central tendency and adopted the interquartile range (IQR) as the measure of statistical dispersion. "N/A" values were skipped when measuring quartiles.

We calculated these metrics for: (1) All participants overall, (2) each professional role separately, and (3) levels of professional experience, (4) developers and non-developers.

Participants could choose their role based on a list of roles or write down their own when necessary. We analyzed only roles that were chosen at least 4 times, and grouped the rest in an "Other" category. As such, we ultimately used the following roles: Developer, Manager, QA, Data Engineer, Admin, Architect, Other.

For open-ended questions, we performed open coding [29] to extract the enjoyment and dislike factors. One author created a list of codes based on the open questions' answers, then a second author reviewed the coding and suggested improvements. During a meeting, they mutually agreed towards one coding scheme and the full coding results. We counted the number of occurrences of these codes in the open questions and ranked them based on frequency.

We summarized the results of this analysis in a single spreadsheet where we ranked all results based on medians (for ordinal scale questions) and frequency (for open-ended questions). This allowed us to find **outliers in the data.**

3.2 Phase 2: Interviews

During this phase, we strived to understand what makes the tasks enjoyable or disliked, as well as to identify what factors may contribute to performing the tasks or avoiding performance. Additionally, based on the interviews, we created a list of 10 initial findings.

3.2.1 Interview preparation. We based all interview questions on the outliers from Phase 1. The questions and reasons behind them are available online [6]. These were, for example:

Table 1: Taxonomy of tasks

Task	Description
Coding	Writing new code or editing existing code to meet the requirements.
Bugfixing	Debugging, and/or fixing errors in the code.
Unit testing	Writing new unit tests to validate behavior of the code.
Manual testing	Interacting with system's user interface to check its reaction – both with and without explicit test cases.
Code review	Analyzing code submitted in code review by another developer. Does not include manual testing.
Version control	Performing actions with a version control system, e.g. cherry-pick, interactive rebase, branch management.
Requirements	Discovering and writing down requirements describing the desired system's behavior.
Documentation	Working on the description of existing system behavior or usage.
Meetings	Planned meetings related to specific technical or business concerns.
Interruptions	Unplanned, informal, typically short meetings or conversations.
Planning	Meetings related to a software development methodology, e.g. sprint planning, sprint review, daily meeting.
Helping	Taking time to help another team member with a specific issue, mentoring.
Learning	Training, courses, looking up information with search engines, etc.
Architecture design	Designing high-level modules of the system and the relationships between.
Software design	Designing the code structure of specific modules, including details of implementation.
Infrastructure	Setting up networks, workstations, virtual machines or containers, arranging CI/CD, deployments, etc.
Maintenance	Working with the system resources without writing its code, e.g., package updates, fixing data in databases, monitoring logs, configuration.
Idling	Activities that are meant to fill time while waiting for a new task.

- What may be the reasons that developers detest interruptions less than non-developers? – Interruptions' preference median for developers was 3, but for non-developers it was 2.
- Factors like monotony, redundancy, and time wasting were often mentioned as reasons for disliking tasks. How do you cope with such challenges? – Those were the most common negative factors, as per open questions.

3.2.2 Interview data-gathering. The first questionnaire contained a question regarding consent for a follow-up interview. We sent interview invitations to all participants who gave their consent in the questionnaire. This resulted in 11 interviewees (see Section 3.5).

All interviews were performed online, led by one of the authors and recorded. Subsequently, the interviewer transcribed the interviews into text form for analysis purposes.

3.2.3 Interview analysis. During the analysis, we searched for factors causing one's preferences towards various tasks, and how these tasks could be performed better or with greater satisfaction. We employed a hybrid of hypothesis and open coding [29], allowing us to prepare codes for closed categories while also permitting the emergence of new codes from the answers to our questions.

Each coded text fragments were labeled with the following codes:

(1) Task - which tasks are impacted by the factor.
(2) Factor category - what the factor is causing.
(3) Factor - short description of the factor (these emerged during open coding).

The "Task" and "Factor category" coding scheme were pre-defined, as tasks were derived from our taxonomy, and we were interested in factors causing: (1) an increase, decrease, or no effect on preference towards a task, (2) made practitioners less or more likely to perform a task, (3) improvement ideas.

The "Factor" codes were created through iterative open coding, since it was not possible to predict all factors beforehand. These codes were created during iterations of coding and review by separate authors, with separate discussions and merging of similar codes during meetings between the coder and reviewer.

Later, the "Factor" codes representing factors for enjoyment & dislike of tasks were merged with the qualitative codes representing these same concepts the first questionnaire (to answer RQ1). The "Factor" codes representing the effect on practitioners work (i.e. performing or avoiding the task) and potential improvements were used separately to answer RQ2 and RQ3.

Additionally, the coders and reviewers created and reviewed memos associated with preliminary findings from the coding process, and codes associated with them. These memos became the source of **the 10 preliminary findings** from this phase.

3.3 Phase 3: Second questionnaire

This phase had three major goals: (1) To verify the preliminary findings from the previous phase, (2) To obtain data from non-Polish practitioners to enhance the validity of the study, and (3) To find whether there are any notable correlations between enjoyment and time spent on the tasks.

3.3.1 Questionnaire update. The existing questionnaire was updated in the following ways:

- For each preliminary finding, an ordinal scale question was added for the new participants to rate their agreement with it, on a scale from "Strongly disagree"(1) to "Strongly agree" (5). With the option to choose "N/A".
- We prepared the option to fill out the questionnaire in Portuguese.

3.3.2 Second questionnaire data-gathering. The questionnaire was distributed through convenience sampling, mainly through social media, by the last author. This way, we obtained an additional 75 filled questionnaires. 70 of them were well filled in Portuguese, and as such, we can assume that the majority of the participants from this phase were based in Brazil (as the last author). The overall information about the questionnaire participants is presented in Section 3.5.

3.3.3 Second questionnaire analysis. Firstly, for all of the questions that matched the old questionnaire, we performed the same data analysis steps and included the new data in the overall results.

Secondly, for all initial finding questions, we calculated the median agreement and IQR to find findings that most practitioners would agree with. We chose a minimum median of 4 (the "Agree" rating) and a minimal IQR of at most 1.25 (meaning that most answers were close to the "Agree" rating) as the cutoff to consider the finding as validated.

Lastly, we calculated the Spearman correlation between levels of enjoyment and time spent on the tasks - both overall and for each task separately. We supplemented this by calculating the p-values: (1) without premutations for all tasks since the sample in that case was large (2599 pairs of values), (2) using the premutation method with 10k premutations for separate tasks since in that case the samples were smaller (between 126-158 pairs, due to some "N/A" answers) [1].

We summarized the data from this phase as a list of findings: (1) the greatest task outliers in regards to enjoyment & dispersion, (2) the verified initial findings, (3) the significant (p-value < 0.05) correlations that were at least weak (Spearman's correlation over 0.2).

3.4 Phase 4: Focus groups

During this phase, we wanted to verify the findings from the second questionnaire and obtain information about possible improvements to practitioners' work that may result from this study.

3.4.1 Focus group preparation. We prepared a set of slides during which we presented the study's previous findings and asked the participants:

- Whether they agreed with the finding.
- Examples from their work that may relate to their finding.
- What problems may result from the finding?
- What improvements can be made in the context of the finding?

3.4.2 Focus group data-gathering. The last author conducted two focus group meetings with two teams from a Brazilian company in the finance field . These were not participants of the questionnaire phase. Details about these participants are detailed in Section 3.5. The focus group meetings were performed online, led by the last author. They were recorded and subsequently transcribed by him as well.

3.4.3 Focus group analysis. Since the focus group goal was to validate the main findings and find their implications, the coding scheme was a hybrid one consisting of the following:

- The open code - showcasing a concept related to the finding. These emerged during the coding.
- Task – the task from the taxonomy the code was related to, or "all" if the code was related to any task.
- Finding – the finding with which the code was connected.

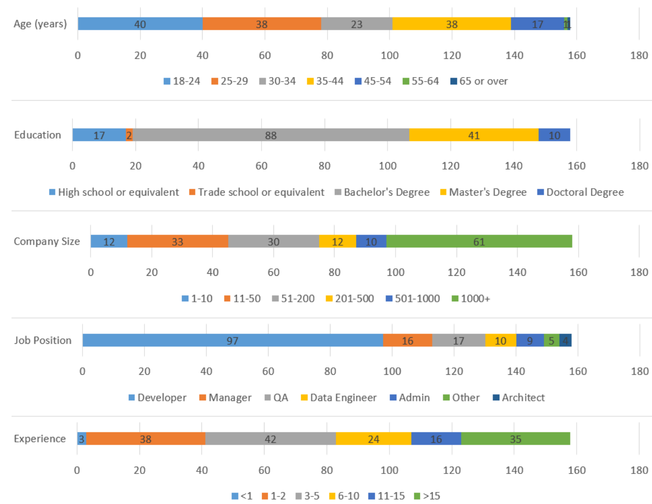

Figure 2: Both questionnaires' participants

- Improvements – a Yes or No label showcasing whether the open code contains any specific actionable improvements related to the finding.

The coding was performed by the first author in two iterations, then reviewed by the last author. Any disagreements were resolved by discussion during a meeting until a consensus was reached.

3.5 Sample

Figure 2 showcases the overall data about all 158 questionnaire participants. Overall, we recruited a diverse set of participants, although some slight bias may be observed towards: (1) most having a Bachelor's degree, (2) around two thirds of the participants' job positions are "Developer", (3) most participants work in large (over 1000 employees) companies. However, since there is no available data about specific target populations that we should strive for, these could also simply reflect the current software industry job market.

The interview and focus group participant details are presented in Table 2. The interview participants were recruited among questionnaire participants. All interview participants were Polish. Focus groups participants did not participate in the questionnaire and were members of two teams from a company in the finance field (51-200 employees) from Brazil.

4 Results

4.1 Most enjoyed and disliked activities

Table 3 showcases the median and IQR of the task ratings from both questionnaires' participants. The number of ratings does not usually match the number of all participants since they could have chosen the "Not applicable" option. Figure 3 showcases the median ratings depending on job position.

Coding and Learning are the two tasks that were universally considered as most enjoyable (a median of 5, i.e. "Strong enjoy"), with little difference between participants (IQR = 1). This seems to be the case for all roles, since the preference median for all of them was either 5 or 4.5 for these tasks.

Table 2: Interviews and focus group discussion participants (EXP – Years of experience in software development, COMP – Company size, ED – Education level)

ID	Age	EXP	Role	COMP	ED
I1	18-24	1-2	Data Scientist	51-200	Bachelor
I2	18-24	3-5	Developer	1000+	Bachelor
I3	18-24	1-2	Developer	11-60	High School
I4	35-44	6-10	DevOps	11-50	Bachelor
I5	30-34	6-10	Developer	1000+	Master
I6	18-24	3-5	Developer	51-200	High School
I7	18-24	1-2	Developer	11-50	High School
I8	18-24	3-5	Developer	51-200	Bachelor
I9	18-24	1-2	Developer	11-50	Bachelor
I10	18-24	< 1	Developer	51-200	Bachelor
I11	30-34	11-15	Developer	11-50	High School
F1.1	30-34	3-5	Developer	51-200	Bachelor
F1.2	18-24	1-2	Developer	51-200	Bachelor
F1.3	35-44	11-15	QA	51-200	Bachelor & PGCert*
F1.4	18-24	3-5	Developer	51-200	Bachelor
F1.5	30-34	6-10	Developer	51-200	Bachelor & PGCert*
F1.6	35-44	3-5	Technical Leader	51-200	Bachelor
F1.7	25-29	6-10	Technical Leader	51-200	Bachelor
F1.8	35-44	18-24	Squad Leader	51-200	Bachelor
F2.1	18-24	1-2	Developer	51-200	Bachelor & PGCert*
F2.2	18-24	1-2	QA	51-200	Bachelor & PGCert*
F2.3	30-34	3-5	Developer	51-200	Bachelor
F2.4	25-29	6-10	QA	51-200	Bachelor & PGCert*
F2.5	30-34	6-10	QA	51-200	Bachelor
F2.6	25-29	11-15	DevOps	51-200	Bachelor
F2.7	45-54	over 15	Product Owner	51-200	Bachelor

* A Postgraduate Certificate (PGCert) is a short postgraduate program, typically lasting 1–2 years, focused on advanced professional specialization.

Interruptions were clearly the most disliked task overall (a median of 2, which is "Dislike"), although there was a big variety in participant responses in that regard (IQR=2). This variety can be role-dependent since managers and "other" disliked Interruptions strongly, and this is neutral for QA and Data Engineers.

Manual testing, Documentation, and Maintenance, despite having a median neutral rating (3), had very diverse ratings (IQR=2). For all of these, role seems to be connected its enjoyment. Manual testing is the most enjoyable for QA, Documentation by Architects and Maintenance by Admins. As such, it seems that individuals enjoying this divisive tasks seem to end up becoming employed in roles that put more focus on these tasks.

4.2 Factors impacting enjoyment

Table 4 presents factors that make practitioners enjoy a task, as well as the number of mentions overall of the factor (in both questionnaires and interviews), and how many times the factor was named as the driving reason for deciding to perform a task (in the interviews). Table 5 showcases the same information for disliked tasks, including factors that impacted dislike and factors that ultimately resulted in avoiding performing a task.

Overall, having an impact on the system's value seems to be the most important factor impacting enjoyment. As participant 6 summarizes "A person feels that they contribute something." This seems to be the main reason behind the enjoyment of coding since code is often associated with value. The second most important factor is the challenge associated with problem solving during the task, e.g. participant 11 stated "The fact that we like puzzles, that it's quite technical, with clear criteria for success." Both coding and learning are impacted positively by this factor. The third most crucial factor is personal growth due to performing the task, as stated by participant 8: "(...) in the sense that career development and improving one's

Table 3: Task ratings

Task	No. ratings	Median	IQR
Coding	152	5	1
Bugfixing	153	4	1
Unit testing	143	3	1
Manual testing	151	3	2
Code Review	148	3	1
Version Control	149	4	1
Requirements	151	4	1
Documentation	157	3	2
Meetings	158	3	1
Interruptions	158	2	2
Planning	156	4	1
Helping	158	4	1
Learning	158	5	1
Architecture Design	146	4	1
Software design	147	4	1
Infrastructure	139	4	1
Maintenance	149	3	2
Idling	144	3	1

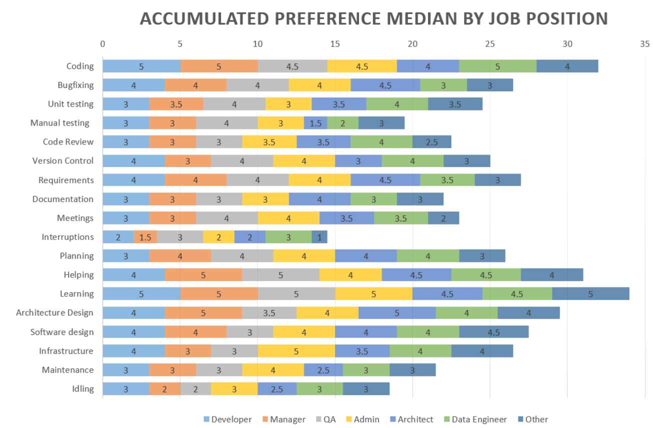

Figure 3: Median enjoyment by job position

Table 4: Positive factors impacting enjoyment and task performance

No.	Factor	Description	Enjoyment Sum (Questionnaires & Interviews)	Performance (Interviews)	
1	Quality/Value impact	The task improves quality or adds value to the system.	68	4	
2	Challenge	The task is a challenge or involves problem solving.	51	1	
3	Growth	The task helps with one's personal development or learning.	46	5	
4	Social aspects	Social aspects improve enjoyment, e.g. good team morale.	42	5	
5	Completion satisfaction	The enjoyment comes from the pleasure of completing a task.	33	1	
6	Preference	The task matches one's personal preferences.	29	3	
7	Productivity	The task allows one to boost their skills and/or productivity.	27	-	
8	Variety	Tasks allows avoiding repetitive work.	26	2	
9	Comfort	The task is enjoyable in a comfortable working environment.	25	1	
10	Importance	Performing the task increases one's social standing.	24	-	
11	Experience	Enjoyment stems from performing a task one has mastered.	21	4	
12	Creativity	The task requires practitioners to be creative.	20	-	
13	Planning	The task has been planned as a part of a big ambitious project.	16	2	
14	Engineering & design	The tasks is directly related to engineering or design.	16	1	
15	Good management	The tasks stems from good "waste reducing" management.	14	1	
16	Autonomy	Practitioners may self-organize when performing the task.	13	-	
17	Interests & hobbies	The task is related to one's personal interests.	9	3	
18	Routine	When enjoyment stems from the appreciation of a good routine.		6	-
19	Client satisfaction	The task makes it possible to directly meet the client's needs.	6	-	
20	Material gains	One obtains tangible gains for performing the task.	6	1	
21	Obligation	The task is performed due to a sense of responsibility.	5	**20**	

Table 5: Negative factors impacting dislike and avoidance

No.	Factor	Description	Dislike Sum (Questionnaires & Interviews)	Avoidance (Interviews)
1	Redundancy & time waste	Task seen as pointless or redundant.	86	-
2	Monotony	The task is considered as boring or tedious.	81	1
3	Difficulty	The task is disliked for being overly hard.	65	2
4	Lack of value	The task or a related artifact is considered as "waste".	60	4
5	Distraction	The task is a as a distraction from the "main work".	49	-
6	Private reluctance	Personal traits cause the dislike, e.g. introversion.	41	5
7	Hostile/Incompetent workplace	The dislike is linked with hostile environment related to it.	36	1
8	Repeatability	The task is disliked due to too many repetitions.	23	-
9	Technical debt	The dislike is caused by technical debt related to the system.	22	1
10	Fear of failure	The fear of consequences for task failure drives dislike.	20	2
11	Poor work delegation	The task is necessary due to bad work delegation.	18	2
12	Too many participants	Too many people are involved in the task.	11	-
13	No personal growth	Task does not promote personal development.	10	-
14	Overwhelming scope	The task is too large to handle comfortably.	10	-
15	Non-technical task	The task is disliked for being "not technical".	10	-
16	Fatigue	Being tired makes the task disliked.	8	1
17	Idleness	Dislike of having no meaningful task to perform.	7	-

skills are very important when performing tasks." Learning is the task directly impacted by this factor.

Redundancy and time wasting are the main factor for task dislike. As Participant 9 stated regarding interruptions: "Perhaps there are more negative emotions in the sense that it delays the implementation of a given topic". Feeling monotony and task difficulty seem to be the second and third factors impacting dislike, respectively. Participant 3 felt that both of them can impact Documentation dislike simultaneously: "Writing documentation is boring and requires you to "put your thoughts on paper". It's not always easy and obvious to write understandable text."

A sense of obligation seems to be the main factor that makes one decide to perform disliked tasks, as Participant 7 stated: "And I don't have any special way of dealing with it. I think I just grit my teeth."

Some participants have private reasons for their task avoidance. Participant 9 gave an example that may cause avoidance of any task, i.e. "And because most programmers are, let's say, more introverted, they need to cool down after such a meeting." Some participants ultimately decide to avoid a disliked task due to perceiving no value from it, e.g., Participant 3 stated in the case of avoiding documentation "If you write documentation that you don't use on a daily basis... it just sits there and your only problem is how to update it, because no one knows what has changed... and no one uses it, because it's definitely out of date."

4.3 Impact of enjoyment level on work

Table 6 shows the Spearman correlation between the reported level of task enjoyment and time spent performing the task. A correlation between 0.2 and 0.399 is considered weak, and over 0.4 is moderate. If a p-value is smaller than 0.05 we can reject the null hypothesis that there is no impact of enjoyment on time spent on task.

For all tasks, there is a significant (p-value almost equal to 0) weak correlation between task enjoyment and time spent. We considered that the practitioners maybe do not avoid disliked tasks, but instead have roles where their disliked tasks are the least necessary, but it does not seem to be the case according to the focus group, i.e. "We've had people avoid challenges, and that's not good because they don't grow. If there's a new activity, we should see it as a learning experience. But if someone runs away saying, "I don't want to do that," they don't evolve."

When analyzing tasks separately, a few do not have any enjoyment & time spent correlation(bugfixing, code review, meetings, interruptions and planning). We asked focus group participants about this, and the underlying reason seems to be that in the case of these tasks, it is hard for a practitioner to control the time spent on them. As one participant stated: "With a bug, for example, we don't know how long it will take before it starts. Once we understand it, we can better estimate it."

Maintenance is the only task with a significant (p-value=0.002) moderate correlation (over 0.4) of enjoyment level on time spent. According to focus group participants, this task can be either enjoyable or extremely disliked depending on the quality of the system, i.e. "When code is poorly structured, maintenance is difficult, time-consuming, and causes headaches. And sometimes we can't improve the code, only patch it, which leads to frustration. But when the code is well-written, maintenance is enjoyable and quick."

Table 6: Correlation between time spent on task and enjoyment

Task	No. of pairs	Spearman correlation	p-value
Coding	148	**0.256**	**0.002**
Bugfixing	148	0.039	0.6369
Unit testing	132	**0.280**	**0.001**
Manual testing	144	**0.236**	**0.0042**
Code review	142	0.151	0.0714
Version control	142	**0.237**	**0.0042**
Requirements	139	**0.354**	**0.0002**
Documentation	146	**0.255**	**0.003**
Meetings	156	-0.040	0.6103
Interruptions	155	-0.040	0.6155
Planning	152	0.140	0.0874
Helping	157	**0.226**	**0.0044**
Learning	156	**0.248**	**0.0026**
Architecture design	139	**0.376**	**0.0002**
Software design	137	**0.224**	**0.0094**
Infrastructure	126	**0.296**	**0.0014**
Maintenance	144	**0.416**	**0.0002**
Idling	136	**0.269**	**0.0010**
All tasks	2599	**0.269**	$\mathbf{3.09 \times 10^{-44}}$

4.4 Findings confirmed by second questionnaire and focus group.

In this subsection, we present initial findings from Phase 2, confirmed by the second questionnaire, and then explored during the focus group. In the second questionnaire, all of these had a median agreement of 4 (i.e. "Agree" in a 1-5 scale from "Strongly Disagree" to "Strongly Agree"), and a small IQR (under or equal to 1.25), showing strong overall agreement.

The individual work vs. teamwork dilemma. A dilemma stems from practitioners' extreme dislike of interruptions and relatively high enjoyment of helping others, since calls for help can be viewed as interruptions. The focus groups suggested that solving this is a matter of sensitivity, i.e. "You have to have sensitivity." and that if a problem is of high priority, they are happy to help: "I ask if they have priority. If the person is blocked, I help. I prefer to unblock a colleague rather than leave them stranded." Another suggestion given by participants was to use a specific timeslot for helping requests, i.e. "When I know I need to interrupt someone, I try to do it early in the day, send a message, and arrange a good time. It's better than interrupting abruptly. But I know sometimes there's no way around it."

Monotony is the main source of documentation and manual testing dislike. The results showed that practitioners simply view some tasks as repetitive and boring, most prominently writing documentation and manual testing. As one focus group participant stated: "It's boring, but it's essential. If the documentation is poor, people outside the team can't find the information and constantly interrupt us. This takes away from development time." The focus group believed that automatizing these tasks as much as possible was the best answer, i.e. "(...) repetitive tasks like manual testing can be automated. Documentation that's frowned upon may be because it's poorly written or verbose." Some believed that the use of LLMs could solve this problem "Creating documentation is tedious, but I had an experience with AI. I refactored a product's documentation so that AI could automatically answer questions. In this process, organizing the documentation was exciting."

The experience vs. exploration dilemma. This finding comes from two seemingly contradictory observations: (1) Practitioners preferred tasks that they were experienced with and could do easily, and (2) Practitioners wanted to learn and try new skills. The focus groups agreed with this as well, i.e. "I think this dilemma really exists. It's enjoyable to do something we've already mastered, but when something new and urgent comes up, it's chaotic. When we have more time to learn, it's less unpleasant. Next time, it'll be easier." To solve this, they proposed blending tasks to give both types of experiences to practitioners: "Here on the team, since we have several different products and technologies, we try to blend them. If someone has been working solely on test automation for several sprints, we give them new activities to vary and learn new things."

Meetings are disliked when too many people take part in them unnecessarily. During the interviews, a point was raised that too many people sometimes participate in meetings. Focus group participants agreed, e.g. "In client meetings, it's even worse. We've had cases with 15 or even 20 people, and only two spoke. Sometimes they spent 80% of their time on off-topic topics. It's disruptive." As a solution, some reported minimizing the number of regular meetings: "We try to avoid standard meetings, like daily meetings." Additionally, to keep the whole team informed despite minimizing the number of meeting participants, the focus group suggests keeping a record of the main decisions from meetings, i.e. "The problem is when there's no record of the meeting, and others don't know what was decided."

Finding enjoyment is possible when including disliked tasks in larger, meaningful tasks. A major factor contributing to task dislike was the belief that the task had no value. During the interviews, this led to the emergence of a solution that can be used for any task – putting these disliked tasks in a larger workflow that clearly leads to the increased value of the software. The focus group sated that this could positively impact both manual testing, e.g. "(...) in the case of IS automation, I've been doing scripting and testing, while participant 6 is building the automation. If he were validating real bugs alongside the automation, it would be more enjoyable. Less interesting activities become more useful when placed in a larger context." Documentation dislike could be managed in such way as well, e.g."Here, we always update documentation for each request."

5 Discussion

In this section we will shortly answer all of the RQ, additional findings and discuss the findings' implications as well as possible improvements.

RQ1: (Dis)liked activities

All information of most enjoyable/disliked tasks is presented in Section 4.1. The most universally enjoyed task by most software practitioners, non-dependent on role, is coding. A key implication of this information is that coding seems to be intrinsically enjoyable and as such, motivating practitioners to code seems to have a rather small priority. This is important since much of previous research focused on increasing developers motivation to code [26] [27] [13], with less focus on other activities. Therefore, we suggest that motivating practitioners to code is, in most cases, not necessary, and motivational efforts should be put elsewhere.

On the other hand, interruptions are the most disliked task. This is in line with previous research [21]. However, in the case of this task, the participants' answers were very diverse. For some roles, i.e., QA and Data Engineers, Interruptions were considered neutral. Other factors may impact these differences as well. As such, it seems that some individuals enjoy interruptions much more than others. Therefore, we suggest that each team should identify the individuals with least interruptions dislike and possibly give them more responsibilities that may be connected with interruptions, e.g., mentoring junior colleagues.

Additionally, our participants themselves propose a solution to the problems caused by interruptions (see Section 4.4), i.e. reserving a timeslot in the workday specifically for interruptions.

RQ2: (Dis)like factors

All information on like/dislike factors is presented in Section 4.2.

The most crucial factor improving enjoyment is the clear impact on the system's value or quality. This seems to match the findings of Meyer et al. [23] who found that "good days" for developers usually have a balance between creating something of value and different tasks. Additionally, it seems that this factor may be an intrinsic factor impacting software engineers, since Li et al. [15] found that "Improving", i.e. always looking to improve the product, is a key attribute of good software engineers. Our study participants clearly agreed (see Section 4.4) that a suitable way to make tasks seem to have more value is to include these tasks in larger workflows that clearly produce value. This way, disliked tasks would not build up, and the practitioners would be able to see the value of their work more clearly.

The factor most notably impacting dislike is the feeling of redundancy and time wasting. This matches the work of Meyer et al. [23] in the regard that the participants of their study often viewed meetings as unproductive, and this feeling caused "bad days". Our study's participants (see Section 4.4) gave us more details about the dislike of meetings. It seems that too many meetings were not only viewed as unnecessary, but also that too many participants took part in them. As such, our focus group suggested that the number of participants in meetings should be minimized to the most crucial group and that notes from meetings were enough to communicate meeting results to others.

The second biggest factor seems to be monotony, which impacts documentation writing and manual testing. While the typical idea of automating these tasks was mentioned by our participants, it seems that they also had high hopes in improving their enjoyment of documentation writing through using LLMs. This means that current efforts in using AI for this task are possibly of big significance [33]. Overall, creating documentation drafts using LLMs seems to be one of its key uses for software development in the near future.

RQ3: Correlation of (dis)like and time spent

Section 4.3 showcases the correlations between task like/dislike and the time spent on them. It is notable that time spent on tasks is not synonymous with productivity [9]. As such, while productivity may rise with time spent, it is not the focus in this case. Instead, we show that some tasks are more or less likely to be avoided by practitioners due to their task dislike/enjoyment.

For all tasks combined, there exists a statistically significant weak correlation between enjoyment and time spent. This means that while impact may not always be big, it is noticeable. For some tasks, this was not the case, and these were tasks on which practitioners could not impact the time spent, i.e., meetings, code review, interruptions, and planning. This means that in the case of these tasks, it is not necessary to motivate practitioners to perform them, since they will most likely do them anyway.

When the practitioner could impact the time spent, they usually spent more time on the tasks they enjoyed. However, maintenance is a notable outlier. The correlation is the strongest of all tasks, i.e. it is moderate. Also, the diversity of enjoyment of this task is high (IQR = 2). As such, it seems that practitioners disliking maintenance are able to avoid these task in many cases. The focus group participants said that this may be caused by technical debt, since it can make maintenance hard to perform. This is in line with existing technical debt research [5], which shows that TD negatively impacts practitioners' productivity. This, in turn, can cause a dangerous vicious cycle since by not repaying the TD (e.g., fixing maintenance problems), practitioners can end in a cycle of TD accumulation [34].

Therefore, we identify maintenance as the key task most impacted by practitioners' enjoyment. As such, it is crucial for all teams to establish whether the maintenance of their systems is "enjoyable enough" and that this task is not avoided. Since there is great variability in personal preference for this task, and practitioners employed in the Administrator position seem to usually enjoy it, it may be prudent to have at least one team member who greatly enjoys this task be explicitly employed in a role responsible for maintenance.

5.1 The experience vs. exploration dilemma

While all other "additional findings" confirmed by the second questionnaire and focus group were already included in the discussion, this one is not clearly related to any of our planned RQ.

This dilemma stems from two enjoyment factors that are contradictory, i.e. learning new things and performing tasks one is knowledgeable in. Our participants proposed blending these two types of tasks to solve this dilemma, yet the specific balance for this is still an unknown and may require further research. Giving tasks that require learning is particularly problematic, Mansood et al. [18] found that in these cases possible conflict arises between developers and management, if business priorities are not given enough consideration. As such, simply "blending tasks" blindly may even be dangerous and risky for the project's success.

As such, we propose that in times when the risk to business is small, more priority should be given to tasks that provide practitioners with personal development. However, this cannot be done without taking into account business needs.

6 Threats To Validity

We discuss threats to validity based on Runeson et al. [28]:

Construct Validity: The design of the questionnaire, interview, and focus group may have influenced the participants' responses. Particularly, during the interview and focus group when we presented the results of the previous design step, it is possible that participants refrained from giving us any disconfirmatory information.

Internal Validity: There is a danger that the self-reported data about time spent on tasks was not accurate or influenced by variability in individual schedules and timekeeping habits. Irregular work patterns can reduce the reliability of these estimates, a common challenge in studies relying on retrospective self-assessments. Additionally, since negative experiences tend to be recalled more vividly than positive ones [2], this could have resulted in an overemphasis on negative aspects during interviews and open-ended questions.

External Validity: In the case of this study it was impossible to calculate the target population due to a lack of knowledge about the typical population of software development practitioners. As such, we strived to obtain a numerous group of diverse individuals – in regards to role, company size, experience and country. However, it is possible that a different group of participants may have resulted in different findings.

Reliability: To ensure the accuracy and reliability of the qualitative data analysis, the process of coding any qualitative data was conducted by two individuals: a primary coder and a reviewer. Any discrepancies between the coder and the reviewer were discussed and resolved through discussion and consensus. While this approach may have made bias smaller, it is still possible that different researchers may have obtained slightly different qualitative codes.

7 Conclusion

In this paper we explored what software development tasks are considered as enjoyable and disliked by software practitioners in various job positions. We achieved this through a 4-phase mixed methods study by employing questionnaires, interviews and focus group discussions. Key findings from this study include:

- A ranking of tasks depending on their enjoyment level. With coding and learning being the most enjoyed while interruptions are the most disliked.
- A list of factors impacting enjoyment & dislike levels. Most notably, the feeling of impacting solution value or quality is the strongest enjoyment factor. Additionally, viewing a task as redundant or a waste of time are the most often occurring dislike factors.
- The existence of a correlation between task enjoyment and time spent doing it. Most notably, a moderate correlation exists for maintenance tasks. This means that practitioners may avoid performing maintenance tasks due to their dislike.
- We found that some tasks are performed by practitioners anyway, despite their dislike, and that there exists an intrinsic motivation that makes practitioners perform their preferred tasks.
- We created a set of practical suggestions for practitioners based on our findings (see Section 5), which are meant to improve their overall work performance.

This has a potential influence on **future research** since it showcases which tasks may be deflected by practitioners due to their task dislike. The most pressing of all being maintenance. Additionally, our suggestions could be validated by future studies. As such, this study shows future directions for research on motivation and productivity in software engineering.

References

[1] 2025. spearmanr. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html Accessed: 2025-09-04.

[2] Roy F Baumeister, Ellen Bratslavsky, Catrin Finkenauer, and Kathleen D Vohs. 2001. Bad is stronger than good. *Review of general psychology* 5, 4 (2001), 323–370.

[3] Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. 2008. Motivation in Software Engineering: A systematic literature review. *Information and software technology* 50, 9-10 (2008), 860–878.

[4] Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch. 2020. The influence of technical debt on software developer morale. *Journal of Systems and Software* 167 (2020), 110586.

[5] Terese Besker, Antonio Martini, and Jan Bosch. 2018. Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work. In *Proceedings of the 2018 International Conference on Technical Debt*. 105–114.

[6] Klara Borowa, Bartłomiej Rasztabiga, Hubert Soroka, Maciej Tymoftyjewicz, and Rodrigo Rebouças de Almeida. 2025. *Additional Material for Happy Coders and Reluctant Maintainers: Understanding Task Enjoyment in Software Development*. doi:10.5281/zenodo.17223156

[7] Fabian Fagerholm, Marko Ikonen, Petri Kettunen, Jürgen Münch, Virpi Roto, and Pekka Abrahamsson. 2015. Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments. *Information and Software Technology* 64 (2015), 132–147.

[8] Fabian Fagerholm and Jurgen Munch. 2012. Developer experience: Concept and definition. In *2012 International Conference on Software and System Process (ICSSP)*. IEEE, Zurich, Switzerland, 73–77. doi:10.1109/ICSSP.2012.6225984

[9] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The space of developer productivity: There's more to it than you think. *ACM Queue* (2021).

[10] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2017. Unhappy Developers: Bad for Themselves, Bad for Process, and Bad for Software Product. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 362–364. doi:10.1109/ICSE-C.2017.104 arXiv:1701.02952 [cs].

[11] Mark Kasunic. 2005. *Designing an Effective Survey*. Scientific Research, Pittsburgh, PA.

[12] Amy J. Ko, Robert DeLine, and Gina Venolia. 2007. Information Needs in Collocated Software Development Teams. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, Minneapolis, MN, USA, 344–353. doi:10.1109/ICSE.2007.45

[13] Kati Kuusinen, Helen Petrie, Fabian Fagerholm, and Tommi Mikkonen. 2016. Flow, intrinsic motivation, and developer experience in software engineering. *Agile processes in software engineering and extreme programming* 104 (2016).

[14] Thomas D LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering*. ACM.

[15] Paul Luo Li, Amy J Ko, and Jiamin Zhu. 2015. What makes a great software engineer?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 700–710.

[16] Sherlock A Licorish and Stephen G MacDonell. 2018. Exploring the links between software development task type, team attitudes and task completion performance: Insights from the Jazz repository. *Information and software technology* 97 (2018), 10–25.

[17] Kurt R Linberg. 1999. Software developer perceptions about software project failure: a case study. *Journal of Systems and Software* (1999).

[18] Zainab Masood, Rashina Hoda, and Kelly Blincoe. 2022. What drives and sustains self-assignment in agile teams. *IEEE Transactions on Software Engineering* 48, 9 (2022), 3626–3639.

[19] Zainab Masood, Rashina Hoda, Kelly Blincoe, and Daniela Damian. 2022. Like, dislike, or just do it? How developers approach software development tasks. *Information and Software Technology* (2022).

[20] Steve McConnell. 1996. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press.

[21] André N Meyer. 2018. Fostering software developers' productivity at work through self-monitoring and goal-setting. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. 480–483.

[22] Andrée N Meyer, Earl T Barr, Christian Bird, and Thomas Zimmermann. 2021. Today Was a Good Day: The Daily Life of Software Developers. *IEEE Transactions on Software Engineering* (2021).

[23] Andre N. Meyer, Earl T. Barr, Christian Bird, and Thomas Zimmermann. 2021. Today Was a Good Day: The Daily Life of Software Developers. *IEEE Transactions on Software Engineering* 47, 5 (2021), 863–880. doi:10.1109/TSE.2019.2904957

[24] Emerson Murphy-Hill, Ciera Jaspan, Caitlin Sadowski, David Shepherd, Michael Phillips, Collin Winter, Andrea Knight, Edward Smith, and Matthew Jorde. 2021. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* 47, 3 (March 2021), 582–594. doi:10.1109/TSE.2019.2900308

[25] William R Nichols. 2019. The end to the myth of individual programmer productivity. *IEEE Software* 36, 5 (2019), 71–75.

[26] Ike Obi, Jenna Butler, Sankeerti Haniyur, Brian Hassan, Margaret-Anne Storey, and Brendan Murphy. 2024. Identifying Factors Contributing to Bad Days for Software Developers: A Mixed Methods Study. In *International Conference on Software Engineering*. http://arxiv.org/abs/2410.18379 arXiv:2410.18379 [cs].

[27] Abdul Razzaq, Jim Buckley, Qin Lai, Tingting Yu, and Goetz Botterweck. 2025. A Systematic Literature Review on the Influence of Enhanced Developer Experience on Developers' Productivity: Factors, Practices, and Recommendations. *Comput. Surveys* 57, 1 (Jan. 2025), 1–46. doi:10.1145/3687299

[28] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

[29] Johnny Saldaña. 2021. The coding manual for qualitative researchers. *The coding manual for qualitative researchers* (2021), 1–440.

[30] Bran Selic. 2009. Agile documentation, anyone? *IEEE software* 26, 6 (2009), 11–12.

[31] IEEE Computer Society. 2024. *Guide to the Software Engineering Body of Knowledge (SWEBOK)* (version 4.0 ed.). IEEE Computer Society.

[32] Shivam Srivastava, Kartik Trehan, Dilip Wagle, and Jane Wang. 2020. Developer Velocity: How software excellence fuels business performance. *McKinsey & Company* (2020).

[33] Yiming Su, Chengcheng Wan, Utsav Sethi, Shan Lu, Madan Musuvathi, and Suman Nath. 2023. Hotgpt: How to make software documentation more useful with a large language model?. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*. 87–93.

[34] Marion Wiese and Klara Borowa. 2023. IT managers' perspective on Technical Debt Management. *Journal of Systems and Software* 202 (2023), 111700.