Developer Experience

Anders Nylund

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo TBA

Supervisor

Prof. Pirjo Professor

Advisor

Dr Alan Advisor

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Anders Nylund

Title Developer Experience

Degree programme Computer, Communication and Information Sciences

Major Software and Service Engineering **Code of major** SCI3043

Supervisor Prof. Pirjo Professor

Advisor Dr Alan Advisor

Date TBA **Number of pages** 40 **Language** English

Abstract

The abstract in english

Keywords Developer Experience, Software Projects

Aalto University
School of Science

Aalto-universitetet, PB 11000, 00076 AALTO
www.aalto.fi
Sammandrag av diplomarbetet

Författare Anders Nylund

Titel Developer Experience

Utbildningsprogram Computer, Communication and Information Sciences

Huvudämne Software and Service Engineering **Huvudämnets kod** SCI3043

Övervakare Prof. Pirjo Professori

Handledare TkD Alan Advisor

Datum TBA **Sidantal** 40 **Språk** Engelska

Sammandrag
Sammandrag på svenska

Nyckelord Nyckelord på svenska, temperatur

Preface

I want to thank everyone so far that has shown interested and been involved in this thesis. Even if the thesis is still in it's early phases, I am surprised of how many have showed interest towards it and offered a helping hand.

After the thesis has been finalized there will probably be many more appreciations to give

Otaniemi, Date to te announced

Anders Nylund

Contents

Thesis dictionary

API	Application Programming Interface
BS	Brainstorming
DX	Developer Experience
EBS	Electronic Brainstorming
EM	Extrinsic Motivation
GL	Grey Literature
HCI	Human Computer Interaction
IDE	Integrated Development Environment
IM	Intrinsic Motivation
MLR	Multivocal Literature Review
MSECO	Mobile Software Ecosystem
NBS	Nominal Brainstorming
OSS	Open Source Software
PAW	Performance Alignment Work
SE	Software Engineering
SLR	Systematic Literature Review
TBS	Traditional Brainstorming
UX	User Experience

1 Introduction

Software development and software engineering is a complex practice that requires both technical and social skills. Compared to other engineering professions, software engineering is a relative new field of practice and study. The practices deemed as "best practice" are still evolving, and new ideas of good practices are being developed and previous ideas are discarded.

Developing and creating software is a social activity that requires both technical and social skills from the developers. Deep technical skills and understanding is required to be able to implement the wanted artifact or end product. However software engineering is a highly social activity, and therefore it has been noted that human factors are the most important when regarding software development performance (DeMarco and Lister, 2013).

Software developers are in an interesting role where they are both creators and designers when they write the code and design the logic that makes up the software. Meantime they are also users of tools that they use to create the software. Developers using a software product that aids them in their creative design work will create an user experience. Human Computer Interaction (HCI), a traditional field of research, studies the interface and interaction between computers and humans. User Experience (UX) is another field of research. UX includes the aspects of HCI, but on top of that includes also emotions and the user's perceptions of the product. UX can be seen as a more hedonic than a pragmatic approach of studying and understanding the usage of a software product citation?.

In recent scientific research and internet articles and blog posts, a concept called Developer Experience (DX) has gotten traction. DX is a term that explains how developers experience the practice of developing software, both technically and socially. The same way as UX is considering the user of a system or tool, DX can be seen as the experience that developers have as users of a complex system. In the case of DX the system includes the everything around the developer, and everything that affects the software development practice.

DX is more prevalent, and therefore also more interesting, in contexts where development happens in teams. DX of individual developers is also important, but a big part of the experience stems from interaction with team members and other developers. Individual developers are aiming more towards creating an individual DX of e.g. their development environment or tools that they use.

In this study we will focus on understanding what DX is and make an attempt to map out how DX is defined in a software consultancy company.

1.1 Motivation

At the time of writing (autumn 2019), a quick search with the keyword *"Developer Experience"* on google.com gives as a result mostly articles on how framework and library authors should consider their user's (developer's) experience with using the product (tool, library, framework). Also, performing searches with the same *"Developer Experience"* keyword on known libraries of conference papers like Google

Scholar and IEEExplore, the content and topic of the results vary much. This shows that there might not be a common and well known definition of what DX is.

In some research the term *Developer Experience* with the abbreviation of DE^x is used (Fagerholm and Münch, 2013), in some other research the term *Programmer eXperience* and abbreviation *PX* is used (Morales et al., 2019), and finally maybe the most common abbreviation is *DX*. This shows that there is still some ambiguity to the terms and definitions in scientific research. Additionally, most results when searching with the term *Developer Experience* gives results about the experience and knowledge level of a developer in e.g. terms of years working in the field of software development or amount of contribution, and not the hedonic and pragmatic experience of participating in development work.

DX has been studied previously, but research on it is still lacking the connection to practical applications. This is one the biggest motivators for this thesis, as the topic is novel and there is huge potential in improving software development processes, and thereby also potentially improve the e.g. performance, quality, and outcome in software projects.

There is possibly huge value that can be gained from studying DX and learning about how it works. A better understanding of DX can help organisations, teams, and individual software developers to create a better experience that enables them to benefit from it in multiple different areas.

1.2 Research problem and questions

Easterbrook et al. (2008) encourage practitioners to document and reason the selection process of the research problem and questions, the philosophical stance, and the selected research methods e.g the research protocol. They encourage this because other researchers can then understand and interpret the study and possibly replicate the study.

During this study the research problem and questions have evolved and been modified while more understanding and knowledge about the research topic has been created and accumulated. The starting point of the study was to understand how DX is linked to software project outcomes. However, this was noted to be too vague, difficult to measure and difficult to research. The current state and understanding of DX does not allow to research correlation and causality of DX to software project outcomes as defined by Easterbrook et al. (2008).

Based on this, the selected approach for defining the problem and the research questions leans towards stating a exploratory research problem and research questions.

Research problem: How is Developer Experience defined and what are the aspects of Developer Experience that are valued by software practitioners?

To analyze the research questions, the categorization, classification, and guidelines of Easterbrook et al. (2008) will be used.

RQ1 is a Description and Classification, but also a Descriptive-Comparative question that compares two different sources of literature. The comparison helps to better understand the definition of DX, and creates the ground for this thesis. The answer to this question will help to understand the phenomena better, but also point

RQ 1	What is the definition and aspects of Developer Experience, and how do they differ between scientific literature and literature written by practitioners?
RQ 2	How is Developer Experience and its aspects defined by different roles in a software consultancy company?
RQ 3	What aspects of Developer Experience do software developers in a software concultancy company value and see important?

Table 1: The research questions

out the absence of definitions. This question

RQ2 is a Description and Classification question and tries to find the specific aspects of DX that exists in a software consultancy company. The question builds upon the first research question, but takes practitioner's view of point.

RQ3 is Description and Classification question that leans towards being a Design question. Mainly it is however a knowledge question that tries to understand the construct of DX, and does not try to design or implement new or better practices in software development.

All of the research questions are exploratory and they try to understand the underlying phenomena, i.e. DX. Because there is a vague and undefined foundation to build upon, it is not an option ask *relationship*, *correlation and causality*, or *design questions* questions The nature of these questions will guide the research and guide with selecting the used methods and techniques.

1.3 Scope and focus

Scope and focus will be defined later. This can still vary quite a lot as it depends on basically everything, including the research problem and questions.

1.4 Structure of the thesis

This will be finalized later

1. Introduction
2. Background
3. Research methods
4. Results
5. Summary
6. Conclusions
7. . . . ???
8. Profit!!!

2 Background

The concept of Developer Experience (DX) introduces and is related to concepts, frameworks, thoughts, models, and ideas that require introduction and discussion. In the following subsections we discuss the relevant topics and explain the background behind the themes in this study. The topics and themes discussed are a result of an informal literature search on the concept of DX and the results of the systematic literature. They are topics that are related to DX and understanding these individual topics help to understand the complex nature of DX. The systematic literature review on the definition of DX is presented in section 4.

2.1 Developer Experience

Current scientific literature on the definition of DX is few in numbers. DX has however a comprehensive and detailed definition by Fagerholm (2015). Their doctoral thesis dives into the core of developers and their experiences with developing software. They define DX into two different environments, a social and a technical environment. These two dimensions are presented in figure 1.

Social aspect of software development plays a crucial role on how a developer experiences the development practice, and is receiving as much consideration as the technical environment in figure 1. It has for long been noted that the social and human factors of software development are not considered as important as they should be (Capretz, 2014).

The technical environment, including programming languages, infrastructure, processes, techniques, plans, diagrams etc., is also part of the DX. A developer is interacting with these artifacts and that generates an experience. Activities with these artefacts are both experienced as an individual but also as an group.

Time-wise, the DX can be both short term impulsive, or related to one event in software development, but it can also be a long term experience over a period of time, in e.g. a software project.

Fagerholm and Münch (2013) takes an approach from psychology, and divides DX into three different sub areas or categories – cognitive (How developers perceive the development infrastructure), affective (How developers feel about their work), and conative (How developers see the value of their contribution). This conceptual framework is presented in figure 2.

DX can be seen as important from multiple different viewpoints. From a viewpoint of project management a better DX can help to understand, evaluate, and plan projects so that they are inline with the three different dimensions of DX. Another viewpoint is when designing a software development platform or environment, it can be beneficial to understand what impacts and affect the DX, so that the platform or environment can be designed to be aligned with the developers using it Fagerholm and Münch (2013).

Figure 1: The Developer Experience Concept (Fagerholm, 2015)

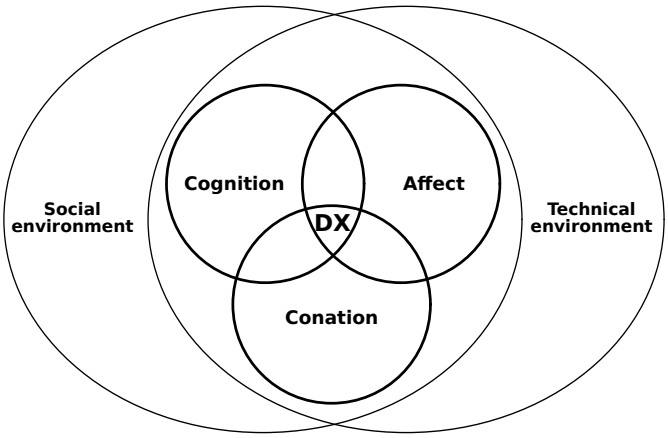
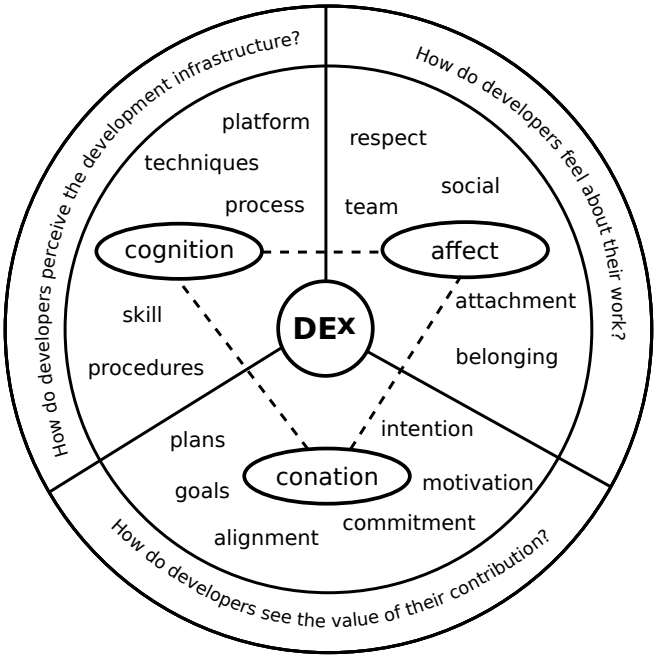


Figure 2: Conceptual framework of Developer Experience (Fagerholm and Münch, 2013)



2.2 Programmer Experience

A software developer is a person with a bigger responsibility than a programmer. If a programmer is following instructions, requirements, and guidelines, the developer is also finding out what the instructions, requirements and guidelines should be and probably also helps in defining them. Therefore DX is also considering more of the surrounding context than what Programmer Experience (PX) is considering.

DX and its related terms have been studied and researched relatively little at the moment of writing (autumn 2019). Morales et al. (2019) performed a literature review of the term *"Programmer Experience"*, that studied 73 articles that matched their defined search criteria. The study concluded that there is still some ambiguity in the term *Programmer Experience* in the context of programming environments, design documents, and programming codes.

Developer Experience (DX) is a bigger construct than PX. DX includes also the motivation of developers, and not only the artefacts like the programming environments (Morales et al., 2019). Developer Experience is considering also the social aspect of being a software developer. Developer Experience is what is felt by the developer while trying to achieve a goal i.e. completing a project

Programmer Experience (PX) can be defined as *The result of the intrinsic motivations and perceptions of programmers about the use of development artifacts* (Morales et al., 2019). A programmer can be seen as person who gives exact instructions on how a program should behave and function. PX is based on the study mainly related to the programming environment, but also programming codes and Application Programming Interfaces (API).

2.3 Selection of tools

Perceived choice is a perception of that the choice has already been made (Kuusinen et al., 2016). Selecting tools in software development projects is in a crucial role, as it can significantly improve the Developer Experience in software projects.

Kuusinen et al. (2016) studied Integrated Development Environment (IDE), and how they are connected with state of flow, intrinsic motivation, and user experience. Their findings reveal that if the developers have a high perception of choice, the also are overall more satisfied with the tools. They also concluded that if the selected tools are selected without their input, (they perceive it chosen already), the developers will have a worse developer experience with it, as e.g. their frustration with the tool will be more common.

There has been a study on the Developer Experience of IDEs (Kuusinen, 2015). However, the study concentrated on the UX of the selected IDE that was studied.

When selecting an IDE it is also important to consider what the other developers in the team or organization is using or what other would prefer to use.

Palviainen et al. (2015)

There can be situations when two different developers use a different IDE, and therefore also the experience can be completely different between them. At the most extreme the 2 IDEs are not compatible with each other as their files related to the

project are different. An example of this is Eclipse and IntelliJ IDEA as Java IDEs.

In a study of IDEs (Kuusinen, 2015), the survey in the study produced answers that were most pragmatic, but not hedonic. This could show that most of the developers are practical, and not feeling based. This has also been proven (Capretz, 2003). This might also be a reason why Developer Experience has not gotten that much attention yet, as big part of people in software engineering are *"Introverts"*. Software engineers are also more logical thinkers than feeling based. As Developer Experience is focusing on the feelings and subjective opinions about things, it might be a difficult topic to research.

2.4 Organization and project onboarding

Mäenpää et al. (2017) have studied the process of entering an ecosystem from the perspective of developers. They have especially focused on the onboarding process of Hybrid Open Source Software projects that have special characteristics as software projects.

Open Source Software (OSS) is software developed, where the source code of the software is accessible to the public. OSS is often developed by a community where anyone participating is allowed to report problems, suggest changes, and also modify the code and develop the software. However, the fact that anyone interested can participate in the community results in that the communities often create their own complex organization and hierarchy.

A hybrid OSS can consist of individual developers, but also sponsored and supported developers that are pursuing the interests of some other organization. One example of this is JavaScript framework React, that was initially developed at Facebook, but later open sourced. Hybrid OSS creates another layer of complexity of the organization.

Mäenpää et al. (2017) argue that onboarding and welcoming new developers to hybrid OSS might be more difficult than normal OSS projects. All in all, onboarding of any kind of software project is part of the whole DX of the onboarding developer.

2.5 Motivation

Intrinsic Motivation (IM) is the motivation that is enabled by someone enjoying their own work, i.e. the motivation is originating from the work itself. Extrinsic Motivation (EM) is motivation that stems from the outcomes of the work performed (Kuusinen et al., 2016) (Self-determination theory. Handbook of theories of social psychology).

2.6 Performance Alignment Work

Fagerholm et al. (2014) and Fagerholm et al. (2015) have created a framework called Performance Alignment Work (PAW), that explains the phenomena of experiencing performance in software development context. Software development performance

is a complex construct where performance measurement is not a straightforward practice.

The PAW framework acknowledges that performance can not be measured trough some objective measures, as there are too many different viewpoints to measure software project performance from. It also acknowledges that performance exists on multiple different levels e.g. individual, team, organization, or customer level. Their study concluded that high-performing teams are considered high-performing because the are able to alter the ways the performance of the team is measured.

The performance of a software development project is highly linked with the DX of the individual and the whole team, and Fagerholm et al. (2014) suggest that by aligning affective and conative aspects with individual developers and within the whole software development team, there could be opportunities to reach improved performance.

2.7 Happiness of developers

Happiness of developers has been reported have high impact on the practice of software development and have consequences. A series of studies, namely Graziotin et al. (2017c), Graziotin et al. (2017b), Graziotin et al. (2017a), and Graziotin et al. (2018) studies the happiness and unhappiness of developers. They concluded that the (un)happiness of the developer has consequences on the themselves, the process, and the end product. The concept of DX (Fagerholm and Münch, 2013) includes the affective dimension that is directly related to the happiness of developers.

2.8 Flow state

The flow state is a state where the task at hand has gotten full attention (Kuusinen et al., 2016). Flow state is something that many developers want to achieve. For some developers it is really difficult to focus if there are external things that disturb them like sound or something similar. Also, people coming and asking questions might disturb or interrupt the flow state. Therefore many developers are now also trying out remote work where they are not co-located. Achieving flow state requires a clear set of goals, continuous feedback, and a good balance between skills and challenge.

2.9 User Starting Experience

Murphy et al. (2018) introduce the concepts of *0 to 200* and *Time to Hello World*. These both concepts are discussing the approachability of APIs. Developers have become more and more in charge of the decision and selection of the tools and 3rd party products that are used when developing software. This might have been more of a responsibility of the organization or the directors or managers before. Now however, it has been seen that the developers are the most knowledgeable persons to make these decisions. Murphy et al. (2018) argue that the first encounter with the API determines if it's selected for usage or not, especially if there is a set of different

options to choose from. Therefore they also say that the *0 to 200* and *Time to Hello World* are important things to consider when developing APIs with a good DX.

0 to 200 comes from the Hypertext Transfer Protocol HTTP code 200 indicating a successful OK response. Adopting a new API and successfully calling the API with a 200 OK response can be seen as the minimal effort to get the integration working.

Time to Hello World comes from the introduction and adoption a new programming language, where often the first task is to print or log the string "Hello World" to the console or terminal. Adoting the basic structures and building block of the programming language prove how easy it is to get going with it, and therefore can be a measurement of the approachability of the language.

However, this shouldn't be limited or restricted to only APIs, programming languages, or frameworks. The concept of User Starting Experience could be used in the measurement of introducing or adopting anything new related to software development e.g. a new development technique or process.

3 Research methods

During the study, the research problem and the research questions evolved as the understanding and comprehension of the problem improved. Because of this the planned research methods and the goals with these also evolved and were reshaped when new information was acquired. From the introduction and motivation in sections 1 and 1.1, it was determined that DX is a novel and abstract construct. The concepts and definitions of it are multiple, and vary both in research and in industry. This means that the author's understanding of the topic improved and evolved during the study.

Easterbrook et al. (2008) define a set of guidelines for empirical research in software engineering. They argue that selecting a clear research question, an explicit philosophical stance, and an appropriate research method are key when researching in the context of Software Engineering (SE). They also note that many research projects in SE fail in defining the stance of the research, which then further complicates all aspects of the research including its interpretation, validity, replicability etc.

During writing of the thesis, the author was working at the case company, allowing to perform ethnographic techniques in understanding and studying the phenomena. Working at the case company also allowed to have discussion with colleagues.

3.1 Research approach

This thesis takes an exploratory approach to study the research problem and to answer the research questions. A research with an exploratory approach considers questions that address the existence of a phenomena, try to describe something (Easterbrook et al., 2008). The definition of DX given by Fagerholm and Münch (2013) will work as the basis of the study, but it is only giving the abstract concept to build upon. This concept and framework has not widely been taken into practice into empirical research.

Also, from initial discussion in the case company there was almost as many definitions of DX as there was discussions. This shows that depending on the role of the practitioner and the context they are working in affects in the definition of DX.

3.2 Research philosophy

According to Easterbrook et al. (2008), making explicit decisions on the research methods is key, and by doing that the research becomes more beneficial for practitioners and other researchers. Therefore they also argue that explicitly defining every step of the research should be done.

This thesis will take a **constructivism** view of the truth, combined with pragmatic approach. A constructivist approach sees the problem as that the human context is always present, and that it is an important part of the research and the study. The **pragmatic** approach is also required because of the case company in this study. The context of a company is important to include, as something that works with one company, might not work with another company. The pragmatic

approach can be seen as an approach from engineering, where the interest is focused on what works at a specific time in a specific context.

An **positivism** is not a suitable approach in this context. A positivistic approach tries to build up knowledge on verifiable observations that is then incrementally built upon. Positivists prefer to create specific theories from where testable hypotheses are extracted. These hypotheses are then tested in an controlled and isolated environment (Easterbrook et al., 2008).

3.3 Unit of analysis

Easterbrook et al. (2008) point out that it is important to pick the unit of analysis of the study. The unit in this case could be a company or organization, a project, a team, or an individual software developer. As DX is the experience of an individual developer, it is selected as the primary focus in this study.

Another viewpoint to this is also the case company, that has its own culture and set of practices. This study will also see the company as a unit of analysis. The company culture and practices are something that is shared between every developer and other employees.

3.4 Research method selection

To be able to investigate and define the definition of DX at the case company, the study has to focus on the foundations. During early stages of the thesis, it was noted that a comprehensive literature review and study was required. A good literature review gives a good foundation on which to build the research upon, and ensures that the studied field is understood correctly.

3.4.1 Multivocal Literature Review

Each research and study should have a background check and literature review where previous material and research is assessed, and from where the current research can be continued from an built upon.

A MLR is a form of systematic literature review, that produces both qualitative and quantitative data. In this study a MLR was seen as a good fit, as it allows to get a broad view of the current state of the research in the topic. It also allows to answer exploratory questions where the existence of a subject or topic is abstract or vague.

A more comprehensive discussion of the MLR is discussed in section 4.

3.4.2 Brainstorming

Brainstorming (BS) is a widely used technique, where the idea is to foster unique and novel ideas. Usually BS sessions are conducted when there is a need for innovation and new ideas. One typical use case of BS would be in a context of developing a product. With help of involving the development team in a BS session, new ideas and possible development directions can be found. The foundation of BS lies in that there are no right and wrong ideas, and that every idea is welcome.

In the context of this thesis, the technique of BS works as an exploratory way of initiating the study. Because of the novelty of the research topic DX, BS is an appropriate way of establishing the initial understanding of DX in the context of the study and the case company. Because the researcher has created their own understanding of what DX is, it would interfere with the study results if the researcher would participate into the sessions. It would also be problematic if the researcher would introduce the topic before the session, because the would either change or deepen the personal image of DX that the participants at that moment have. This is in line with the exploratory research philosophy described by Robinson et al. (2007). If the initial session would have been for example a workshop facilitated by the researcher, the researcher might have affected the session, and it would not have yielded in the same results as with a BS session.

Brainstorming types can be identified into *Traditional Brainstorming (TBS)*, *Nominal Brainstorming (NBS)*, *Electronic brainstorming (EBS)* (Al-Samarraie and Hurmuzan, 2018). *TBS* are verbal brainstorming sessions where the idea is to brainstorm verbally together in a group face-to-face. *NBS* can be seen as individual BS sessions, where each individual does the brainstorming without working together. *EBS* is a way to conduct BS with help of computers, from where each individual can concurrently input their ideas. Meanwhile the participants can see ideas from others. However there is a level of anonymity that defeats some of the problems with TBS and NBS. All in all, the different types of BS has been studied and their efficiency is dependent on many factors (Diehl and Stroebe, 1987), (Pinsonneault et al., 1999), (Faste et al., 2013).

There are some problems with brainstorming sessions (Pinsonneault et al., 1999) (Faste et al., 2013). *Evaluation apprehension* means that group members might get the feeling of being judged, and therefore they might feel reticent with bringing out their ideas. *Production blocking* might also be a problem, as working in groups will require everyone to talk on their own turns, which might lead to ideas becoming irrelevant, ideas might be forgot, the ideas might be rehearsed that blocks the generation of new ideas. *Free riding* can also happen when some BS group members rely on other members to perform the BS actions. According to Isaksen et al. (1998), the proponents of brainstorming is the practitioners, and the opponents are often researchers.

In this study the initial brainstorming session was selected to be a combination of TBS and NBS. NBS allows everyone to work individually. The case company has also established itself as working very agile, and therefore mentioned problems with BS do not require any extra actions to mitigate them. The BS session used Think-Pair-Share method (Lyman, 1987), where each participant of the session first thinks about the problem and possible ideas by themselves, then after that each individual is paired together where they share their own initial ideas, discard duplicate ideas, and further develop new ideas after the discussion. After pairing, each pair presents their ideas and findings to the whole group.

3.4.3 Interviews

Interviews can be conducted with the concept and framework of developer experience, even without having the participants being aware of the definition of Developer Experience. By using the conation, affection, and cognition, it is possible to initiate discussions of the different elements of DX.

3.5 Timeline

Mixed-methods approaches are more complex research strategies where multiple different approaches of an empirical study is combined (Easterbrook et al., 2008). In mixed-methods approaches the limitations of the different methods are acknowledged, and the limitations of them are mitigated.

Sequential exploratory strategy is one specific mixed-methods research approach. In this approach first qualitative data is collected and analyzed, after which the focus shifts towards quantitative data. Purpose of this approach is to first explore the topic with help of qualitative research, and then possibly test some emerging theory with help of quantitative research. Another reason might be to help in interpreting the qualitative data.

Concurrent triangulation strategy is another specific approach of mixed-methods (Easterbrook et al., 2008), where different research methods are used concurrently. The concurrent research enables to use results from one method to another, and vice versa. This enables to cross-validate research findings.

In this thesis both *sequential exploratory strategy* and *concurrent triangulation strategy* was used as research approaches. The sequential exploratory strategy was implemented by first conducting the initial multivocal and traditional literature reviews, and then only after that the ideation and planning of the workshops started. However at this point the MLR was not finalized, and still had some open questions, and ongoing interpretations and analysis.

The technique of concurrent triangulation strategy was utilized by conducting the MLR and the brainstorming concurrently. While the first iteration of the MLR was being performed, the initial workshop at the case software consultancy company was held. The idea behind a concurrent research was to get as comprehensive definition of the research problem as possible from as many different viewpoints and contexts as possible.

The initiation of the thesis started during the summer of 2019. The topic of the thesis was selected, and the initial and the preliminary research question and problem were defined. At that moment, the concept of DX was too vague to the author. After some discussions and studies on the topic, it was decided that the MLR should be taken as the first step of the study. When the MLR had been initiated the first results from it emerged, at the meantime the context of the empirical study started also to get defined.

3.6 Analysis of data

Renner and Taylor-Powell (2003) give a set of guidelines and practices on analysing

qualitative data. They state that using qualitative data in research requires discipline, creativity, but also a systematic approach. The divide the analysis of qualitative data into 5 different step: get to know your data, focus the analysis, categorize information, identify patterns and connections within and between categories, and finllay interpretation. Categorization of the data is explained more in the specific sections where categorization, theming, or coding of data is performed.

4 Multivocal literature review

Traditionally, in SLRs the reviewed literature consists only of literature that is formally published, and thats motivation of publishing is the publication in itself, e.g. publications in journals and conferences. Material that is produced with commercial interests and informally published material and publications are not considered in SLR (Garousi et al., 2019).

MLRs, are a way to include grey literature into SLRs (Garousi et al., 2016). Grey literature can be defined in different ways, and research fields define grey literature in ways that are meaningful to that specific field.

> *"Grey literature stands for manifold document types produced on all levels of government, academics, business and industry in print and electronic formats that are protected by intellectual property rights, of sufficient quality to be collected and preserved by library holdings or institutional repositories, but not controlled by commercial publishers i.e., where publishing is not the primary activity of the producing body."* (Schöpfel, 2010)

The Prague Definition of grey literature is strict and therefore not allowing e.g. blog posts to be used on MLRs. However, a specific guideline for including grey literature in literature reviews has been created (Garousi et al., 2019). This guideline is based on the guidelines on how to perform SLR in SE (Kitchenham and Charters, 2007).

4.1 The motivation behind a MLR

DX is a complicated subject and topic, and a clear and well defined definition of it does not exist at the moment of writing (August 2019). There is a need to create an understanding of the definition of DX and a basis to build the rest of the thesis upon. Normal literature reviews can help in these cases, and they create a common understanding of the topic that is going to be discussed. However, normal literature reviews are prone to be biased. To avoid bias of the author, and because DX is a subjective concept of the developer, the definition of DX can be reviewed with a help of a Systematic Literature Review (SLR). Systematic literature reviews are a way of producing evidence based results, and they are effective in complex and opinion based fields where a common agreement of a concept or topic might be difficult to find.

In software engineering practitioners constantly produce valuable literature in e.g. technical reports or blog posts, but this material is not considered in SLRs. This has been identified as a problem, and there's been a call for MLRs in SE (Garousi et al., 2016).

An SLR would include only the scientific papers, and therefore it might not be sufficient to only focus on that. In a MLR the GL should provide a current perspective and fill in the gaps of scientific and formal literature (Garousi et al., 2019).

SE practitioners are producing a lot of literature, that would not be considered in normal literature reviews or SLR. This GL can provide insights about the field of SE, and especially about DX.

A SLR has been conducted on the concept *Programmer Experience* (Morales et al., 2019). This SLR will be used to guide this MLR.

4.2 The review protocol of the MLR

DX is a novel concept, and therefore there has been little formal research on the topic. Based on the different levels of literature, white, grey, and black (Garousi et al., 2019) (find correct reference), DX could be seen even to be in the category of black literature.

All data of the MLR can be found **here**. The data collection is done with Google Sheets and is based on the example shown in (Garousi et al., 2019).

4.2.1 Research questions

The foundation for the MLR is the first research question RQ1 (What is the definition and aspects of Developer Experience, and how do they differ between scientific literature and literature written by practitioners?). The goal is to create a definition of *Developer Experience* with help of both white literature and grey literature.

4.2.2 Search process

The search is performed as a manual search by the author from various libraries to gather the academic literature. For grey literature, the Google search engine (https://www.google.com) will be used. To limit the amount of grey literature, only the first 2 pages of searches from google.com will be included. Only 91 percent of participants in a study went to the second page of google results page (van Deursen and van Dijk, 2009). Therefore it's appropriate to include only the most relevant results from the google search.

IEEExplore	(https://ieeexplore.ieee.org/Xplore/home.jsp)
ACM	(https://dl.acm.org/)
ScienceDirect	(https://www.sciencedirect.com/)
Scopus	(https://www.scopus.com/search/form.uri?display=basic)

Table 2: Scientific literature sources for the MLR

Other possible libraries and sources for resources could have been the following sources:

- Google Scholar (https://scholar.google.com)
- Citeseer (https://citeseerx.ist.psu.edu/index)
- and SpringerLink (https://link.springer.com/)

However, they do not provide advanced search with the author keyword as a search criteria.

Search string for both scientific and grey literature will be **"developer experience"**. Because of an ambiguous definition of DX, only one search string is uses. This assures that all relevant publications are included. Including more words in the search string or creating a more complex search string would require a better understanding of DX, that the author does not have at this moment. Also, including other search strings would bias the search result.

At the moment of writing (August 2019), using "developer experience" as search string produces 3410 results on Google Scholar. On the first round of searching from Google Scholar, the first search with keyword **"developer experience"** resulted in 2 included papers and 8 excluded. The search keyword needs to be adjusted.

To further narrow down the search, the search was modified to include only results where author keyword was "developer experience". This narrowed down the search significantly, and removed irrelevant results.

Using the author keyword gives results where the author is intentionally discussing the topic. In the case of DX, with searching only with the author keyword, the inclusion/exclusion rate is significantly better than with a full-text search. However, this approach will remove the possibility to discover definitions of DX where the author is not aware of this concept or phenomenon.

4.2.3 Inclusion criteria

The material must be in English. Articles that show up in the searches, and that have the words **developer experience** in consecutive order order are included in the review.

4.2.4 Exclusion criteria

Papers that discuss about developer's experience level e.g. *senior* or *junior developer*, will be excluded.

4.2.5 Quality assessment

Because of the novel topic the quality assessments might not be that crucial in this study.

4.2.6 Data collection and analysis

All papers will be collected into one form with the following data points:

- The source
- Year of publication
- Classification of paper
  - Type of research

  - Scope (Research trends or specific research question)
- Main software engineering topic area
- The author(s) and affiliation (organisation and country)
- Research question/issue
- Definition of Developer Experience
- Point of interest towards Developer Experience (context)
- Summary of paper

Analysis will be done on the found definitions and concepts of Developer Experience. During the study the collected data points and especially the about developer experience will be refined.

4.3 Data Collection

During collection of data different aspects emerged. The process of collection was a continuous refinement of the search method, inclusion and exclusion criteria, data extraction points. During the collection the comprehension and understanding of the base concept of DX was continuously refined.

4.4 Data Analysis

The following sections go trough the data points that emerged during the collection of the data

4.4.1 Definition of DX

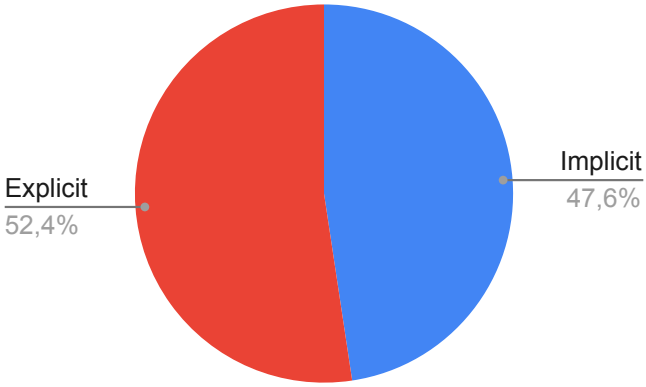
The most common definition of DX in scientific papers is the concept and definition of in (Fagerholm and Münch, 2013). This definition is at the moment the only stated definition of DX in the formal literature. There is also further derivations on this concept and definition, but nothing that is revolutionizing or that takes another viewpoint.

To analyze how the the definition of DX is given, all articles and papers will be grouped into either having **implicit** or **explicit** definition of DX. This division was something that did emerge in the collection and analysis of the material.

Explicit definition of DX means that the author has in some words explained or defined the concept of DX, or referenced some other material that gives the definition to it.

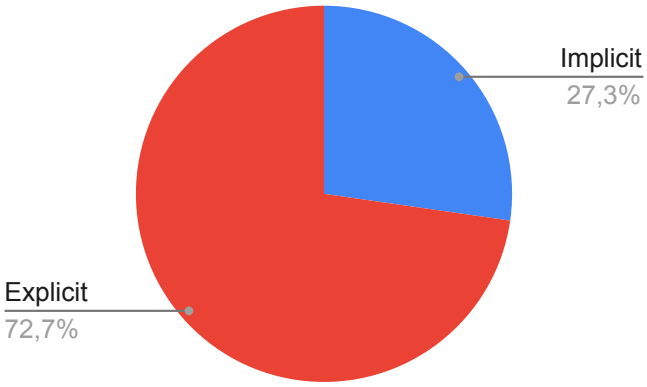
Implicit definition of DX means that the author doesn't give any definition or explanation of what DX is. The definition can often however be inferred from the context of the paper.

Figure 3: Implicit vs. explicit definition of Developer Experience in scientific literature



Scientific papers are almost equal in both giving an implicit or explicit definition of DX. It shows and confirms that DX is an topic and research area that has not gotten much attention at the moment of writing. Some research areas have been developed so far that there is things that can be taken for granted. However, the are of DX might not yet be at that point yet and therefore

Figure 4: Implicit vs. explicit definition of Developer Experience in grey literature



Grey literature has a majority of explicit definition of DX. The articles are often starting with giving an explanation of their viewpoint and definition of DX. Quickly the articles then continue to discuss the topic from the selected context and viewpoint.

4.4.2 Context of DX

From the analysis of the material, there is a clear indication that there are different viewpoints to DX. The different context that emerged from the analysis are listed in table 3.

Definition
Development Environment
API
Product or service
User Experience
Team, Collaboration, & Community
Mood & Feelings

Table 3: Different contexts that emerged during the literature review

Grey literature takes to a large degree a viewpoint where DX is a form of UX, where developers are users of products and services. In this viewpoint the DX consists of features that are also used when measuring the UX of a service. These include factors like functionality, usability, and reliability.

The grey literature is also heavily influenced by businesses marketing their services or products. To gain visibility and recognition, they are publishing articles and posts on their blogs to write and discuss a specific topic. These businesses are defining DX from their own point of view where they are providing products and services, that are directly used by developers.

Some articles (which?) mentioned that back in the days, it was executives that made the business and purchase decisions of tools, frameworks and other products and the developer's opinion were not considered. Developers were forced to use whatever they were offered.

Today, the purchase decision has more and more shifted to be a responsibility of the developer. Developers are the final users of the product and therefore businesses have probably realized that developers are the ones to make the decisions. All in all, it can be seen from the current grey literature that developers are being considered more and more (Devs are people too), and that this movement has created the concept of DX.

DX allows developers to reason about things that before has been difficult. Making statements that are in the favour of developers might have been difficult as there hasn't been any term to coin the feelings, emotions, needs,

Businesses have taken notice on this movement, and are now utilizing it to create products and services.

DX can be seen that there is always a developer that is a user. The role of the user is the variable, and can vary from being a user of a product where the DX is seen in the product, or then the user can be a user of a developer workflow in a software project.

In many of the grey literature articles the authors have their own view and definition of what DX is. Only in few articles there is actual questioning of the definition of DX.

The formal research on DX has taken a step further and is also considering the social aspects of software development.

A research group in Brazil has to a large extent researched the DX in the context of Mobile Software Ecosystems (MSECO). To these ecosystems belong mobile application development platforms as Android and iOS. Their approach to DX can however be seen as something applicable to all kinds of products and services that aim to create a better DX and improve on it.

Another group of researchers in Finland have studied the mood of developers and its effects at varying levels of software development.

Overall the amount of formal research on DX is lacking. The lack of definitions and the amount of search results in the search speaks for this.

Team, community, and collaboration.

Many articles use the keyword developer experience but only mention DX briefly in their material. This forces the readers to

Definition	5
Development Environment	11
API	10
Product or service	9
User Experience	14
Team, Collaboration, & Community	16
Mood & Feelings	8

Table 4: Scientific and grey literature combined

Definition	4
Development Environment	9
API	4
Product or service	2
User Experience	7
Team, Collaboration, & Community	11
Mood & Feelings	7

Table 5: Scientific literature

Definition	1
Development Environment	2
API	6
Product or service	7
User Experience	7
Team, Collaboration, & Community	5
Mood & Feelings	1

Table 6: Grey literature

Figure 5: Combined

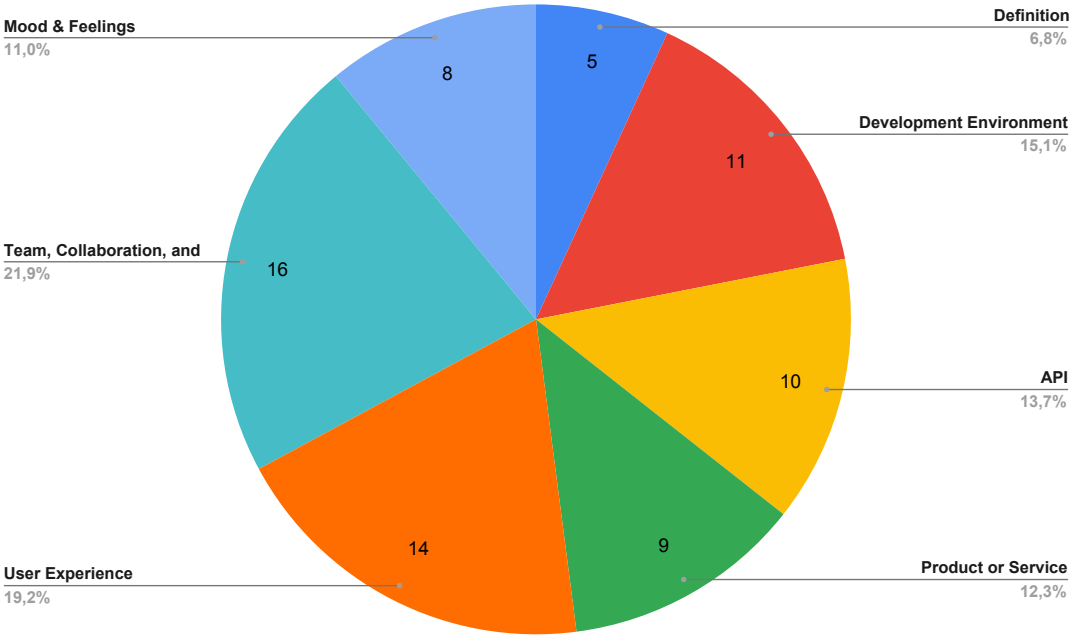


Figure 6: Scientific

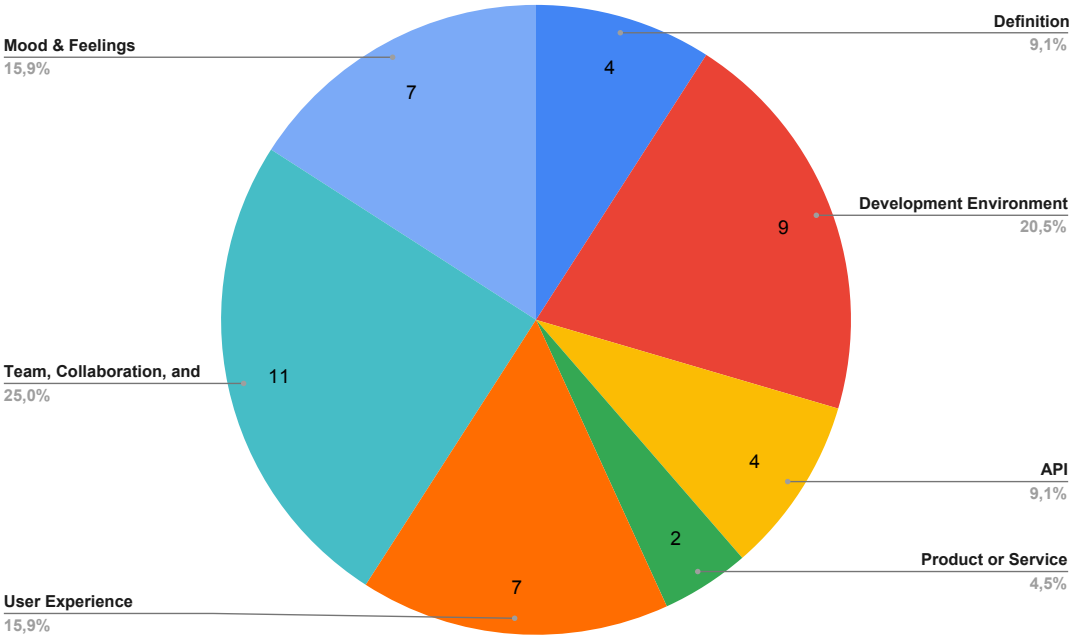
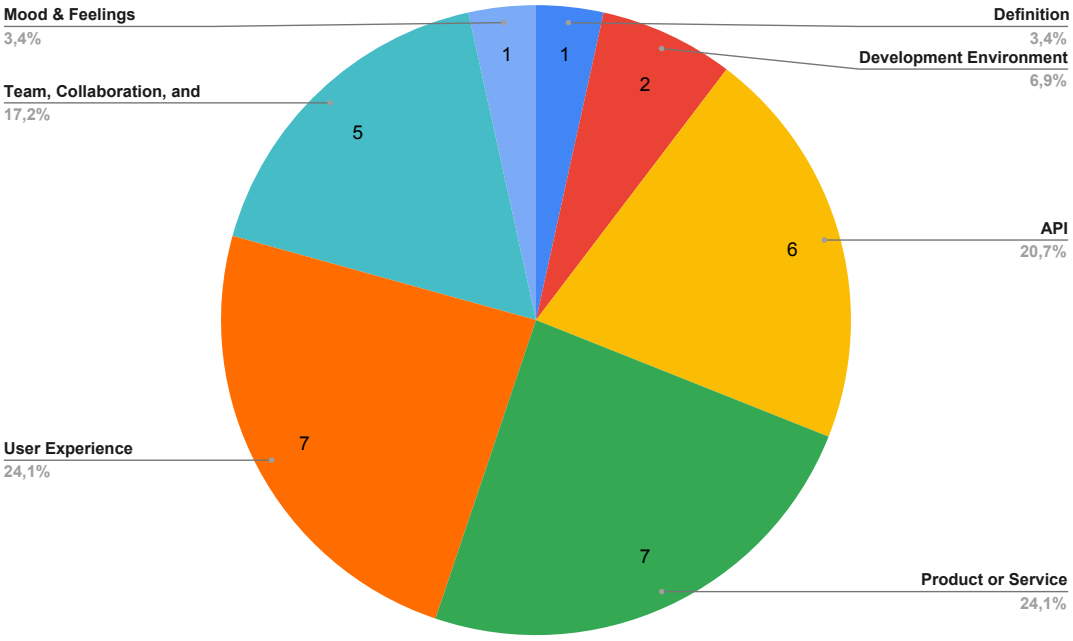


Figure 7: Grey



4.5 Validity of search results

The search engine Google is known to provide results based on many different variables on the user e.g. previous searchers, internet profile etc. Therefore the search results from Google might not present results that are applicable for anyone. To mitigate this, private sessions were used when performing the searches.

5 Current state analysis

6 Exploratory research

Based on the results from the background section and the Multivocal Literature Review, it can be derived that the research topic is complex and difficult to define. Therefore, to gain a better understanding and grasp how DX is thought of in the industry, a workshop and a series of interviews will be conducted.

Possible approaches for the practical part of the study:

- Based on the this article, create a set of guidelines and good practices of DX, and interview and/or measure how well established the application of the practices are
- Conduct interviews studying the development environments of different projects. *"When starting a new project, be upfront that you want to tailor your tools and methods for collaboration"*(uxdesigndx)
- Workshop to map current practices and viewpoints on DX
- Interviews with individual developers on their viewpoint of DX

6.1 Workshop

6.2 Survey (or interviews)

7 Results

Answer the research questions and problem.

7.1 Validity of results

Tässä osassa on syytä myös arvioida tutkimustulosten luotettavuutta. Jos tutkimustulosten merkitystä arvioidaan »Tarkastelu»-osassa, voi luotettavuuden arviointi olla myös siellä.

8 Summary

9 Conclusions

References

Al-Samarraie, H. and Hurmuzan, S. (2018). A review of brainstorming techniques in higher education. *Thinking Skills and Creativity*, 27:78 – 91.

Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies*, 58(2):207 – 214.

Capretz, L. F. (2014). Bringing the human factor to software engineering. *IEEE software*, 31(2):104–104.

DeMarco, T. and Lister, T. (2013). *Peopleware: productive projects and teams*. Addison-Wesley.

Diehl, M. and Stroebe, W. (1987). Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of Personality and Social Psychology*, 53:497–509.

Easterbrook, S., Singer, J., Storey, M.-A., and Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer.

Fagerholm, F. (2015). *Software Developer Experience: Case Studies in Lean-Agile and Open Source Environments*. PhD thesis, University of Helsinki.

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., and Abrahamsson, P. (2014). How do software developers experience team performance in lean and agile environments? In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 7:1–7:10, New York, NY, USA. ACM.

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., and Abrahamsson, P. (2015). Performance alignment work: How software developers experience the continuous adaptation of team performance in lean and agile environments. *Information and Software Technology*, 64:132–147.

Fagerholm, F. and Münch, J. (2013). Developer experience: Concept and definition. *CoRR*, abs/1312.1452.

Faste, H., Rachmel, N., Essary, R., and Sheehan, E. (2013). Brainstorm, chainstorm, cheatstorm, tweetstorm: new ideation strategies for distributed hci design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1343–1352. ACM.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2016). The need for multivocal literature reviews in software engineering: Complementing systematic literature reviews with grey literature. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, EASE '16, pages 26:1–26:6, New York, NY, USA. ACM.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101 – 121.

Graziotin, D., Fagerholm, F., Wang, X., and Abrahamsson, P. (2017a). Consequences of unhappiness while developing software. In *Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering*, pages 42–47. IEEE Press.

Graziotin, D., Fagerholm, F., Wang, X., and Abrahamsson, P. (2017b). On the unhappiness of software developers. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 324–333. ACM.

Graziotin, D., Fagerholm, F., Wang, X., and Abrahamsson, P. (2017c). Unhappy developers: Bad for themselves, bad for process, and bad for software product. *CoRR*, abs/1701.02952.

Graziotin, D., Fagerholm, F., Wang, X., and Abrahamsson, P. (2018). What happens when software developers are (un)happy. *Journal of Systems and Software*, 140:32–47.

Isaksen, S. G. et al. (1998). *A review of brainstorming research: Six critical issues for inquiry*. Creative Research Unit, Creative Problem Solving Group-Buffalo Buffalo, NY.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kuusinen, K. (2015). Software developers as users: Developer experience of a cross-platform integrated development environment. In Abrahamsson, P., Corral, L., Oivo, M., and Russo, B., editors, *Product-Focused Software Process Improvement*, pages 546–552, Cham. Springer International Publishing.

Kuusinen, K., Petrie, H., Fagerholm, F., and Mikkonen, T. (2016). Flow, intrinsic motivation, and developer experience in software engineering. In Sharp, H. and Hall, T., editors, *Agile Processes, in Software Engineering, and Extreme Programming*, volume 251. XP 2016, Springer, Cham.

Lyman, F. (1987). Think-pair-share: An expanding teaching technique. *Maa-Cie Cooperative News*, 1(1):1–2.

Mäenpää, H., Fagerholm, F., Munezero, M., Kilamo, T., Mikkonen, T. J., et al. (2017). Entering an ecosystem: The hybrid oss landscape from a developer perspective. In *CEUR Workshop Proceedings*.

Morales, J., Rusu, C., Botella, F., and Quinones, D. (2019). Programmer experience: A systematic literature review. *IEEE Access*, PP:1–1.

Murphy, L., Kery, M. B., Alliyu, O., Macvean, A., and Myers, B. A. (2018). Api designers in the field: Design practices and challenges for creating usable apis. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 249–258. IEEE.

Palviainen, J., Kilamo, T., Koskinen, J., Lautamäki, J., Mikkonen, T., and Nieminen, A. (2015). Design framework enhancing developer experience in collaborative coding environment. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 149–156, New York, NY, USA. ACM.

Pinsonneault, A., Barki, H., Gallupe, R. B., and Hoppen, N. (1999). Electronic brainstorming: The illusion of productivity. *Information Systems Research*, 10(2):110–133.

Renner, M. and Taylor-Powell, E. (2003). Analyzing qualitative data. *Programme Development & Evaluation, University of Wisconsin-Extension Cooperative Extension*, pages 1–10.

Robinson, H., Segal, J., and Sharp, H. (2007). Ethnographically-informed empirical studies of software practice. *Information and Software Technology*, 49(6):540–551.

Schöpfel, J. (2010). Towards a Prague Definition of Grey Literature. In *Twelfth International Conference on Grey Literature: Transparency in Grey Literature. Grey Tech Approaches to High Tech Issues. Prague, 6-7 December 2010*, pages 11–26, Czech Republic.

van Deursen, A. J. and van Dijk, J. A. (2009). Using the Internet: Skill related problems in users' online behavior. *Interacting with Computers*, 21(5-6):393–402.