Like, Dislike, or Just Do It? How Developers Approach Software Development Tasks

Zainab Masood[a], Rashina Hoda[b], Kelly Blincoe[a], Daniela Damian[c]

[a]*The University of Auckland, Auckland, New Zealand*
[b]*Monash University, Melbourne, Victoria, Australia*
[c]*University of Victoria, Victoria, Canada*

Abstract

Context: Software developers work on various tasks and activities that contribute towards creating and maintaining software applications, frameworks, or other software components. These include technical (e.g., writing code and fixing bugs) and non-technical activities (e.g., communicating within or outside teams to understand, clarify, and resolve issues) as part of their day-to-day responsibilities. Interestingly, there is an aspect of desirability associated with these tasks and activities. *Objective:* However, not all of these tasks are desirable to developers, and yet they still need to be done. This study explores desirability and undesirability of developers for software development tasks. *Method:* Based on semi-structured interviews from 32 software developers and applying a grounded theory research approach, the study investigates what tasks are desirable and undesirable for developers, what makes tasks desirable and undesirable for them, what are the perceived consequences of working on these tasks, and how do they deal with such tasks. *Results:* We identified a set of underlying factors that make tasks (un)desirable for developers, categorised as personal, social, organisational, technical, and operational factors. We also found that working on desirable tasks has positive consequences while working on undesirable tasks has negative consequences. We reported different standard, assisted, and mitigation strategies that aid software practitioners manage developers' likes and dislikes. *Conclusion:* Understanding these likes and dislikes, contributing factors, and strategies can help the managers and teams ensure balanced work distribution, developers' happiness, and productivity, ultimately increasing the value developers add to software products.

Keywords: software tasks, desirability, undesirability, contributing factors

1. Introduction

Software development involves working on technical tasks and collaborative activities [1, 2]. Tasks are high-level work assignments, e.g., new features and bug fixes. Activities are interactions to accomplish any task, e.g., discussions, meeting, debugging, or testing [2]. We refer to such technical tasks and non-technical/collaborative activities as 'tasks' in this paper. These tasks vary in terms of effort involved, thought process, intellectual/cognitive load, and stakeholder communication. Some tasks may require cooperation and collaboration among members, while others may require more intellectual and cognitive work. For example, coding a new feature or enhancing an already working feature would have different requirements compared to documentation, design, or support tasks [3]. Similarly, fixing a bug or an issue requires familiarity and specific knowledge of the developed feature.

Software development tasks and activities are both technical and non-technical/collaborative. Technical tasks require technical skills to accomplish, e.g., core development, testing, and automation skills. Some examples of technical tasks are implementing a new feature, maintaining features, fixing bugs, migrating data, configuring environments, reducing technical debt, and providing technical support such as reviewing code for other teams. These technical tasks are done by dedicated roles such as developers, testers, or architects. Software development also involves non-technical tasks and activities. They may not require technical skills, but for some non-technical tasks, these skills are good to have. Examples of non-technical tasks are creating user guides, conducting feasibility studies, preparing demos, and coordinating with other teams. These tasks and activities could be performed by dedicated roles such as scrum masters, business analysts, technical writers. Roles like programmers and testers are also seen to share these responsibilities. Both technical and non-technical tasks and activities play a significant part in software development. For example, non-technical activities such as formal and informal team discussions can provide input on technical tasks such as implementation details or architectural solutions.

With a growing interest of the software engineering community in investigating the human aspects, numerous researchers have explored developers' feelings and behaviors such as happiness [4], frustration [5], and emotions [6, 7]. This study explores another similar human aspect, i.e., task desirability and undesirability as perceived by developers (referred to as (un)desirability throughout the paper). The term 'developers' represents testers, programmers, architects, and all other roles involved in software development. Desirability is the quality of being worthy of desiring, liking, wanting. Task desirability has to do with the pleasure of a task or an activity, i.e., working on a desirable task will make the developer happy [8]. When a task is desirable, given a choice, developers would like to work on such a task or activity. Similarly, wherever possible, developers

Type	Sub-Category	Examples	Ref
Tasks	*Requirement-related*: tasks that focus on initial stages of software development, i.e., requirement identification, analysis, representation	Identifying constraints, assessing potential problems, requirements classification	[4]
	General software: tasks that focus on later stages of software development, i.e., user support, testing, code reusability	Code restructuring, dead code removal, code inspections, personal debugging, user documentation, on-line help, tutorial production, user training	[5]
	Information-seeking: tasks that involve seeking information	Browsing web, documentation, articles or FAQs, asking coworkers	[14]
	Clerical: tasks that can be completed using a routine procedure	Generating reports/documents , storing design versions, maintaining changes	[13]
	Intellectual: tasks that require non-routine thought processes	Requirement elicitation, requirement classification, estimate tasks/projects	[13]
	Software: tasks related to bug fixing, documentation, or providing new functionality or extending any previous feature	Defects, support tasks, enhancements	[3]
Activities	*Development/coding*: activities related to code-writing tasks	Coding, reading/reviewing code, editing code, navigating code, bug-fixing, testing, committing code, submitting pull requests.	[1], [6], [7]
	Version control: activities related to change management	Reading changes, accepting changes, submitting changes	[6]
	Documentation: activities that involves reading or writing documents	Reading artifacts, editing artifacts, writing artifacts	[6]
	Organizational: activities that involve managing project community, assigning/ un-assigning tasks to developers	Assigning GitHub issue or reviewing pull request	[7]
	Supportive: non-coding activities related to documentation, versioning control, code branch management	Writing documentation/wiki page, managing development branches & releasing or archiving code versions	[7]
	Communicative: activities that involve visible communication	Providing comments on issues, commit, and project milestones	[7]
	Collaboration-heavy: activities that involve working with people	Meetings, emails, networking, helping or mentoring others	[1]
	Other: activities not directly related to development tasks or working with people	Learning and administrative tasks, planning, infrastructure setup	[1]

Table 1: Examples of software tasks and activities from literature

will try to avoid an undesirable task or activity.

2. Background

Literature on software development tasks & activities indicate that software developers work on various tasks as part of their day-to-day responsibilities. Different researchers have used different definitions, terms, and *classifications* for the tasks involved in software development. Table 1 lists examples of software development tasks from literature [1, 9, 10, 11, 12]. Some previous research has classified the types of tasks performed during software development based on the software development life cycle. Brackett et al. classified tasks involved in initial stages of the software development life cycle [9], while Jones reported tasks involved in the later stages of software development [10]. These include *requirement-related* tasks such as tasks involved in requirement identification, requirement analysis and requirement representation, *code-related* tasks such as restructuring of existing code, removal of dead code, code inspections, personal debugging, *testing-related* tasks, e.g., test case development, test library control, and *user-related* tasks, e.g., user documentation, online help, tutorial production, user training.

Some researchers classified the types of tasks based on the nature of software tasks. Robert et al. referred to software tasks as *clerical* tasks, routine tasks that can be completed using a standard way, and *intellectual* tasks that cannot be done following a standard procedure, they require non-routine thought processes to accomplish such tasks [13]. Milewski defined *information-seeking* tasks as the tasks that involve browsing the web (documentation, articles or FAQs), asking friends and/or coworkers, etc. [14]. Licorish et al. classified software tasks as defects, support tasks, and enhancements for providing new functionality or extending any previous feature or fixing issues [3].

Researchers have also classified software development activities. Meyer et al. record activities developers pursue during their workdays. These include development activities related to coding (reading, editing, navigating code), version control (e.g. reading, accepting, submitting changes), documentation (read, edit, write artifacts) [11]. In another work, they classified these activities into development-heavy (coding, bug-fixing, testing) and collaboration-heavy (meetings, helping, networking) activities, and other activities (learning, administrative) [1]. Another study classified software developers' daily activities into four categories [12]. The *organizational* activities involve managing the project community and delegating tasks, e.g., assigning an issue or reviewing a pull request. The *communicative* activities involve communication, e.g., commenting on issues, commit, project milestones. *Supportive* activities are non-coding activities related to documentation, versioning control, code branch management, e.g., writing documentation/wiki page, managing development branches. *Typical* activities are conventional code-writing tasks submitted as code reviews, commits, and pull requests.

Our Grounded Theory study while investigating the phenomenon of *self-assignment in agile software development teams* indicated that there is an aspect of fun and personal enjoyment associated with different software development tasks [8, 15]. Not all tasks are equally fun or exciting to work. Some software developers like to work on front-end related tasks, while others like back-end development. Some developers prefer specific tasks over others for a variety of reasons, e.g., due to the opportunity they provide to grow or because they have the potential of outside endorsement. However, not all tasks are desirable, yet they need to be done. We know from previous studies that working on (un)desirable tasks influences developers' motivation and happiness [4]. Individuals working on tasks they don't enjoy can build up a certain feeling of resentment leading to unhealthy consequences in the long-term. If developers are given a healthier environment and opportunities to work on tasks of their choice most if not all of the time, they can be more productive.

Current literature informs us that developers work on differ-

P#	Age	Sex	Role	Exp
P1	31-35	M	Tech Lead	11
P2	21-25	M	Software Engineer	2.5
P3	26-30	M	Assoc. Tech Lead	5
P4	21-25	M	Software Engineer	2.5
P5	36-40	M	Developer	7
P6	26-30	W	Sr. Software Engineer	4
P7	31-35	M	Tech Lead	7.5
P8	21-25	M	Developer	3.5
P9	26-30	M	Tech Lead	5
P10	26-30	M	Assoc. Tech Lead	4
P11	21-25	M	Sr. Software Engineer	3.5
P12	26-30	M	Assoc. Tech Lead	4.5
P13	31-35	M	Lead Developer	13
P14	36-40	M	Lead Developer	17
P15	21-25	W	Developer	2
P16	21-25	M	Developer	2.5
P17	41-45	M	Lead Developer	20
P18	36-40	W	Scrum Master	9
P19	31-35	M	Lead Developer	8
P20	46-50	M	Lead Developer	25
P21	36-40	W	Tester	5
P22	36-40	M	Configuration Engineer	15
P23	31-35	W	Tester	10
P24	36-40	M	Quality Engineer	15
P25	46-50	W	Senior Tester	20
P26	30-35	M	Developer	5
P27	30-35	M	Developer	15
P28	30-35	M	Tester	10
P29	35-40	M	Development Manager	20
P30	41-45	M	Scrum Master	20
P31	31-35	M	Product Owner	12
P32	36-40	M	Developer	12

Table 2: Demographics of participants

ent types of tasks [1, 9, 10, 11, 12]. A limited number of studies explored developers' (un)happiness and how developers perceive their tasks [4, 8]. These studies indicate that not all tasks are desirable to developers, but they do not elaborate on what tasks are (un)desirable and what makes them (un)desirable for developers and most importantly how developers and teams approach these (un)desirable tasks. To fill this gap, we explored this aspect further in this study.

3. Research Objectives

The overarching research question ***How do developers perceive and deal with (un)desirable tasks?*** drives this study. To answer this, we conducted 32 in-person, semi-structured interviews and focused on the following four sub-questions.

(RQ1) *What types of tasks are (un)desirable for developers?*
(RQ2) *What makes a task (un)desirable for developers?*
(RQ3) *What are the perceived consequences of working on (un)desirable tasks?*
(RQ4) *How do developers deal with undesirable tasks?*

It is important to understand the *what, why, and how* aspects of 'Tasks Desirability' to answer the overarching research question. RQ1 aims to confirm if all software development tasks are desirable to developers. It builds the foundation for RQ2 that examines what makes a task (un)desirable for developers. RQ3 determines if working on (un)desirable tasks leads to different impacts, and finally, RQ4 looks at how developers deal with undesirable tasks.

Our findings indicate that developers have different likes and dislikes towards certain tasks. Not all tasks are (un)desirable to developers. The paper contributes by reporting a set of underlying factors that make tasks (un)desirable for developers. As there will always be tasks that are (un)desirable to developers, the management and teams need to be aware of these factors to provide an environment where team members can freely express their likes and dislikes and efficiently address developers' concerns. Not surprisingly, we also found that working on desirable tasks has positive consequences while working on undesirable tasks has negative consequences. Managers and teams need to work on strategies to deal with undesirable tasks. We report a list of strategies to deal with undesirable tasks. Managers and teams can benefit from these strategies to manage developers' likes and dislikes.

4. Research Method

'Task desirability' was identified as one of the factors developers consider while self-assigning tasks in our previous work on self-assignment [8]. We used Grounded Theory to explore various aspects of self-assignment as part of a broader study. This paper explores this factor further and reports key findings related to tasks (un)desirability as perceived by the developers. We applied data analysis techniques from the Grounded Theory methodology to the data collected through interviews related to task (un)desirability to the data collected through interviews related to task (un)desirability.

4.1. Data Collection

Out of the entire data set from the broader Grounded Theory study with 54 participants from 26 software companies, this study data set includes semi-structured interviews with 32 software practitioners from New Zealand, India, and Pakistan. Due to the iterative and emerging nature of Grounded Theory studies, 'desirability' emerged after the first set of interviews, so questions were added to the interview guide to elicit this information and only the last 32 participants shared information regarding (un)desirability.

For participants recruitment, we contacted developers through personal contacts and networking sites (e.g., MeetUps and Linkedin). All the participants worked for small to large software development companies with 2-25 years of software development experience. They worked under various roles such as developer, software engineer, tester, team lead which brought diversity in our data set. Table 2 presents the participants demographics. The first column states participant numbers P1-P32 to

Quote (P#)	Concept	Sub-Category	Property
Problem is not that you don't want to do that work, the problem is having a conflict with somebody (P20)	Having conflict with others	Social	Conflicts: Undesirability
As this (automation) skill is in market demand and gives you an experience of new learning...(P28)	Learning automation, (skill in demand)	Personal	Learning Opportunity: Desirability
No one values the time spent on documentation reviews as the focus is always on a running software...(P21)	Reviewing documents (Undervalued)	Organisational	Contribution to organisation: Undesirability

Table 3: Open-Coding Examples: Emergence of sub-categories (social, personal, organisational) from participants' quotes leading to the category (contributing factors covered in (Section 5.2)

maintain participant anonymity following human ethics guidelines. The second and third column lists their ages and gender while the remaining columns list their roles in the current company and the total professional experience in the software development at the time of interview.

Before conducting the interviews, we gathered the participants demographics using a pre-interview questionnaire (a google form). The form collected participant information such as professional background, e.g., role, company, personal, e.g., age, gender, qualification, etc. Gathering this information helped us assess the participants' suitability in advance, e.g., their work experience was not less than two years, they were self-assigning tasks to themselves (mostly if not all the time). It also saved time to focus on key areas of the research during the interviews. The questions used in the interviews evolved iteratively throughout the study. New questions were added to focus on new findings (concepts, categories) that emerged from the previous iterations of data collections following Grounded Theory procedures. For example, it was found from the initial round of interviews that task desirability is one of the factors developers consider while self-assigning tasks. At that stage, open-ended questions focused on factors developers consider while self-assigning tasks to themselves [8, 15]. In the future interviews, we asked open-ended questions focusing on task desirability. A few examples are: *What types of tasks are (un)desirable to you? Why are they (un)desirable to you?* These interviews were conducted at the participants' workplaces (e.g., in New Zealand) or venues suiting the participants' convenience (e.g., coffee shops, public places) or through Skype video calls (e.g., for participants in the UK). These interviews were audio recorded and transcribed verbatim for analysis.

4.2. Data Analysis

Data analysis comprised open coding and constant comparison following Strauss and Corbin guidelines [16]. It involved identifying concepts by asking questions and comparing one concept to another. *Concepts* are the words that represent an idea in the data based on researcher's interpretations. Through constant comparison, similar concepts are grouped together to generate a higher level of abstraction, i.e., *Categories*. This classification from concepts to categorisation helps the researcher to manage enormous amounts of data.

We highlighted the potential areas of interest by annotating relevant details in the transcripts to facilitate open coding. We analysed the annotated areas focusing on tasks (un)desirability. We identified concepts in the data from quotes of participants such as from P20, P21, P28 as shown through examples in Table 3. These quotes result in concepts such as 'having conflict with others', 'learning skills in demand or valued by company', 'reviewing documents treated as undervalued activity in company'. These concepts were defined in terms of their properties (characteristics that explain a concept) for further understanding and refinement. For example, 'conflict' was defined with regards to disagreement within (with team members) or outside (external stakeholders) teams. Then, we grouped similar concepts under different categories, e.g., contributing factors to desirability and undesirability were grouped into subcategories, e.g., personal, social, organisational, technical, and operational factors factors for understanding and better representation. Other examples of categories are perceived impact and strategies presented in Section 5.3 and 5.4.

The first author collected the data and performed open coding and the other authors reviewed the final set of concepts, subcategories, and categories. This was done through frequent and detailed discussions amongst the co-authors throughout the data analysis and reporting of results. We proceeded with data collection and analysis until theoretical saturation was reached for the wider study [8, 15]. This study only employed Grounded Theory procedures of open-coding and constant comparison. Therefore, we do not claim theoretical saturation (with 32 interviews) for this part of the study.

5. Results

Software developers work on different tasks and not all these tasks are (un)desirable to developers. We organize the results of our analysis in Figure 1. Table 4 lists examples of tasks shared by the participants. Knowing about these different tasks will help build the context for the study where participants referred to these tasks as desirable or undesirable tasks. We outline the factors that make tasks desirable or undesirable for the developers, the positive and negative impacts of working on (un)desirable tasks, and the strategies developers use to deal with these impacts.

5.1. What tasks are (un)desirable to software developers (RQ1)?

We asked the participants what tasks are desirable and undesirable to them. Their responses confirmed that some tasks are more desirable than others.

Tasks	Examples
Coding	Front-end development (UI, UX), back-end development (application logic, data & application integration)
Bug reproducing & fixing	Debugging or fixing bugs or clients/users issues
Testing	Exploratory testing, API testing, testing, performance & smoke testing
Specification	Requirements gathering, elicitation
Automation	Writing scripts, framework designing, automation tools
Monitoring	Regression build farms, test suites
Reviewing	Code reviews (within or outside teams)
Environmental & Configuration	Infrastructure setup, Configure containers
Documentation	Manuals, technical documentation
Meetings	Scrum events, customer meetings, technical meetings
Emails	Reading/writing emails
Supporting teams	Helping other team members, demos, presentations
Mentoring	Trainings, knowledge transfer

Table 4: Examples of tasks shared by the study participants

5.1.1. Desirable Tasks

We found that most software developers desire to work on tasks that involve emerging research areas, cutting edge technologies, and novel development tools such as using different APIs, programming languages, tools, frameworks, and libraries. Tasks that are complex and provide developers a solid grasp on the system's underlying architecture, logic, complexities, or domain knowledge are desirable for developers. For example, a junior developer stated that broad-level testing is desirable to them as they get to explore the system to ensure their implementation was not breaking the system. This helped them get familiar with the system. For some developers, understanding the intricacies underlying the complex system fascinated them. Some participants stated tasks which take them out of their comfort zone are desirable to them. Conversely, a very few stated tasks that provide opportunities to serve people are desirable irrespective of the nature of the task.

For some developers front-end development tasks are more desirable as indicated by a developer.
"For me, I really like doing front-end stuff" [P14]

For other developers, the back-end tasks are more desirable as quoted by P1.
"They [developers] want heavy, meaty back-end coding work".

For some developers working on enhancements is desirable.
"The enhancements, I guess are a lot more fun".

Generally, many participants reported that working on new features is desirable as brought up by P11.
"Always the new development..."
"Working on new applications, I'd say is probably the most desirable." [P19].

Developers acknowledged reviewing others' code as a desirable task as part of pair-programming or code reviews. However, this relied on the code quality and the peer author. Junior developers said examining experienced programmers' code was a desirable activity and a learning opportunity for them. To summarise, task desirability seems to vary between front-end and back-end development. Developers expressed their desirability for working on new features and enhancements over bug fixes and change requests.

5.1.2. Undesirable Tasks

The majority of the participants reported that tasks that do not require thinking and analytical skills or are redundant or mundane are generally undesirable. Some examples mentioned were testing the UI fields, monitoring daily regression results, and troubleshooting daily build failures. A participant said,
"I don't like working with UI ... doing one pixel up, other pixel down" [P26].

Another common undesirable task reported involves working on live or production servers, e.g., providing live support to customers, changing and uploading data.
"Our team does live support for one of our legacy products. This work is mostly undesirable for individuals" [P15]

Not surprisingly, developers expressed dislike for reviewing and refactoring a bad quality code. Similarly, most of the developers agreed that documentation is one of the most undesirable tasks. Documentation here refers to different activities such as writing test cases, providing code comments, documenting testing results, adding details to discussion threads on any work items, documenting technical details for writers to include in product documentation. Similarly, many developers find fixing bugs or automation failures undesirable.
"Fixing bugs is an undesirable task, let's be clear .." [P20]
Participants responded to unit testing as an undesirable task. Most of them appreciated having dedicated testers for detailed features verification. Also, testing of the historic production issues is regarded as one of the most undesirable tasks.
"Developers often complain about unit testing" [P28]

For some participants, coordinating with different teams is undesirable. Tasks that involve coordinating, or supporting other teams is reported as undesirable, e.g., integrating modules developed by other teams who are unwilling (to provide supporting details). Similarly, tasks like status meetings, planning meetings, and generally meetings are reported as undesirable to developers but unavoidable. Another task, which is not actually related to software development but reported as **undesirable** by a couple of participants, is the involvement in the *hiring of resources* at a technical stage where they need to shortlist and interview the candidates.

5.2. What makes a task (un)desirable to software developers (RQ2)?

Our findings reveal that tasks become (un)desirable due to multiple contributing factors. The categories that emerged in our analysis are personal, social, organisational, technical, and operational. These are explained with examples below.

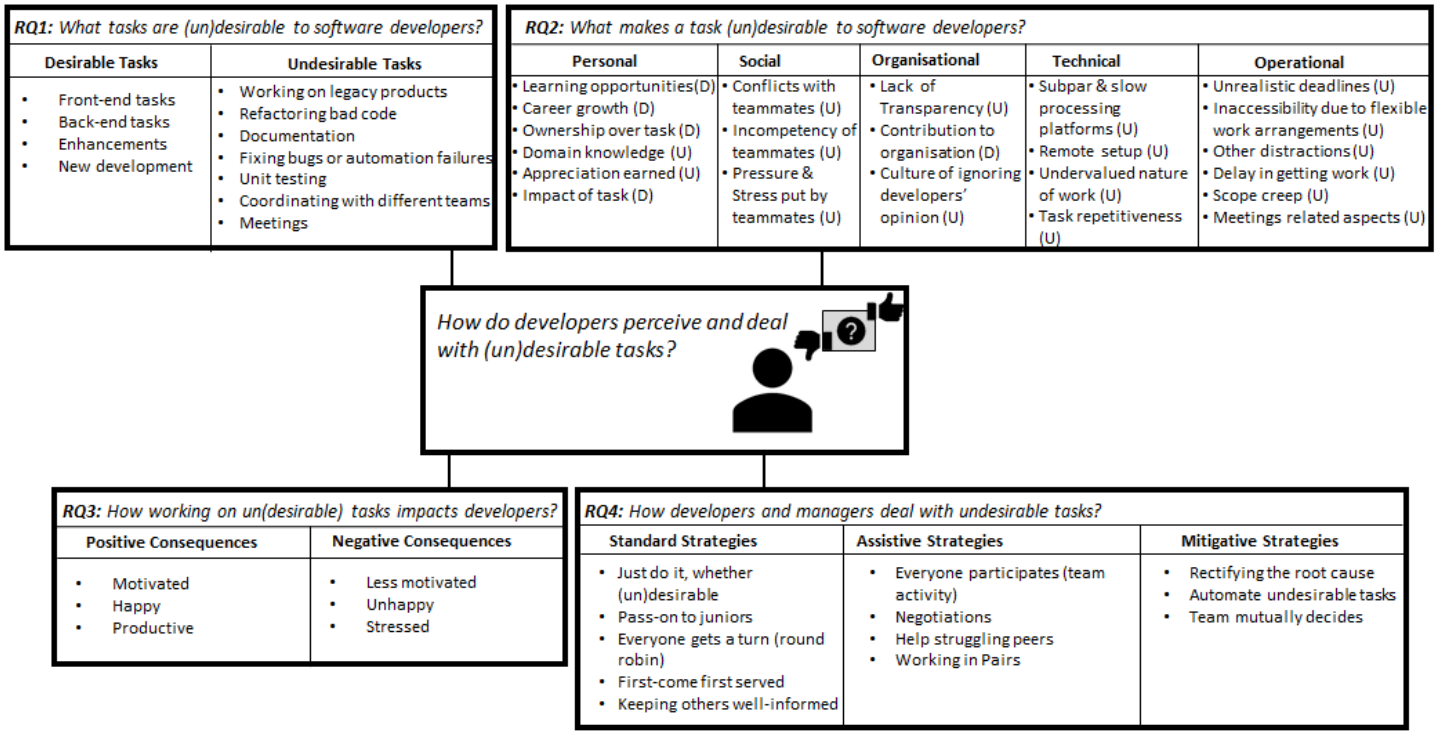

Figure 1: (Un)desirability for different tasks

5.2.1. Personal factors

Personal factors include factors directly related to developers such as learning opportunities, career growth, ownership over task, domain knowledge, appreciation earned, and impact of task on developer.

Learning opportunities: We found that developers desire to work on tasks that involve new and emerging tools, technologies or frameworks. This is primarily because working on such tasks helps them learn new tools or technologies and improve their technical skills as indicated by a participant.

"I liked it [automating build process] coz it was a new experience for me. I designed framework from scratch..... creating new instance on open stack, assigning floating IP, setup ssh connection, installing all prerequisites packages..... etc. and then adding it to Jenkins worker pool" [P22]

Working on a certain technology stack such as Android, iOS, Microsoft, etc., seems to interest some developers making associated tasks **desirable** to them. Some prefer to keep working on the same tech stack, others are happy to switch if the role demands.

Career growth: In addition to a new learning experience, working on new, emerging tools or technologies improve developers' profile, therefore **desirable**. They get to add new skills to their resume which helps them secure a higher rank in current team or even a better job in the future. As indicated by P24, developers take it as an opportunity to upskill, improve their profile, and increase chances of success in the job market.

"These skills are in demand and looks good on CV" [P24]

Ownership over task: Developers enjoy ownership over their work and reported it as another contributing factor in making a task **desirable** for them. They love working on tasks that give them autonomy and freedom. A developer expressed pleasure for getting full ownership and responsibility over their task by their manager as:

"My manager allows me to do a to z for that feature and lets me do all of it which I love." [P23]

Domain knowledge of developers is indicated as another factor. Sometimes developer's lack of domain knowledge makes a task **undesirable**, e.g., working on a core financial or payroll problem can be challenging without domain knowledge. Developers cannot propose a solution until they understand the details of the issues. They get these domain-specific details from the external stakeholders.

Appreciation earned by the developer from any task is reported as another contributing factor. We found instances when not being appreciated by the seniors was demotivating, hence making such tasks **undesirable** for future. Senior developers shared instances when their junior co-workers felt awkward when their work was not considered worthy. A senior participant revealed how their intervention changed co-workers' responses towards a junior developer.

"You can just see difference in how they're treated. I've seen another tester in the team telling them [developers] something's wrong and been given the brushoff. So, I've come in to back them and then, magically they're taking it seriously" [P25]

Participants also indicated that this behavior could be due to other reasons, e.g., the incompetency of the junior or some recent negative reputation that the junior earned from any of their previous work. Other participants revealed how their working relationships improved over time, as indicated below.

"We have been working on it for a long time and with the product our relationship also flourished. Now, whenever there is any new feature or an issue, we sit together discuss stuff before starting. It wasn't the same before I still remember in the beginning it was really hard getting hold of him [developer]. But I think over the time I built a nice reputation" [P22]

***Impact of task** on developer:* The task's *impact* often makes them undesirable. A participant shared an example where a task was **undesirable** because of associated mental fatigue and burnout. A small mistake can lead to repercussions as the changes made were going on the live server simultaneously. The developer found it mentally exhausting and stressful.
"This task needed to update our website with all components and corresponding versions for all supported platforms. It requires a lot of concentration as the data become live simultaneously. It was a simple task, but mentally exhausting". [P22]

5.2.2. Social factors

These include factors that involve influence of others such as conflicts with teammates, incompetency of teammates, and unnecessary pressure and stress put by teammates.

Conflicts with teammates can occasionally lead to **undesirability** for a task. For example, a business analyst or product owner may not admit that the initial requirement they shared with the developer was ambiguous. Instead, they hold the developer responsible for making incorrect changes. This makes such a task undesirable for the developer.
"So, the challenge is, an undesirable task becomes undesirable when somebody is asking for a change because they messed up or whatever and not willing to take the responsibility" [P20]

Incompetency of teammates: Participants reported few instances when working with an underperforming team members made tasks **undesirable**. Developers shared disliking situations when testers report bugs that reproduce in their unique environment or focus on cosmetic issues missing real issues in significant functionality and treat unreproducible issues as showstoppers. Interestingly, a tester shared similar feelings for developers' low-quality work.
"...when you've had bad developers who weren't very good at their job. That ends up being a nightmare. So every time you worked on their stuff you would, you'd just do Oh no! So it's always nicer when you're working with the really good developers when you've got quality code to start with and you've not got the silly issues." [P25]

If there is a friendly environment that supports open conversations between teammates, and if the teammates take constructive criticism positively, this generally fascinates developers. It goes back to what type of relationships and reputations the developers have with their teammates. Typically, developers appreciated discussions that can help them to gain clarity towards tasks. P11 stated
"I mean I work in other projects as well but I've got a senior and junior dev and they're both terrific to work with because they both, they'll both talk to me" [P11]

We also noticed that working with incompetent seniors or even seniors who do not provide details can make a task **undesirable**. In such situations, gathering details, discussing the impact of the changes, telling them if something is working wrong gets challenging, as indicated by a participant.
"We have one who isn't, and unfortunately he's the boss and he's the only one who's not quite up to snuff. So whenever you have to work on something of his, everyone literally does 'Oh no', when you realise you've got something coming through from him. Because it just starts from the very beginning of there's not enough information about what the problem was, what the change was. So you actually end up working blind, trying to figure out what is this thing and is it going to do any damage." [P25]

Pressure and stress put by teammates: Participants reported that there are times when managers and senior teammates put unnecessary pressure and stress, which they don't like. Working under such co-workers makes related tasks **undesirable**.
"We have X in Y who has a habit of putting unnecessary pressure saying, 'What's going on with this bug, its urgent we want it right away!' and then you go back and you can't do it until Friday." [P18]

5.2.3. Organisational factors

These factors are related to organisations developers work for, i.e., transparency given to developers by the organisation, contribution of the developers' work to organisation, and culture of organisation to ignore developers' opinion.

Lack of Transparency in the workplace: Participants reported a lack of transparency as another cause towards tasks **undesirability** and disliked when the management does not keep things open. They are unaware of the impact their work can make. As an example, a developer was quite frustrated about making some changes repeatably. However, developer's views were changed when the lead shared the significance of these changes and the impact on the company.

Contribution to organisation. Another contributing factor was when any piece of work was benefiting a product or company. It was common for a developer to like a task if the company valued it. For example, a participant who mentioned working on automation tasks was exciting and **desirable** because the company valued it, they were aligning with the company's goals.
"Automation tasks are desirable to me.. in my company, automation tasks are valued more than manual ones" [P24]
We noticed these tasks varied with job roles and the company's areas of interest. They change with time, business needs, and technology trends. A different yet related perspective shared by one of the participants is that any task that provides a chance to contribute to people's lives positively makes tasks **desirable**, e.g., working on healthcare products and retail services.

Culture of ignoring developers' opinion: Participants reported that while making any decision, management ignores developers' perspectives or recommendations. These decisions are related to the readiness of product release or any potential critical issue. Developers feel that their say does not matter, making them uncomfortable and frustrated, as indicated by a participant.
"You've got no control over it going out and you know it's not right, and you know it's not ready. So that makes it hugely

stressful, and that's really common. You have no, you don't really have any say over when it gets released. And it's invariably released before it's finished, which they're fine with. But I'm not fine with that so that makes it really hard" [P25]

Developers shared that management do not give importance to product's quality. They are aware of the product's poor quality but do not prioritize bug fixing unless the customer reports, which frustrates developers.

"We do have a very buggy product so I watch our logs all the time. That's like the first thing, every day I have a look at logs to see what problems are occurring in production and see if I can spot what's occurring... which I find quite frustrating" [P23]

5.2.4. Technical factors

These are related to technical constraints developers face such as working on slow platforms, in a remote setup, and undervalued nature of work. They also include factors related to attributes of technical tasks such as repetitiveness.

Subpar and slow processing platforms: make related work undesirable. Adding features in a product on an obsolete platform that customers hardly use is **undesirable**. A developer shared an example where compiling a few lines of code on the X platform takes ages which is very frustrating. A tester agreed that setting up a machine and verifying a few basic level test cases while connecting to a remote machine can take hours for a few minutes of activity.

Remote setup is reported as another contributing factor. Developers working in a remote setting shared that they have to perform pre-requisites before starting any work. A developer reported frustration over a painful process for committing any code keeping the security policies in place. Connecting to remote servers can be frustrating due to the time it takes. Similarly, inaccessibility to remote machines due to network issues is sometimes troublesome making such tasks **undesirable**.

Undervalued nature of work: Developers indicated that working on old, legacy applications is **undesirable**. They find it unexciting as understanding an older technology can be *difficult*, *time-consuming*, and *undervalued*. Working on such outdated applications requires many changes, sometimes doubling the effort, which is neither rewarding nor impactful. Another developer compared why working with such applications is undesirable compared with newer applications.

"A couple of applications that we work with are using older technology and a lot of changes. If any changes need to be made to those applications, you can pretty much take the level of effort you think it's gonna be and double it... the newer applications have that all set up so it's really easy to deploy them. The older ones are very complicated and there's always something that's going wrong."[P19]

Sometimes, a client asks for changes on such platforms that require major architectural changes, potentially impacting other unknown areas of the application. Such tasks are **undesirable** due to complexities.

"I mean a lot of those application migration ones are [undesirable] because some of them can just be kind of endless holes (laughing)" [P19]

Task repetitiveness is another common reason behind an **undesirable** task. Some common examples shared by the participants are retesting the same bug again and again or on different supported platforms, regressions caused as a result of recent changes in the code base, e.g.,

"If dev fixes one defect but the fix causes ripples, like already tested functions do not work anymore, then retesting same thing again and again becomes undesirable" [P23]

It is also reported that certain activities are monotonous and repetitive, yet they need to be done to fulfil the company's SOP.

".. Every time we have any major or minor release of our product, we need to go through an exhausting and repetitive set of activities as part of our company's practice although we have automation suites for them" [P24]

Setting up an environment may involve tedious infrastructure and configuration-related tasks, e.g., setting up multiple different environments which are acknowledged as **undesirable**, not because they are difficult but because they are time-consuming.

5.2.5. Operational factors

These are related to business operations, their planning, and management. These include unrealistic deadlines, resources inaccessibility due to flexible work arrangements, other distractions, bottlenecks and delays, scope creep, and meetings related factors.

Unrealistic deadlines: Participants reported that there are times when management proposes and agrees on *unrealistic deadlines*. Working on such time pressing tasks that involve unrealistic deadline is *frustrating, stressful, exhausting* and **undesirable**. As an example, developers are expected to address the high priority issues reported by the customers on the spot, irrespective of their reporting time which can be on a weekend, after working hours, or a public holiday.

"To keep the things going we need to deliver a customer build as a workaround as soon as possible no matter what" [P22]

If there is not enough time to finish a task properly, it makes developers anxious. In some cases, they are aware in advance that all their *efforts will go to waste*, and the management will release the deliverable before it's even ready.

Inaccessibility due to flexible work arrangements: In a flexible working environment, some developers come to work early or late depending on their convenience, and their unavailability at certain times causes delays which annoys teammates. But for others, it is not a big concern, as shared by a developer who mentioned making a call if there is anything important.

"One team member starts earlier [due to personal reasons] and others join 2-3 hours later. His day ends earlier so sometimes feels like we see him quite less. Though this isn't a big concern we call him for any clarifications if needed but would be nice to have him around like the others" [P24]

Other distractions: Participants reported that involvement in the recruitment such as conducting technical interviews, promoting company's recruitment at a career fair is **undesirable** because it is time consuming and distracts them from regular activities. They are expected to do these tasks in addition to their routine, business as usual tasks.

Bottlenecks and Delays: Many participants, specifically testers reported that the *delay in getting the work* from the programmers causes a bottleneck and leads to frustration which makes related tasks **undesirable**. It is because testers have to work extra hours to meet the project deadlines. The testers shared that they get the code to test closer to the end of the day, and then the programmers expect them to verify the changes as soon as possible. Many testers reported this as frustrating and happening more like a norm, irrespective of their raised concerns.

Scope Creep: We also found that tasks that involve *continuous change in scope* are **undesirable**. They lead to never-ending requirements and communication loops and waste of effort. Developers find working on such tasks unpleasant. They have to keep waiting for clarification from different stakeholders, and the work remains unfinished. For example, developers often have to wait for the clients to respond to queries, clarifications, or approval, keeping them hanging to tasks, and they often complain about this.

Meetings related aspects: cover factors related to meetings such as *duration of meetings, not following meeting agenda, irrelevance to meeting agenda,* and *limited accessibility to clients* make tasks **undesirable**.

Duration of Meetings: Meetings are generally understood to contribute to project success and completion, yet many participants quoted them as **undesirable**. The participants acknowledge the importance of having these meetings but at the same time declared that *prolonged duration of meetings* wastes a lot of their time. What makes it more undesirable is when these meetings do not add any value to the attendees, and there is little or nothing to take away from them.

"It's undesirable for me, like at my current company the meetings don't add value. So, you know, I don't enjoy them, they often feel like a waste of time. But in my previous companies, you know, the meetings were really fun, loved them because hey were really useful" [P25]

Another participant, reported client meetings to be **undesirable**. This is primarily because meetings with the customers take a lot of their time. On the other hand, another participant shared a different perspective towards such activities and acknowledged these meetings as a medium to clarify requirements with the customer.

"I love communicating with developers and BA from time to time. It helps me to understand the feature, to foster a good relationship within the team and to ensure a quality product is shippable" [P21]

Not Following Meeting Agenda: Participants agreed that if meetings do not add any value to the product, there is no point in having them. But if they have a well-defined agenda, are time-boxed, and are limited to the relevant people, they add value by all means. A participant highlighted that their teammates were unaware of the dangers of not following the structure of the daily stand-ups. They were skipping valuable details, (e.g., their current progress, any impediments) and expressed happiness over finishing stand-ups earlier. *"It [stand up] was like basically instead of people actually doing what you're supposed to do in a stand up, saying what you're actually working on. People basically said I'm working on this like this is the area, without any detail and at the end our team leader was like that was the shortest stand up we've ever done, and everybody was like yay!. And I didn't say anything but I was like, do you actually realise that there was nothing of value actually got said in that meeting. We already know what areas everyone's working on"* [P25]

Meeting Style: It's not just about the time and the relevance to the meeting which can make them **(un)desirable**, but also how they are conducted. A participant stated that team meetings on Zoom are more tiring than physical meetings with remote meetings due to multiple reasons such as missing physical contact, connectivity issues, exhausting with missing breaks between back-to-back meetings.

5.3. How working on (un)desirable tasks impacts developers (RQ3)?

We asked developers about the perceived impact of working on (un)desirable tasks and found that many developers shared a common opinion about the negative impact of working on undesirable tasks and positive impact of working on desirable tasks. Not surprisingly, many participants reported increased **motivation** while working on desirable tasks, e.g.,

"Working on desirable tasks affects me positively by giving me a sense of achievement and motivation" [P24].

Some reported decreased motivation while working on undesirable tasks, e.g.,

"working on repetitive, boring ones definitely makes you less motivated.." [P23].

Participants reported **positive emotions** such as **relaxed, happy, less stress** while working on desirable tasks.

"Working on something impactful (desirable) is emotionally rewarding. You feel happy that you are playing a part in some way or the other" [P16]

On the other hand, **negative emotions** such as **loss of interest, low dedication, boredom** were also reported.

"On every release, when we are assigned different platforms for testing, it's always boring to run same steps on X number of platforms. I find this activity useless and have raised my concern too" [P22]

Sometimes, personal emotions like stress, frustration are linked to making tasks undesirable as indicated by the developer.

"I think there is a strong link because when I am stressed or frustrated, I lose my real interest in doing it [task]" [P24].

Participants also reported losing track of time and effort they put in while working on a desirable task. It increases their **productivity** but, on the downside, could impact their well-being as when they work outside office hours, it could get exhausting. But since they enjoy doing it, they don't care as much about how it affects their well-being.

"I feel more productive when I am doing them [desirable] and I do it with all my heart and feel really happy about it. Sometimes doing desirable task make you feel losing track of time coz you are so much into it and my productivity increases." [P23]

We noted some negative responses towards consistent undesirable work. One example is **redesigning entire code** to avoid waste of time and effort.

" So, the senior developer, he took another week, rewrote entire thing and then it got fixed." [P32]
Another, rather extreme response reported is **quitting job** as indicated by a participant.
"I have witnessed colleagues leaving our company because as they said its getting too much for them" [P23].
One participant admitted that recent work had been overwhelmingly frustrating but in current times, due to Covid-19, even desperately wanting for a job switch is not possible due to shrinking opportunities.
"With COVID this is the worst possible time to think of finding another job. Like in my case, I absolutely would. If something came up I'd be gone, it's just been too frustrating" [P21].
Another participant indicated that the duration of working on the task also influences the impact.
"... but working even on an undesirable task is not frustrating if done for a short time but not motivating either" [P2]

There are instances when working on a task reaches far beyond the developer's patience and there comes a point when it starts affecting their **well-being**, as shared by a participant. This could be due to complexity of the problem, e.g., understanding the underlying architecture or logic applied, or regressions due to new changes in the code. One example revealing junior developer's stress as:
"It [retesting tests] was just going in circles. And he got to the point where he was like I've got PTSD (Post-traumatic stress disorder), I can't work on this any longer (laugh). This is the more junior developer, so he was like that's it, I just cannot do it, I'm too stressed, too hard" [P23]

5.4. How developers and managers deal with undesirable tasks (RQ4)?

We asked participants how they respond to (un)desirable tasks. Whether a task is desirable or undesirable to developers, it still needs to be accomplished to add business value and deliver customer's requirements. Yet, participants revealed that there are tasks that remain on the board unless being reminded of or intentionally made visible by the leads or managers. It was found that developers adopt a set of strategies to deal with undesirable tasks.

Three main types of strategies emerged from participants responses which we have described below.
Standard Strategies: are the strategies in which the developers do not take any steps to make undesirable tasks desirable. These are followed as team assignment protocols or norms.

Just do it, whether (un)desirable: A majority of the developers agreed to the fact that all tasks are part of their job, and they have to do it whether they like it or not. Even if they are undesirable, they still cannot be avoided as refusing undesirable is not an option.
"That's part of job so can't ignore, I do my best" [P24]

Pass-on to juniors: It is seen that quite often boring and mundane tasks are assigned to the junior developers. However, sometimes this is to help them understand the system. Juniors are reluctant to bring this up as part of conversation, even to address their own needs.
"So often, the junior will get stuck with, updating text or, you know, mundane tasks" [P28]

Everyone gets a turn (round robin): Another popular strategy evident from the participants is everyone in the team gets their turn at some point.
".....team members work on these tasks in rotations" [P15]
Developers and mangers consider it as a fair strategy to deal with such tasks.

First-come first served: In some teams, the work distribution is such that whoever gets out of work will have to take the next available work. In such cases, it's more of luck than the desire to work on the task.
"The first person who runs out of task (un)desirable above that story will have to grab the task" [P14]

Keeping others well-informed: Developers who cannot voice their concerns or avoid working on an undesirable task stated just keeping their managers or leads well-informed as a strategy.
"I try to keep my manager in loop letting him know that document review task is taking a lot of time" [P24]

Assisted Strategies: are the strategies in which the developers assist other developers to finish an undesirable task.

Everyone participates (team activity): to get the undesirable tasks done. This happens like a round robin strategy where every team member gets an equal share of the work from the list. For example, a team gets installers for different platforms before every major release; though the processes are automated, a set of steps need to pass before they are released. This is considered an undesirable task but important to business, and so platforms are distributed within teams and across teams for smooth execution. They perceive it as an accepted strategy as it reduces the frequency of working on a task for every developer.
"The best part is we all contribute to finish this [installer smoke testing] unavoidable task on our chosen platform" [P22]

Negotiations: Team members negotiate with each other to work on or avoid any task which happens very commonly. These happen during team meetings and discussions.
"They will say 'Look, I know you've taken this card. Do you mind if I do it, I've got particular skills in this area'?" [P28]

Help struggling peers: Developers often ask for help or even re-assign undesirable tasks to others within teams as another strategy.
"If someone struggles they are encouraged to call out for help and that works most of the time" [P11]
"We often swap tasks we don't want to work on" [P17]

Working in Pairs: These negotiations usually end with the team members working in pairs for quicker completion in case of undesirable tasks or learning for desirable tasks.
"If two are interested in one, then we wire them up together. People are, 'Oh, you want to fix that, I have a clue maybe, we can introduce that in the framework to resolve that'" [P31]
Mitigation Strategies: are the strategies in which team members make collective efforts to mitigate the undesirable tasks.

Rectifying the root cause: We have seen in Section 5.2, that many times it is not about the task, rather it is the underlying factor that makes it undesirable. A manager quoted that

"Writing code is fun to every developer whether it's to develop a new feature or changing an already build thing" [P20]
It is the responsibility of a manager to identify the root cause behind an undesirable task and rectify it for the team through discussions. Unfortunately, not many managers do this. On the contrary, a few developers stated if they see something making a task undesirable, they try to communicate with the manager as soon as possible. This way they try to prepare or deal with them minimising any waste of time, effort, or money.
"So he was like I'm just going to rewrite it, and we all thought by that point that we should've started with rewriting the code because it was just too complicated which is why we just had issue after issue. So he started with rewriting it" [P25]

Automate undesirable tasks: Another commonly practiced strategy to overcome repetitive, mundane tasks is automating them. This not only helps them focus on useful work, but also reduces the possibility of human error.
"We try to automate letting us focus on productive stuff" [P31]

Team mutually decides: is a future mitigation approach in which teams bring undesirability related concerns up at retrospectives, and mutually address *"Is there anything we can do?"* Then, the team makes a decision on how they want to approach that sort of tasks in the future collectively.
"You've got to bring everybody in and have a reasonable conversation and appeal to reason" [P20]

6. Discussion

Here we discuss our findings in light of related work, touch on additional aspects that could be investigated in future work, and finally present how we mitigated threats to validity.

6.1. Comparison to Related Work

Researchers acknowledged the need to focus on software developers and their happiness to improve developers' productivity and quality of software products more than thirty years ago [17]. This study is an attempt to explore task (un)desirability. While comparing our findings (Q1 and Q2) to related work, we found that a limited number of studies have explored the emotions and behaviours of developers for software development tasks. A study explored causes for unhappiness among software developers and reported them as internal and external causes [4]. Some of these are related to tasks, i.e., *being stuck in problem solving, time pressure, bad code quality and coding practice, mundane or repetitive task, unexplained broken code*. Some of the causes reported by them, e.g., time pressure, mundane or repetitive tasks are common to our findings. In this study, we explore why developers want and don't want to work on certain tasks. This study is an initial step towards understanding tasks desirability as another indicator of developer's happiness, a less explored area in software engineering. It extends the limited literature within software engineering on desirability and undesirability for different tasks. We reported not only the contributing factors that make tasks undesirable but also the factors that make tasks desirable. Our findings suggest that contributing factors are not limited to technical aspects. They are also related to other aspects, e.g., social, organisational, operational.

In relation to Q3, several studies reported a link between developers' emotions with their performance [18]. Positive affective states were perceived to enhance developers' productivity, while negative states such as frustration deteriorated productivity. Another study by Ford and Parnin [5] explored frustration in software engineering. Some of the causes for frustration were related to factors such as *learning curves of programming tools, too large task sizes, the time required to adjust to new projects, lack of resources, perceived lack of programming experience, fear of failure, internal hurdles and personal issues, limited time, and issues with peers*. Developers' happiness and impact on performance have been investigated in multiple studies. These studies have confirmed links between affect, emotions, politeness, and software quality [19, 20]. In contrast, emotions such as anxiety and burnout link with issues priority [21]. Interestingly, shorter resolution time is associated with positive affect, emotions, and polite attitudes [19, 20]. Prior research on software tasks has covered variations in tasks and attitudes of teams while working on different forms of tasks. For instance, an empirical study explored the influence of personality types in software task choices [22]. In that study, opposing psychological traits, such as extroversion–introversion, sensing-intuition, thinking-feeling, and judging-perceiving were mapped to software development tasks to find evidence of relationships between software engineers' Myers Briggs Type Indicator (MBTI) types and role preferences. Another study explored the relationship between task type and team attitudes [23]. That study found that teams expressed different attitudes when working on various forms of software tasks. While working on the support, enhancement tasks, or resolving defects, teams expressed attitudes such as positive indicated through words, e.g., *beautiful*, *relax*, *perfect* and negative indicated through words, e.g., *hate*, *suck*, *stupid*. This study further explores the behaviours of developers from a different dimension i.e., working on (un)desirable tasks and confirms that working on desirable and undesirable tasks have positive (e.g., happy, motivated) and negative (e.g., frustrated, stressed) impacts on developers.

Recent studies confirm that happy developers also perform better than unhappy developers and providing incentives and flexible working environments makes them happy and more productive [24, 25]. Multiple studies have explored general indicators of developers' motivation and happiness such as job security, sense of belonging, supportive relationships, rewards and incentives, increased pay, recognition, etc. Managers and team leaders of software companies take care of developers' happiness in several ways. They provide different perks and benefits to teams and individuals, such as fun things to do, providing food during working hours, or flexibility to work from home [4].

Emerging research on behavioral software engineering confirmed a positive relationship between developers' happiness and work-related constructs. In the past, Hall et al. found that specific work-related constructs such as a variety of work, task

significance, and technically challenging work are reported to contribute to developers' motivation [25]. While these studies indicated different ways to keep developers happy, our study on the other hand reveals that working on desirable tasks also contributes to developers happiness. So, managers should also keep this aspect into consideration. To the best of our knowledge, not many studies explicitly studied (un) desirability of tasks and specifically how teams, managers, and developers deal with undesirable tasks. A prior study pointed out strategies to overcome challenges introduced by the self-assignment practice. For example, strategies such as pairing up with experienced resources, offering work, self-assigning the next available task, informal team discussions and negotiations, fixed work assignment were used [15].

6.2. Implications

Our findings indicate that (un)desirability is personality-dependent and varies from developer to developer. It is also context-dependent and changes with situations and time. It is quite likely that two developers with the same domain knowledge, experience, and skill sets may not like the same tasks at any or all times. It is also common that working on new technologies or tools fascinates a developer at the beginning and working on certain technologies or tools too frequently or for a long duration makes it boring for them. Some developers get fed up, bored, frustrated and so keeping them occupied with exciting pieces of work is challenging. It would be beneficial to investigate how managers and teams can keep work exciting for developers. Both developers and managers agreed that the task's priority takes precedence over the task's (un)desirability in practice. It indicates that the developer's liking is not a preference, and they keep working on such tasks even if it's not fascinating.

Our findings also suggest various constructs such as developers' personality and their professional traits such as working experience, skill set, nature of their work, i.e., high-severity and priority work items (e.g., support issues) impacts (un)desirability for tasks. We noticed that task (un)desirability varies with project contexts such as product type and team culture. A participant pointed out that if teammates or the manager appreciates developers' effort they put on an undesirable task, appreciation gains precedence over the task's undesirability. Similarly, an undesirable task (due to complexity, missing prior knowledge, etc.) can become desirable with support as shared by P23.

"One example is of API test where I didn't had prior knowledge but because of supportive peers it became desirable". [P23]

We have noticed variations around likes and dislikes within the same roles. Some programmers or testers have different preferences towards tasks compared to other programmers or testers. For some developers' front-end work is boring, tedious, while the back-end is appealing. For others, working on front-end development involving a user interface is more desirable. Similarly, enhancements and bug-fixing are typically not desirable tasks, but some developers said they were exciting as they provide opportunities to explore the system before any modifications or fixation. Others stated bug fixing was good, but reproducing is frustrating and time-consuming. We know that a team comprises members with different traits, likes, and dislikes, and this study confirms that team members have different (un)desirability for tasks. Future research can further investigate how these likes and dislikes are catered at the team level by looking at the relative importance of the reported strategies and how well these strategies balance varying individual (un)desirabilities from a limited pool of tasks. Future research can find out how do managers balance between undesirable tasks among developers with different traits or how easy or hard is it for a team to handle these tasks with developers of different traits.

We also noted that (un)desirability varies with the job tenure. Novice developers are still enthusiastic to work on all tasks to gain some experience. But as developers advance in their careers, they tend to be more selective about their work. They switch companies, departments or teams if they don't find the work desirable and fascinating. We noticed that some participants did not care much about whether tasks are desirable or undesirable and treated them as part of their job. We noted this specifically in junior participants and participants on term or contractual roles. A possible reason could be that the contractual roles are for a fixed tenure. After the completion of the contract, the developers naturally get to work on a variety of tasks. Similarly, graduates and novice developers are open to all types of tasks for learning and experience purposes. It needs to be explored further if these are the only developers who claim not to care about the type of work and get to work on monotonous and mundane tasks. It would also be interesting to further examine how working on such tasks impacts the quality of their work and how they stay motivated and maintain their productivity while working on undesirable tasks.

We also found instances of undesirable tasks becoming desirable for developers. For example, the former company did not value the task making it undesirable for the developer, but a similar task was significant to the current company and its business so it became desirable for the developer. Some managers are aware of their team members' (un) desirability for certain tasks. They try to make undesirable tasks desirable. For example, a participant, P23, was shy of public speaking, and shared how their manager helped them change their behaviors and choices over time. The manager's support turned an undesirable task of presenting demos into a desirable task. Sometimes, managers discuss these aspects with the developers in one-to-one meetings. Few managers would not regard tasks (un)desirability unless team members raise them. Future researchers can study how managers can support and motivate developers to deal with undesirable tasks and what support is more effective for developers.

6.3. Threats to Validity

We mitigated the potential threats to validity by considering *reliability*, *construct validity*, and *external validity*. The *reliability* of research findings can be impacted by researcher bias. To mitigate researcher bias, we conducted extensive discussions amongst all authors on the data collection, analysis, and results to ensure mutual consensus, understanding,

and cross-verification. *Construct validity* relates to the extent to which the scales, metrics, and instruments used, actually test the hypothesis, theory or properties they are supposed to measure [26]. To address the threat of *construct validity*, instruments used for data collection (interview guides and pre-interview questionnaires) were developed, reviewed, and revised iteratively through discussions between the research group members throughout the study. Also, we conducted a pilot study to ensure that the interview questions were understandable and sufficient to collect the desired data within a suitable time frame. Regarding threats relating to *external validity* and *generalizability* of the research findings, we recruited participants through multiple channels (e.g., social media, networking platforms). To ensure diversity in our data set, we included participants from different age, experience, gender, role, company size, and project domain. Still, we do not claim generalizability due to the subjective nature of the study. However, we did have a suitable number of participants for a qualitative study [27, 28].

Our findings report more on tasks undesirability than desirability. It is because participants have provided more details on undesirability. Generally, humans recall negative experiences more easily than positive experiences [29]. We acknowledge that answers to questions Q1 and Q3 are less detailed than others. Grounded Theory procedures's open coding and constant comparison methods allow important concerns of the participants to surface. We report on the four key concerns that emerged from coding. However, given the semi-structured nature of the data collection, typical of Grounded Theory studies, some aspects can be more evolved and detailed than others, as is the case with the contributing factors and strategies reported in response to RQ2 and RQ4. It does not imply that the other aspects were not important; rather, they provide the necessary contextual background and supplementary information to complete the findings. Future studies could focus on exploring these aspects in more depth.

Future studies can further explore the desirability and undesirability of software developers for different tasks through a qualitative or quantitative survey with a rich data set involving software developers from different demographics like gender, experience, or age and involved in both corporate and open-source software (OSS) development. Future researchers can uncover these and other related aspects in-depth, exploring the impact on software developers' productivity, motivation, happiness, well-being, professional growth, and personal life. We provided a variety of strategies (in Section 5.4 for teams to manage (un)desirable tasks. We suggest teams' periodic reflection on these strategies to create a balanced distribution of (un)desirable work among team members. Future researchers can investigate the effectiveness of the reported strategies specifically looking at how some strategies could be productive and counterproductive for the teams and individuals. Our findings indicate a link between task undesirability and negative emotional responses from working on undesirable tasks (in Section 5.3). Exploring this link further in future studies would be interesting to see if they provoke and progressively aggravate each other over time.

7. Conclusion

The (un)desirability of tasks is subjective and varies across software developers. Developers perform various tasks while developing any software, which include both technical and non-technical tasks. This study explored (un)desirability for different tasks based on data from 32 software practitioners. Our findings confirm that developers have certain (dis)likes towards specific software development tasks and working on (un)desirable tasks can impact software developers. We provide a set of factors that makes tasks (un)desirable to developers. We identify how software development teams and managers deal with undesirable tasks to ensure a fair work distribution using different strategies.

Acknowledgements

We thank all participants who contributed to this study. We would also like to thank our anonymous reviewers for their feedback that helped to improve the quality of this work. This study was conducted under approval from the Human Participants Ethics Committee at the University of Auckland. This research was partially funded by the Natural Sciences and Engineering Research Council (NSERC) and Google Research Faculty Award.

References

[1] A. Meyer, E. T. Barr, C. Bird, T. Zimmermann, Today was a good day: The daily life of software developers, IEEE Transactions on Software Engineering (2019).

[2] M. Konopka, P. Navrat, Untangling development tasks with software developer's activity, in: 2015 IEEE/ACM 2nd International Workshop on Context for Software Development, IEEE, 2015, pp. 13–14.

[3] S. A. Licorish, S. G. MacDonell, Exploring software developers' work practices: Task differences, participation, engagement, and speed of task resolution, Information & Management 54 (3) (2017) 364–382.

[4] D. Graziotin, F. Fagerholm, X. Wang, P. Abrahamsson, On the unhappiness of software developers, in: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, ACM, 2017, pp. 324–333.

[5] D. Ford, C. Parnin, Exploring causes of frustration for software developers, in: 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, IEEE, 2015, pp. 115–116.

[6] A. Murgia, P. Tourani, B. Adams, M. Ortu, Do developers feel emotions? an exploratory analysis of emotions in software artifacts, in: Proceedings of the 11th working conference on mining software repositories, 2014, pp. 262–271.

[7] K. Madampe, R. Hoda, P. Singh, Towards understanding emotional response to requirements changes in agile teams, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, 2020, pp. 37–40.

[8] Z. Masood, R. Hoda, K. Blincoe, What drives and sustains self-assignment in agile teams, IEEE Transactions on Software Engineering (2021).

[9] J. W. Brackett, Software requirements, Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST (1990).

[10] T. Jones, Why choose case?, American Programmer 3 (1) (1990) 14–21.

[11] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, T. Fritz, The work life of developers: Activities, switches and perceived productivity, IEEE Transactions on Software Engineering 43 (12) (2017) 1178–1193.

[12] Z. Wang, Y. Feng, Y. Wang, J. A. Jones, D. Redmiles, Unveiling elite developers' activities in open source projects, ACM Transactions on Software Engineering and Methodology (TOSEM) 29 (3) (2020) 1–35.

[13] R. L. Glass, I. Vessey, S. A. Conger, Software tasks: Intellectual or clerical?, Information & Management 23 (4) (1992) 183–191.
[14] A. E. Milewski, Global and task effects in information-seeking among software engineers, Empirical Software Engineering 12 (3) (2007) 311–326.
[15] Z. Masood, R. Hoda, K. Blincoe, How agile teams make self-assignment work: a grounded theory study, Empirical Software Engineering 25 (6) (2020) 4962–5005.
[16] A. Straus, J. Corbin, Basics of qualitative research: Grounded theory procedures and techniques (1990).
[17] B. W. Boehm, P. N. Papaccio, Understanding and controlling software costs, IEEE transactions on software engineering 14 (10) (1988) 1462–1477.
[18] M. R. Wrobel, Emotions in the software development process, in: 2013 6th International Conference on Human System Interactions (HSI), IEEE, 2013, pp. 518–523.
[19] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, R. Tonelli, Are bullies more productive? empirical study of affectiveness vs. issue fixing time, in: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, IEEE, 2015, pp. 303–313.
[20] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, R. Tonelli, Software development: do good manners matter?, PeerJ Computer Science 2 (2016) e73.
[21] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, M. Ortu, Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity?, in: Proceedings of the 13th International Conference on Mining Software Repositories, 2016, pp. 247–258.
[22] L. F. Capretz, D. Varona, A. Raza, Influence of personality types in software tasks choices, Computers in Human behavior 52 (2015) 373–378.
[23] S. A. Licorish, S. G. MacDonell, 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.
[24] S. Beecham, N. Baddoo, T. Hall, H. Robinson, H. Sharp, Motivation in software engineering: A systematic literature review, Information and software technology 50 (9-10) (2008) 860–878.
[25] T. Hall, H. Sharp, S. Beecham, N. Baddoo, H. Robinson, What do we know about developer motivation?, IEEE software 25 (4) (2008) 92–94.
[26] P. Ralph, E. Tempero, Construct validity in software engineering research and software metrics, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, 2018, pp. 13–23.
[27] J. W. Creswell, C. Poth, Qualitative Inquiry & Research Design Choosing Among Five Approaches, Sage Publications. Thousand Oaks, CA, 2007.
[28] N. K. Denzin, The discipline and practice of qualitative research. in. nk denzin & ys lincoln, Handbook of qualitative research (2005) 1–42.
[29] R. F. Baumeister, E. Bratslavsky, C. Finkenauer, K. D. Vohs, Bad is stronger than good, Review of general psychology 5 (4) (2001) 323–370.