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** 54+4 **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** 54+4 **Språk** Engelska

Sammandrag
Sammandrag på svenska

Nyckelord Nyckelord på svenska, Utvecklar upplevelse

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
SEO	Search Engine Optimization
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 & 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 or services that aid them in their creative design work, will result in an User Experience (UX). Human Computer Interaction (HCI), a traditional field of research, studies the interface and interaction between computers and humans. 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 (Hassenzahl, 2003).

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, service or product, DX can be seen as the experience of developers developing software in a complex social and technical context. In the case of DX the context 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 also 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.

Lately there has been mentions in the software engineering industry of DX like *"I love using this framework as it has such a good developer experience!"*, and *"The conventions of the project were confused and caused a really bad developer experience"*. This study aims to build on the understanding of what phrases and opinions like these mean. Focus of this study is on understanding the definition of DX on a broad level 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 & Münch, 2013), in some other research the term *Programmer eXperience* and abbreviation *PX* is used (Morales, Rusu, Botella, & Quinones, 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.

Law, Roto, Hassenzahl, Vermeeren, and Kort (2009) conducted a comprehensive research on the notion of UX. UX is significantly more mature than DX, but still there are problems of communication of UX and misunderstandings of what the notion of UX is. Therefore they saw the need of performing their study. This is a clear indicator that DX is also in the need of a clear and well defined.

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.

Ozkaya (2019) talk about *The Voice of the Developer*, and how the focus of software development has been on the customer's and product's perspective on e.g. code quality and technical debt. The voice of the developer considers more on the developer's perspective, and on how the product or service under development resonates with the developer's satisfaction and well-being. The voice of the developer resonates with DX, and there can be seen a lot of commonalities with what is considered with them.

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, Singer, Storey, and Damian (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?

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 1.1	What objects/entities have been studied with respect to developer experience?
	RQ 1.2	What methods have been used to study developer experience?
	RQ 1.3	What are the main results of the existing research on developer experience?
	RQ 1.4	What is known about factors that improve or worsen developer experience?
RQ 2		**How is developer experience and its aspects defined by different roles (or software developers) in a software consultancy company?**
	RQ 2.1	What different experience objects of Developer Experience are there in the software consultancy company?
	RQ 2.2	What factors related to the experience object improves or worsens the developer dxperience?
	RQ 2.3	How the developer experience of the experience objects be improved?

Table 1: The research questions

To analyze the research questions, the categorization, classification, and guidelines of Easterbrook et al. (2008) is 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 out the absence of definitions.

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.

Both 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 selected research questions will guide the research and guide with selecting the used methods and techniques.

1.3 Scope and focus

The scope of the thesis changed remarkably during its conviction and during the time it was worked on. Initially the philosophy of the thesis was more of a positivistic approach, but the more information was gathered about the topic, the more exploratory the nature of the thesis became. More of this in section 3.2.

The scope of this thesis is narrow. Even if the there is a lot of open questions about the concept of DX, the thesis is only scratching the surface of the current state and understanding of the definition of DX. The thesis focuses on getting a broad view on the research problem. It is acknowledged that the problem is also most likely not solvable, and that there are no definitive answers to the research questions.

The empirical study is focused only on one case company, where the aim is to understand the current notion of DX in the context of the company. Initially the idea was to generate a set of practices or even processes that would improve the ways of working at the company. However this was noted to be too ambitious.

This thesis doesn't give any clear and well defined definition of DX, and there is not any "right" or "wrong" answers presented in the thesis.

1.4 Structure of the thesis

1. Introduction
2. ???
3. 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 Experience

Moilanen, Niinioja, Seppänen, and Honkanen (2018, p. 167) differentiates UX from DX by stating that in UX the focus is on using the product and in DX the focus is on creating a product. They also note that UX is a user-centered model of operation, whereas DX is a process-product-centered model of operation. This differentiation of using versus producing is used in this thesis

The notion of experiencing can been seen in various different ways in SE. Sometimes the different variations of experiences are used intertwined, but is important to differentiate them to avoid misunderstandings. Different experiences related to consuming, using or interacting with something provided by someone else includes *user experience*, *product experience*, *brand experience*, and *service experience*. Experiences related*programmer experience*, and finally also *developer experience*.

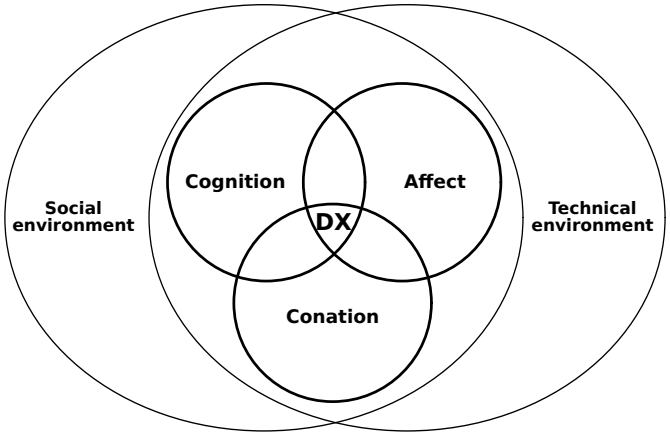
Law et al. (2009) investigate into the notion of UX and simultaneously differentiates the different kinds of usage experiences. *Brand experience* includes interaction not only with the product, but also the company and it's services and is therefore a broader concept than UX. *Product experience* is narrower the user experience and deals with the objects that are products. *Service experience* concerns about face-to-face services and experiences related to that. *User experience* is discussed in section 2.1.3

2.1.1 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).

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



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.

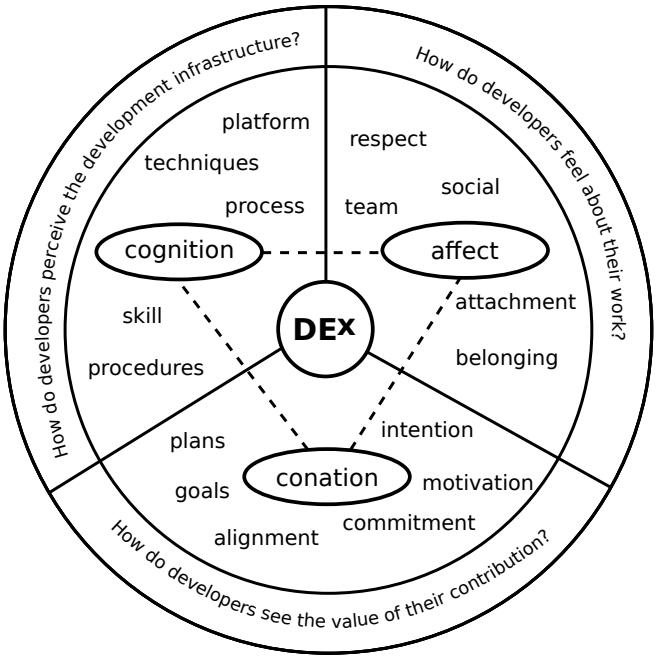
DX can be seen as important from multiple different viewpoints. From a viewpoint of project management a better DX can help to understand, evaluate, and plan projects so that they are inline with the three different dimensions of DX. Another viewpoint is when designing a software development platform 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).

2.1.2 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

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



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).

2.1.3 User Experience

User Experience (UX) emerged from the traditional field of HCI. It was seen that only focusing on the instrumental value of the product or service is not enough. Hassenzahl and Tractinsky (2006) studied recent research of UX, and also proposed the future of it to aid in finding a direction for research of it. They found 3 important perspectives of UX. 1) UX is beyond the instrumental by considering the holistic, aesthetic, and hedonic aspects of usage. 2) Emotion and affect are to interest when understanding the subjective view on usage, that often is focused on the positive antecedents and consequences. 3) The experience in itself is a complex and dynamic phenomena that is near to impossible to replicate.

Law et al. (2009) found that the notion of User Experience (UX) has been widely adopted, without having a clear understanding and definition of what UX. Interestingly,this relates exactly to the situation that DX is in now.

2.1.4 User Starting Experience

Murphy, Kery, Alliyu, Macvean, and Myers (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. Adopting 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.

2.2 Selection of tools

Perceived choice is a perception of that the choice has already been made (Kuusinen, Petrie, Fagerholm, & Mikkonen, 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 (Kati 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 (Kati 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.

2.3 Organization and project onboarding

Mäenpää, Fagerholm, Munezero, Kilamo, Mikkonen, 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.

2.4 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).

2.5 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.

2.6 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, Fagerholm, Wang, and Abrahamsson (2017c), Graziotin, Fagerholm, Wang, and Abrahamsson (2017b), Graziotin, Fagerholm, Wang, and Abrahamsson (2017a), and Graziotin, Fagerholm, Wang, and Abrahamsson (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 & Münch, 2013) includes the affective dimension that is directly related to the happiness of developers.

2.7 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.

2.8 Application Programming Interfaces

Application Programming Interfaces (APIs) are interface that developers use to communicate and transfer information with external service, but also to build products and services with. As developers are in the role of using APIs, the APIs UX, and also DX, has been researched (Murphy et al., 2018). Developers are the main users of APIs, the API's DX has been considered as a very important aspect of the API. A good DX aids in getting users for an API.

APIs, and especially Web APIs have enabled to create businesses around APIs (Evans & Basole, 2016), (Tan, Fan, Ghoneim, Hossain, & Dustdar, 2016), (Moilanen et al., 2018). This type of business is called an API economy, and has quickly become a viable way of generating revenue for businesses.

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) works as the basis of the study, but it is only giving the abstract concept to build upon. This concept and framework has not yet been widely taken into practice in 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.

In Morse et al. (2016) Juliet Corbin reflects on the grounded theory research methodology and how she approached a research she conducted in combination while writing a book (Corbin, Strauss, & Strauss, 2015) about grounded theory.

> *"I was just going to sit down with that first piece of data in front of me and let it flow, let the research take me where it wanted"* - Juliet Corbin (Morse et al., 2016, p. 43)

Taking an approach that enabled and open mind and building the research questions was the most suitable in this thesis, both from the viewpoint of the research topic and researchers own skills, competency, and way of thought.

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 and 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, Segal, and Sharp (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 & 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 & Stroebe, 1987), (Pinsonneault, Barki, Gallupe, & Hoppen, 1999), (Faste, Rachmel, Essary, & Sheehan, 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 finally 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 Systematic Literature Reviews (SLR) the reviewed literature consists only of literature that is formally published, and of which the motivation of publishing is the publication in itself, e.g. scientific publications in journals and conferences. Material that is produced with commercial interests and informally published material and publications are not considered in SLR (Garousi, Felderer, & Mäntylä, 2019).

Multivocal Literature Reviews (MLR), are a way to include grey literature into SLRs (Garousi, Felderer, & Mäntylä, 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 & Charters, 2007).

4.1 The motivation behind a Multivocal Literature Review

DX is an abstract concept and framework. As pointed by Fagerholm (2015), the framework they present can be used to guide inquiries into DX. There was a need to create an understanding of the definition of DX from the point of view in this thesis. Traditional literature reviews can help in these cases, and they create a common understanding and basis of the topic that is going to be discussed. However, traditional 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 could have been reviewed with a help of a SLR. SLRs 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.

4.2 The review protocol of the Multivocal Literature Review

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), DX could be seen even to be in the category of black literature. DX can still be seen to be on the level of ideas, concepts, and thoughts.

All data of the MLR can be found the following link: https://bit.ly/31QzQ7z. 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 understanding of the current situation of the definition of *Developer Experience* with help of both white literature and grey literature.

4.2.2 Search process

The search process was divided into a database search and a snowballing procedure performed on the first initial set of the results. The following sections motivate the decisions made before and during the search process, and explain the details on how the search was conducted.

4.2.2.1 Database search

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

Other possible libraries and databases that could have been used are listed in table 3. However, the sources listed in this table does not provide advanced search. Advanced search by author keyword was a search criteria and because of that these libraries and databases were excluded from the initial search of sources.

Wohlin (2014) state that with manual searches of sources, the results will vary and are hard to replicate because of the lack of a system. This is key when performing

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: Databases used for the MLR

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

Table 3: Databases excluded from the MLR

systematic reviews so other researchers can replicate the study. Wohlin (2014) also mention that performing database searches might result in a big set of irrelevant results, from where it can be difficult to find the relevant result from. The process of manually filtering out results is also error prone.

Search string for both scientific and grey literature was **"developer experience"**. The aim of the MLR review was to get an overview of the definition of DX, only that specific search string was used. That assured that all relevant publications were included. Including more words in the search string or creating a more complex search string would have required a better understanding of DX. Also, including other search strings would have biased the search result. 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. Wohlin (2014) mention that creating complex queries and search strings might be difficult because of the lacking standardized terminology of the research topic. This was exactly the case with this study, as the topic of DX is relatively novel and some research might be studying DX without explicitly stating or even being aware of that themselves.

Using the author keyword gave results where the author is intentionally discussing the topic of DX. 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 removed the possibility to discover definitions of DX where the author is not aware of this concept or phenomenon.

Law et al. (2009) performed a search on the definition of UX. Their search strings used a combination of keywords. However, trying to replicate these for the search of the definition of DX did not yield any satisfactory results, and either the number of search results was too big or too small.

4.2.2.2 Snowballing

The result of the database search gave good results that however were too narrow. With the guidance of the thesis supervisor who has done extensive research on DX, the decision of using snowballing to collect more sources was done. Snowballing allows to find sources that would otherwise go undiscovered. For example obscure journals, conferences, or magazines can be difficult to find with only using database searchers (Wohlin, 2014).

In this thesis one iteration of snowballing was performed on the scientific articles. The first set of scientific literature included 21 articles. Snowballing was performed on these 21 articles, and the first round of backward and forward snowballing resulted in 5 articles from backward and 14 articles from forward snowballing. Snowballing on

Snowballing was not used on grey literature. However, 2 different searches with the same keyword was performed in Google Search. The first round of grey literature about the topic was gathered at September 2019, and the second round in November 2019. Having two different occasions of gathering grey literature sources did take into action timely changes in search results of Google.

4.2.3 Inclusion criteria

The included material had to be written in English. Articles that showed up in the searches, and that had the words **developer experience** in consecutive order, were included in the review. This inclusion criteria of having the term "developer experience" was applied both the database search and the snowballing technique.

The selected sources had to discuss about DX explicitly mostly by using the term as it is. Without this criteria it would have been a complex job to draw a line between inclusion and exclusion, as basically every study concerning software engineering to some degree is more or less related to topic of DX.

4.2.4 Exclusion criteria

Papers that discussed about developer's experience in terms of the experience level were excluded from the review. These included papers where developers were compared on the experience level e.g. *senior* or *junior developer*. Also, where experience is defined as number of years working in software development or the number of commits in a software commit were excluded.

4.2.5 Quality assessment

Garousi et al. (2019) give guidelines on how to assess the quality of the sources. This can be done with help of quality points based on some set of questions or points of measure. This is useful when the aim is to achieve a certain level on the sources. It also helps to filter out a big number of possible sources. Because of the novel topic the quality assessments were not considered in this study. The amount of sources were low and because of the exploratory nature of the study, every possible and available source were considered.

4.2.6 Data collection

All sources of the review were collected into one form with the selected data points. The process of collection was a continuous refinement of the search method, inclusion and exclusion criteria, and data extraction points. During the collection the comprehension and understanding of the base concept of DX was continuously refined. The author's understanding about the topic improved, and therefore there was also need to continuously update and refine the data collection form.

First version of the data collection form was simple and included only some basic characteristics and data points of the sources. With help of ongoing collection, refinement of the search process, and adjustments to the research questions the data points emerged to their final form. A more comprehensive explanation of the collected data points is given in section 6.1.

4.2.7 Data analysis

Data collection was rather successful and multiple different data points was extracted. These data points were both qualitative and quantitative in nature. During the data

collection, the view of the data collection was focused on a horizontal perspective i.e. on source at a time. During analysis the view was flipped to a vertical view i.e. looking at one data point at a time.

The quantitative data analysis was performed with simple aggregations and visualized with help of charts. No specific methods or techniques of quantitative data were used. Iterating on the categories by removing, adding, and combining them, new categories and patterns emerged.

Qualitative data analysis was performed one data point at at a time. Each data point was copied to an excel table where electronic affinity diagramming was performed. Affinity diagramming or the KJ method (Scupin, 1997) is a technique that allows to make sense of qualitative data. The KJ methods allows to understand and interpret data that would otherwise be difficult to make something out of.

The idea is to shuffle statements that are written cards or notes. Next step is to group things that relate to each other together so that "teams" of cards are formed. Then each of these teams are titled, and finally the relations between teams are explained and reported. This method allows to create new interpretations and reveal hidden meaning behind the collected data. This method was suitable for this MLR, as the majority of the data collected was qualitative e.g. statements and loose interpretations from the articles.

5 Current state analysis

5.1 Case company

Reaktor is a Finnish company that provides consultancy and agency services. The company was founded in 2000 and has since then been a leading IT-consultancy company in the Finnish market. In addition to offices in Finland, Reaktor has offices in New York, Amsterdam, Tokyo, and Dubai (Reaktor, 2019b). Reaktor has won the Great Place to Work award multiple times in Finland (Info, 2011) and the company is actively improving the wellbeing of everyone at the company (Talouselämä, 2019).

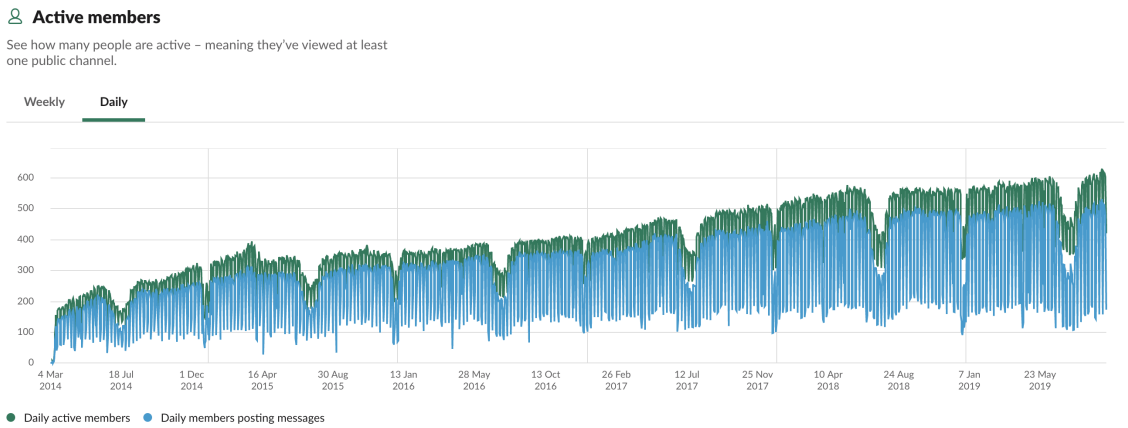
A significant part of Reaktor's employee's are technical experts, engineers, and developers. These roles include titles as Senior Software Engineer, Full Stack Engineer, Experience mobile developer (iOS/Android), and DevOps engineer (Reaktor, 2019a). This makes studying DX at Reaktor an interesting subject of research. Different roles and specialities have their own viewpoints of DX.

5.2 Developer Experience at Reaktor

Developer Experience has not been previously explicitly studied at Reaktor. However, during the initial phases of the thesis, the topic of DX in this thesis caused some interesting informal discussions about some problems and occurrences of DX. These discussions varied in length and depth, but allowed to create a good basis for the empirical study at the company.

Discussion channels used at Reaktor are very active, and information sharing is a constant activity. Vilkko (2016) took a deep dive into the ways of of working at Reaktor, and specifically into how agile methods form the experience of work engagement. They also mention that information sharing and knowledge management is something that gets a lot of care at Reaktor.

Figure 3: Daily active Slack members at Reaktor



By analyzing the company's Slack channel visualized in figure 3, it can be seen that people in the company are very active. However, by performing a quick search on the history of the chat's with the term "Developer Experience", only 62 results were found. In comparison 441 results was found when searching with the term "User Experience". However, it is worth to mention that a big chunk of the discussion is in Finnish.

6 Results

This section reports the different result of the research.

6.1 Results of the Multivocal Literature Review

The Multivocal Literature Review (MLR) performed on the definition of DX resulted in interesting results that presented and reviewed in the following sections. The data points collected emerged during the analysis, and the more analysis was done the more interesting revelations were found.

There are two different research groups that research DX. 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. To this group is also linked other studies related to the DX of IDEs.

	Scientific literature	**Grey literature**	
Included	39	19	58
Excluded	50	2	52
	89	21	110

Table 4: Number of sources analyzed for the MLR

6.1.1 The object / entity of DX under study

Based on the DX framework presented by Fagerholm (2015), a data collection point was included to understand the object or entity under study of DX. This addresses the first research question, specifically RQ1.1. The experience object is what is experienced in itself. This could be some artifact e.g. the IDE in the developer's development environment. However, it can also be something more abstract like the conventions or unwritten rules that a software team has established.

6.1.1.1 Object under study in scientific literature

In scientific literature the most prominent object under study was the *usability* of various objects e.g. the IDE and API. This stems most likely from the fact that DX is seen as a variation or extension of UX. In many articles the definition of developer experience is related to or derived from UX. Usability of developer tools, programming languages, and APIs are concrete, important, and measurable examples of objects of DX. DX is however not limited to UX of tools and services.

Other emerged objects of DX under study in scientific literature included *the support for developers, technical environment, developer's activities in a community, wellbeing of developers, and individual developers in a bigger context.*

Support for developers was addressed by discussing about frameworks that support developers in their daily development activities. This includes development practices, processes, and guidelines that aid developers in developing software. Support is not only limited to technical procedures, but includes also emotional and social support. This emotional aspect of DX is visible in other places and is not limited to support for developers.

Technical environment or the development infrastructure is concrete example of an object under study. The technical environment includes the development products and services and other tools that developers use to create and develop software. The most studied tool that developers use was the IDE or code editor. The reason behind this is not known, but one possible reason is that developers spend a big part of their time reading and editing code. Therefore also the DX of the technical environment is getting attention in the studies.

Developer's activities in a community relates to the activities a developer has in a community of other developers. Development is a social activity, that often is performed in conjunction with other developers in a team. Developers create communities on different levels that can be very local e.g. a single software team

Wellbeing of developers was discussed in multiple articles. Mainly this was a discussion topic of the research group studying the happiness and unhappiness of developers (Graziotin et al., 2018), (Graziotin et al., 2017c), (Graziotin et al., 2017a), (Graziotin et al., 2017b). They focus mostly on the happy-productive theory of developers. This set of studies and research focuses mostly on the happy-productive theory of developer, but there can also be seen a general interest towards the wellbeing of developers.

Individual developers in a bigger context was not a self-evident object under study. Mäenpää et al. (2017) and Fagerholm and Pagels (2014) were the two most obvious studies of this aspect of DX. However, this can also be found from other objects under study, sometimes directly but mostly more indirectly.

Further iterating on the emerged objects / entities under study, it became more clear that the underlying object under study was the developers' point of view as in Ozkaya (2019). In the past decision relating to software tools or other used techniques and procedures were made by managers and directors. Today there can be seen a trend that has turned the tides, and given developers more decision power when creating and developing their own ways of working. In general there is an interest towards supporting developers and their daily activities in the environment they are working in. The wellbeing of developers is seen important and different viewpoints on how this can be improved

6.1.1.2 Object under study in grey literature

The first iteration of the grey literature and its objects under study revealed categories as *starting experience, API DX, developer portals, usability, developer developing for*

other developers, developer tools, developer support, development platforms. Most obvious object under study were APIs, their usability and the necessities around the APIs.

Starting experience was found in one grey literature article, and it captured an important object of DX. Developers are constantly learning new stuff and adopting new technologies and techniques. The starting experience of new tools, products, and services is in a crucial role when developers pick their selections for upcoming projects and products. The starting experience was also called *"0 to 200"* and *"Time to Hello World"*.

API DX seems also to be a hot topic in the industry. There has been lately discussions about a new term called *API Economy* Tan et al. (2016). APIs have become important assets for companies, and some companies are even relying their whole businesses on to providing APIs. Developers are the main users of APIs and therefore the API's UX and DX are important for the business. DX in the context of API is mostly usability, but includes also other aspects like the starting experience, documentation, and adoptability.

Developer portals are websites and documentation libraries where developers can find information and examples about products and services aimed towards developer, mostly APIs. The portals can include and overview of the service's different functionalities, specifications and guidelines. The gist of developer portals is to allow developers to make well informed decisions.

Developers developing for other developers makes DX different than UX. In many articles in grey literature there were given guidelines on how to create a good DX of a software library or tool. The open source movement and community is a good example of where a developer develops for another developer. Developers are technical and therefore there is also a risk that their libraries and products that they write become too complex and unintuitive to use. Popular software libraries and frameworks are putting effort into creating a good DX, that will benefit the developers using them.

Developer support continues on the category of developers developing for other developers. The category is vague, but in the literature there were discussions about topics like developer relations and support provided by communities. One article discussed about sympathy and empathy towards developers, and how the lack of it can be harmful for the DX.

Development platforms did also get attention in the grey literature. Different development platforms are competing of developers. For example, in the mobile software development context, the main two platforms iOS and Android are "competing" against each other for developers. One way these platforms can gain more developers and build communities is by creating and maintaining a good DX.

As a conclusion we can see that the objects of study of DX in grey literature is more focused on the practical and technical parts of DX. And because most of the found grey literature articles were blog posts, they were focusing on the business of products and services. This caused that the most discussion revolved around usability and technical aspects of DX.

6.1.1.3 Comparison between scientific literature and grey literature in the object of study of DX

There is a lot of similarities between the scientific and grey literature regarding the object under study. Usability was the most occurring object of study in both of them. The technical environment was also clear on both sides.

Scientific articles were dwelling more equally on cognitive, affective, and conative, the 3 aspects of DX while grey literature was clearly more focusing on the cognitive parts of DX. Scientific literature was also focusing more on the feelings and moods of developers, while only a small part of the grey literature focused on this.

6.1.2 Factors that improve or worsen the DX

What improves and/or worsens the DX as in Fagerholm (2015) was also included in the data collection form. When understanding the object of DX, there is also something that improves or worsens the experience, i.e. what influences the experience. The discovered factors were no direct statements found in the sources, but were factors that emerged when analyzing the articles as a more coherent set of articles.

6.1.2.1 Factors that improve/worsen the DX found in scientific literature

Notable factors in scientific literature that improve DX were many, including *shared understanding, available support, collaboration, expectations, mitigating complexity, being in control, ability to tackle challenges, having a meaning as a developer, guidelines, information availability.*

Shared understanding was mostly discussed in the articles discussing Performance Alignment Work (Fagerholm et al., 2015), (Fagerholm et al., 2014). However, this pattern can be found in most of the articles. Having a shared understanding between developers, but also with other stakeholders seems to improve DX.

Collaboration and the benefits that are achieved from working together improves DX. Getting quick feedback from teammates, having agreed policies and roles but being flexible and agile all improve the experience developers have when collaborating with others. Mäenpää et al. (2017) studied the different systems surrounding a developer in a open source community

Mitigating complexity and simplifying things was mentioned explicitly in some articles, but was also apparent in many other articles implicitly. Developers are solving difficult technical problems in complex domains. Developers have to understand both the technical opportunities and constraints, but they also need to understand the business value of what they are doing. This means that developers have a lot on their mind, and therefore all complexity might affect developers DX negatively.

Being in control relates to the developers feeling that they are in control of what they are doing. Overwhelming situations were developers feel that they are not in control might worsen their DX. Here the selection of tools also plays a role. The perceived selection of tools affects also the developer DX (Kati Kuusinen, 2015). Throughout the years there has been a shift in selection of tools, that has more leaned

towards developers. Reaching a flow state is also an important part of feeling of being in control. Achieving a flow state shows that external and internal distractions of developers are minimal, and that developers can focus on the essential, developing. Uncertain situations was

Ability to tackle challenges. Challenges are inevitable when developing software. However, the ability to tackle them was seen as a factor that improves DX. Graziotin et al. (2018) concluded that being stuck in problem solving affects significantly developers' mood and happiness. These problems can be anything relating to development practices like difficulties with a framework or library, algorithm logic, IDE, other development tools, deployment of software, maintenance, etc. The ability to tackle challenges relates also to the before mentioned factor "being in control".

Having a meaning as a developer and having explicit and aligned values within the team and organization improves the motivation of developers, and presumably also the DX. Aligned values can relate to the product and its goals, but also to teammates and their goals. Fagerholm and Pagels (2014) concluded being able to put the value system of a team into words is more important than the methodologies. This can help with understanding what the team is aiming for and what the goals of the team is. Developers that understand the values, and possibly also share them with others might have a greater sense of meaning regarding their development tasks.

Information, guidelines, and support was the most obvious emerged factor that improves DX. Developers are constantly in a learning process, and for example the information, guidelines, support, code examples, and instructions affects the developers DX significantly. Availability, quality, and relevance of all these are important as developers time is limited. The time between getting stuck in problem solving and finding a solution and way out is minimized with good support for developers.

All the combined scientific factors that improve DX shows that DX is improved by mostly affective and conative aspects. This includes the feelings and perceptions of developers. Even if the studied articles were mostly discussing about cognitive aspects like tools and processes, and especially usability, there is the common pattern that developers own feeling and perceptions is what creates a good DX.

6.1.2.2 Factors that improve/worsen the DX found in grey literature

The most significant factor that improves DX was the UX and usability of tools. Usability of APIs, tools, and products all affect the DX. These usability factors include esthetics, exclusivity, fun, completeness, functionality, experience, pleasure, and reliability. When further analyzing the grey literature, factors as *transparency towards developers, explicit guidelines, developers loving their tools, sympathy, feeling of capability, developers are human too, support towards the development process* emerged.

Transparency was one of the indirect factors that emerged in the grey literature. Transparency means in this context the transparency of the development artifacts that developers interact with in their daily work. One example of transparency could be the running application in the production environment, where the state

and the possible errors are available to the developer. Another example, that was not apparent in the literature, could be the transparency of the whole organization where the developer works. Developers that have a honest and clear understanding of the whole organization that they are working in, will probably have a better DX.

Explicit guidelines of tools, products, and APIs reduces the complexity and by that also improves the DX. All kinds of communication like documentation, guides, and examples were factors that contribute to the DX.

Developers using tools they love was an interesting factors found out in one of the articles in grey literature. Developers are spending most of their work time with the development tools, and therefore it is regarding their DX important that they show affection towards them. The perceived choice of developers when selecting tools, and concluded that if developers are able to make their selection themselves, they are going have a significantly better DX.

Sympathy was mentioned as improving DX in one specific article in the grey literature. However, sympathy can be seen in all the other factors as well. Different roles in software team that understand each other, create an experience that is also affecting the DX.

Feeling of capability emerged as an factor and developers need to have the feeling of that they are capable of solving the problems they face to have a good DX. An empowering environment increases the motivation of developers to solve problems.

Developers are human too was a title of one of the titles in the grey literature, and seemed to also be a notable factors when understanding what improves the DX. The phrase "developers are human too" indicates that there might have been a lack of understanding towards developers, that should be changed. Interestingly some articles were also discussing about how developers hate marketers, and that they want to make their tool buy-ins based on facts and experiences, and not by material produced with the aim of selling products. This shows that developers are aware of that they might not get the treatment that they deserve, and that the developers know that they are not just a resource or object utilized by software development organizations.

Support towards the development process was identified to contribute to a better DX. Supporting the development process, like improving tools and services. For example, the management showing their support and allocating resources to improving the development process could improve the DX.

6.1.2.3 Comparison of scentific and grey literature of factors that improve/worsen DX

UX and usability of tools, APIs and products is the most obvious factor that improves DX. This is probably because that is the most easy and tangible definition and measurement of DX. Further analysing the results however shows that there is more of what impacts the DX. In both scientific and grey literature there was the underlying factor of *"feeling of being capable"* that improves the DX. This was also in line with *"being in control"* that was also found. These both categories touch on how the developer sees their own work.

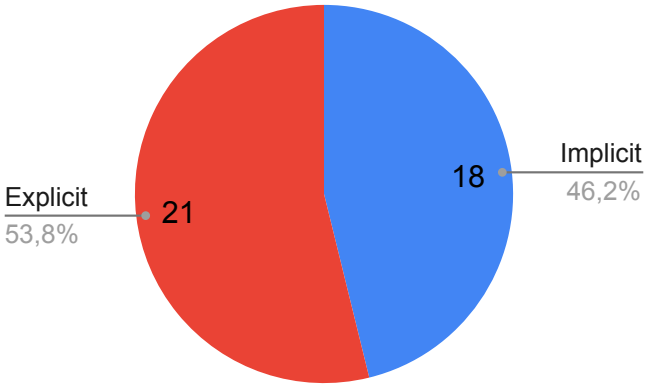
6.1.3 Explicit / implicit definition of DX

Interestingly, during the analysis of the sources, it emerged that some articles were stating an explicit definition of DX, but some were not. To analyze how the the definition of DX is given, all articles and papers were grouped into either having an **implicit** or an **explicit** definition of DX. This division was something that did emerge in the collection and analysis of the material. Many scientific articles use the keyword "developer experience", but only mention DX briefly in their material. This forces the readers to create an understanding of what DX, and "read between the lines" while acquiring the gist of the articles.

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.

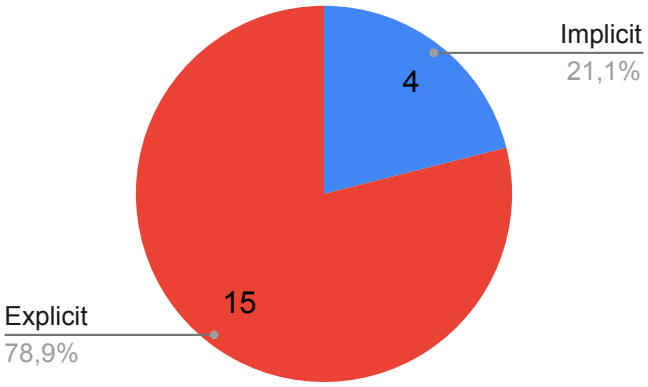
Figure 4: Implicit vs. explicit definition of Developer Experience in scientific literature



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

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.

Figure 5: Implicit vs. explicit definition of Developer Experience in grey literature



6.1.4 Context of DX

From the analysis of the material, there is a clear indication that there are different viewpoints and contexts to DX. The different contexts that emerged from the analysis are listed in table 5. The context were completely unknown before starting the MLR. One source address the topic of DX from multiple contexts and viewpoints.

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

Table 5: Different contexts of Developer Experience that emerged during the literature review

Definition is the context that considers directly the definition of DX. Articles having the context of definition are discussing and explaining the concept of DX. They either implicitly or explicitly add to the definition of DX, either from their own point of view or then from a more broader context.

Development Environment is related to the technical environment where the developer is developing the software. In the found articles the most notable artifact under research was the IDE. Other artifacts that could be included in the development environment could be programming languages and framework or the ease of use of setting up the development environment.

APIs emerged as a context. APIs are interfaces that developers use when building

software. This makes developers users of the APIs, and their UX of an API can formulated to be a DX.

Product or service is related to the APIs, but was selected as a separate context. Software developers use products and services to develop software, and was seen as an important context of DX. The grey literature is heavily influenced by businesses marketing their services or products. To gain visibility and recognition, businesses 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 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 (cite "Devs are people too"), and that this movement has created the concept of DX.

User Experience is the experience that emerges from using a software product or service. Multiple sources explained that DX is a form of UX. 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. 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.

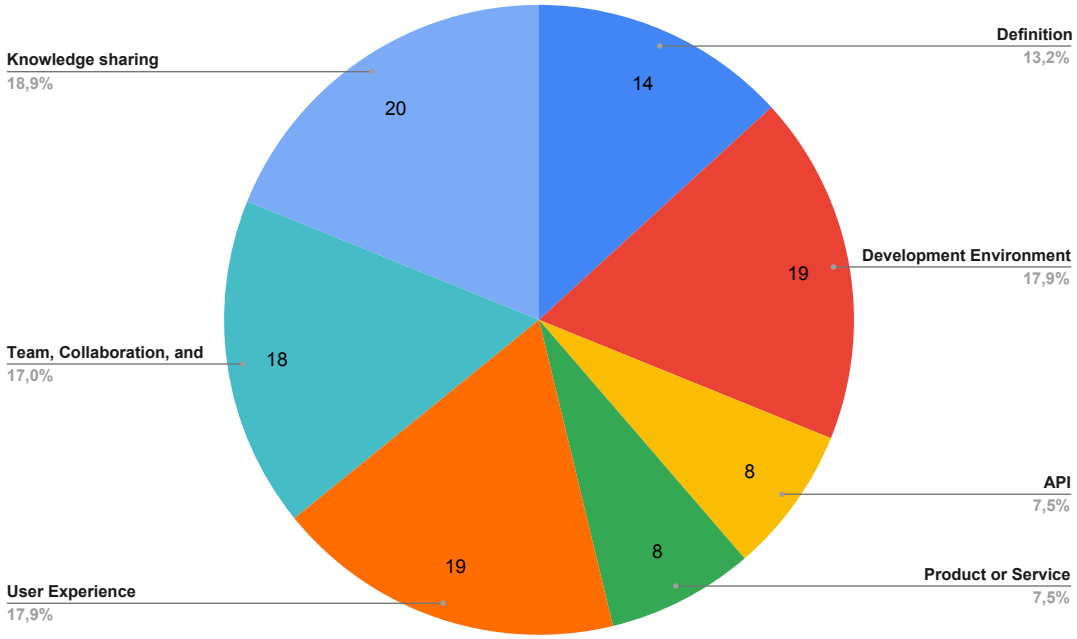
Team, Collaboration, & Community was seen as a specific context that includes the social aspect of DX. Multiple sources discussed about the team and communities where software is developed. They were considered to have a great impact on the DX. The scientific research is more focused on the social parts of DX, and scientific research has taken a step further in this than the grey literature.

Mood & Feelings emerged from the multiple articles discussing the happiness of developers. The mood and feelings should not only be restricted to happiness and/or unhappiness. 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, and desires.

6.1.5 Categories of definitions of DX

To summarize the analysis of what the studied objects of DX are, what the factors that improve/worsen the DX are, the explicit and implicit definitions of DX, and the context of from where DX is looked from, and attempt to understand the definition of DX was made. This was also a data point in the data collection form, and each

Figure 6: Context of Developer Experience in scientific literature



article's short definition of DX was added to the form. The subsequent sections analyze the definitions on both the scientific and grey literature.

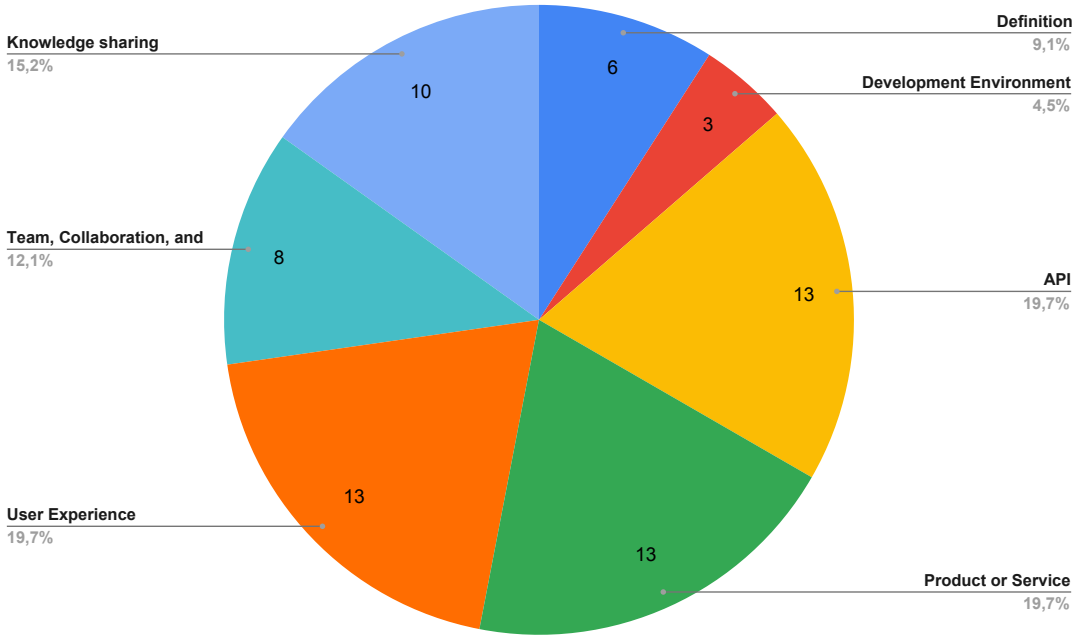
6.1.5.1 Categories of definitions of DX in scientific literature

The most common definition of DX used in the scientific papers is the concept and definition presented by Fagerholm and Münch (2013) and Fagerholm (2015). This definition is at the moment of writing (autumn 2019) the only explicit attempt and deep dive into giving a definition of DX in the scientific literature. There is also further derivations on this concept and definition, but nothing that is refuting their definition and concept or nothing that takes another viewpoint on the definition of DX.

The definition given by Fagerholm and Münch (2013) was apparent when looking at results of the analysis of the definition of DX in scientific literature. The most apparent and noted definition of DX was *developers thoughts and feelings towards their work*. The thoughts and feelings included subjects as the mood of developers, working environments, social aspects, processes, and basically anything that relates to the daily work activities that developers are conducting. Other categories of definitions were *activities, technology, usability, values, agility combined with the starting experience, and expectations.*

Activities refer to the all the different activities, and the experiences of them, of a developer. In the literature these activities are defined as anything that the developer

Figure 7: Context of Developer Experience in grey literature



encounters in their software development tasks. A good example of a definition of DX regarding activities was

> *"Developer Experience relates to all kinds of artifacts and activities that a developer may encounter as part of his/her involvement in software development"*
>
> - Ekwoge, Fontão, and Dias-Neto (2017) based on Fagerholm and Münch (2013)

Technology was another, quite broad, category of definition that was found. Almost all articles discuss about the technologies that developers use, and that they are a main contributor to the definition of DX. In e.g. Nebeling, Zimmerli, and Norrie (2013), measurement of the DX of different mobile development platforms was performed by grading the platforms from the point of view of the technical (e.g. programming language, IDE, debugging tools), subjective (feelings towards the platform), and the level of productivity (ratio of output vs. effort). However, for example, Silva et al. (n.d.) took an exclusively technical viewpoint on the definition of DX, and could be loosely interpreted to be defined as *as the ability to construct models with help of visual modeling languages.*

Usability and UX was seen as being part of DX, and one definition of DX falling into this category was *"UX characteristics of an IDE in the developers development environment"* (Kati Kuusinen, 2015). The other definitions found in this category

are also related to this definition, the UX and usability of any artifact in the software development activities.

Aligned values of developers and the bigger organizations that they are working in captured the different definitions were the DX is defined as the set of values, and their alignment with bigger entities of the software development process. (Fagerholm & Pagels, 2014) state that *Developer experience is formed in the value system of the development environment, both individually and in groups.*

The definition of the selected sources for the majority follow the definition grounded by Fagerholm (2015). Almost all sources take some approach to DX, either cognitive, affective, or conative. This is especially true in the scientific articles, where there are more comprehensive groundwork done on why the research in question is performed.

6.1.5.2 Categories of definitions of DX in grey literature

In grey literature the definition of DX varies a lot more and there are no references to any scientific sources in the grey literature. The definitions are given by the practitioners themselves and from their viewpoint and context of software engineering. 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. During the analysis there were 4 different categories of definitions of DX that emerged. These include *the experience of using a software product, understanding the developer, experience from idea to code to delivering business value*, and finally *support and help for developers.*

The experience of using a software product was interpreted as the definition of DX in multiple articles. Jarman (2017)

6.1.6 Conclusions of the sources

The articles report that DX is an important factor of software engineering. From the different contexts emerged in this study it is apparent that DX can be seen from multiple different viewpoints.

One main finding from the is that overall there is a shift in the empathy and sympathy towards developers. This is also noted in Ozkaya (2019). However, there is still a lot of "objectifying" of developers the same way as there might have been towards users of the early stages of HCI and before user-centered design.

6.2 Validity of results

6.2.1 MLR

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 browsing sessions were used on the browser when performing the searches.

Google has not revealed how the search engine optimization (SEO) works on the Google search engine. However there are some broad guidelines on what a web page has to include, so that it will rank higher on the results page. For companies Google search results is a huge asset and a big opportunity to gain visitor traffic and publicity. Having a web page or site ranking high on the Google search results can probably even determine the existence of some companies businesses. All in all, this means that ranking high on the Google search results page requires knowledge and effort. Therefore we can deduce that the grey literature resources included in this MLR retrieved from Google search are funded or backed by companies that deliberately aim for high ranking on Google searches to sell or promote their products or services.

Depth of data collection has probably varied while performing the MLR. In most cases no explicit analysis was required to extract the required data when collecting it. However, based on the quality of the source sometimes the collection required some analysis to be able extract the data. E.g. understanding the object of DX under study forced sometimes the researcher to "read between the lines", adding their own interpretation and possible bias to the data. This same problem was present also if the research topic of the source was not explicitly studying DX, but still related to concepts of DX.

7 Summary

8 Conclusions

References

Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies*, *58*(2), 207–214. doi:https://doi.org/10.1016/S1071-5819(02)00137-4

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

Chatley, R. (2019). Supporting the developer experience with production metrics. In *Proceedings of the joint 4th international workshop on rapid continuous software engineering and 1st international workshop on data-driven decisions, experimentation and evolution* (pp. 8–11). IEEE Press.

Claussen, J., & Halbinger, M. (2019). The role of pre-innovation platform activity for diffusion success: Evidence from consumer innovations on a 3d printing platform. *Available at SSRN 3453220*.

Corbin, J., Strauss, A. L., & Strauss, A. (2015). *Basics of qualitative research*. sage.

de Lima Fontão, A., Dias-Neto, A., & Santos, R. (2017). Towards a guideline-based approach to govern developers in mobile software ecosystems. In *International conference on software reuse* (pp. 208–213). Springer.

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

Diehl, M., & Stroebe, W. (1987). Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of Personality and Social Psychology*, *53*, 497–509. doi:10.1037/0022-3514.53.3.497

Dong, T., & Khandwala, K. (2019). The impact of cosmetic changes on the usability of error messages. In *Extended abstracts of the 2019 chi conference on human factors in computing systems* (LBW0273). ACM.

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

Ekwoge, O. M. [Oswald Mesumbe], Fontão, A., & Dias-Neto, A. C. (2017). Tester experience: Concept, issues and definition. In *2017 ieee 41st annual computer software and applications conference (compsac)* (Vol. 1, pp. 208–213). IEEE.

Evans, P. C., & Basole, R. C. (2016). Revealing the api ecosystem and enterprise strategy via visual analytics. *Communications of the ACM*, *59*(2), 26–28.

Fagerholm, F. (2015). *Software developer experience: Case studies in lean-agile and open source environments* (Doctoral dissertation, University of Helsinki). Retrieved from http://urn.fi/URN:ISBN:978-951-51-1747-2

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., & 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* (7:1–7:10). EASE '14. doi:10.1145/2601248.2601285

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., & 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., & Münch, J. (2013). Developer experience: Concept and definition. *CoRR*, *abs/1312.1452*. arXiv: 1312.1452. Retrieved from http://arxiv.org/abs/1312.1452

Fagerholm, F., & Pagels, M. (2014). Examining the structure of lean and agile values among software developers. In *International conference on agile software development* (pp. 218–233). Springer.

Faste, H., Rachmel, N., Essary, R., & 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* (pp. 1343–1352). ACM.

Fontão, A., Santos, R., Dias-Neto, A., et al. (2016). Mseco-dev: Application development process in mobile software ecosystems. In *Proceedings of the international conference on software engineering and knowledge engineering* (pp. 317–322).

Fontão, A. [Awdren], Bonifácio, B., Santos, R. P. d., & Dias-Neto, A. C. (2018). Mobile application development training in mobile software ecosystem: Investigating the developer experience. In *Proceedings of the 17th brazilian symposium on software quality* (pp. 160–169). ACM.

Fontão, A. [Awdren], Dias-Neto, A. [Arilo], & Viana, D. (2017). Investigating factors that influence developers' experience in mobile software ecosystems. In *2017 ieee/acm joint 5th international workshop on software engineering for systems-of-systems and 11th workshop on distributed software development, software ecosystems and systems-of-systems (jsos)* (pp. 55–58). IEEE.

Fontão, A. [Awdren], Ekwoge, O. M. [Oswald M], Santos, R., & Dias-Neto, A. C. (2017). Facing up the primary emotions in mobile software ecosystems from developer experience. In *Proceedings of the 2nd workshop on social, human, and economic aspects of software* (pp. 5–11). ACM.

Fontão, A. [Awdren], Santos, R., & Dias-Neto, A. (2015). Research opportunities for mobile software ecosystems. In *Proceedings of the 9th workshop on distributed software development, software ecosystems and systems-of-systems (wdes)* (pp. 97–98).

Garousi, V., Felderer, M., & 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* (26:1–26:6). EASE '16. doi:10.1145/2915970.2916008

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, *106*, 101–121. doi:https://doi.org/10.1016/j.infsof.2018.09.006

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2017a). Consequences of unhappiness while developing software. In *Proceedings of the 2nd international*

workshop on emotion awareness in software engineering (pp. 42–47). IEEE Press.

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2017b). On the unhappiness of software developers. In *Proceedings of the 21st international conference on evaluation and assessment in software engineering* (pp. 324–333). ACM.

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

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

Hassenzahl, M. (2003). The thing and i: Understanding the relationship between user and product. In *Funology* (pp. 31–42). Springer.

Hassenzahl, M., & Tractinsky, N. (2006). User experience-a research agenda. *Behaviour & information technology*, *25*(2), 91–97.

Henriques, H., Lourenço, H., Amaral, V., & Goulão, M. (2018). Improving the developer experience with a low-code process modelling language. In *Proceedings of the 21th acm/ieee international conference on model driven engineering languages and systems* (pp. 200–210). ACM.

Info, S. (2011). Euroopan parhaat työpaikat 2011 -tutkimus: Reaktor euroopan 2. paras työpaikka. Retrieved October 27, 2019, from https://www.sttinfo.fi/tiedote/euroopan-parhaat-tyopaikat-2011--tutkimus-reaktor-euroopan-2-paras-tyopaikka?publisherId=3283%5C&releaseId=48887

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.

Ivo, A. A., Guerra, E. M., Porto, S. M., Choma, J., & Quiles, M. G. (2018). An approach for applying test-driven development (tdd) in the development of randomized algorithms. *Journal of Software Engineering Research and Development*, *6*(1), 9.

Jarman, S. (2017). The best practices for a great developer experience (dx). Retrieved September 10, 2019, from https://hackernoon.com/the-best-practices-for-a-great-developer-experience-dx-9036834382b0

Kärpänoja, P., Virtanen, A., Lehtonen, T., & Mikkonen, T. (2016). Exploring peopleware in continuous delivery. In *Proceedings of the scientific workshop proceedings of xp2016* (p. 13). ACM.

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

Kuusinen, K., Petrie, H., Fagerholm, F., & Mikkonen, T. (2016). Flow, intrinsic motivation, and developer experience in software engineering. In H. Sharp & T. Hall (Eds.), *Agile processes, in software engineering, and extreme programming* (Vol. 251). XP 2016, Springer, Cham.

Kuusinen, K. [Kati]. (2015). Software developers as users: Developer experience of a cross-platform integrated development environment. In P. Abrahamsson, L. Corral, M. Oivo, & B. Russo (Eds.), *Product-focused software process improvement* (pp. 546–552). Cham: Springer International Publishing.

Kuusinen, K. [Kati]. (2016). Are software developers just users of development tools? assessing developer experience of a graphical user interface designer. In *Human-centered and error-resilient systems development* (pp. 215–233). Springer.

Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009). Understanding, scoping and defining user experience: A survey approach. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 719–728). ACM.

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

Macvean, A., Church, L., Daughtry, J., & Citro, C. (2016). Api usability at scale. In *Ppig* (p. 26).

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*.

Miranda, T., Challenger, M., Tezel, B. T., Alaca, O. F., Barišić, A., Amaral, V., ... Kardas, G. (2018). Improving the usability of a mas dsml. In *International workshop on engineering multi-agent systems* (pp. 55–75). Springer.

Moilanen, J., Niinioja, M., Seppänen, M., & Honkanen, M. (2018). Api-talous 101. *Alma Talent Oy*.

Morales, J., Rusu, C., Botella, F., & Quinones, D. (2019). Programmer experience: A systematic literature review. *IEEE Access*, *PP*, 1–1. doi:10.1109/ACCESS.2019.2920124

Morse, J. M., Stern, P. N., Corbin, J., Bowers, B., Charmaz, K., & Clarke, A. E. (2016). *Developing grounded theory: The second generation*. Routledge.

Murphy, L., Kery, M. B., Alliyu, O., Macvean, A., & 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)* (pp. 249–258). IEEE.

Myers, B. A., & Stylos, J. (2016). Improving api usability. *Communications of the ACM*, *59*(6), 62–69.

Nazário, M. F. C., Guerra, E., Bonifácio, R., & Pinto, G. (n.d.). Detecting and reporting object-relational mapping problems: An industrial report.

Nebeling, M., Zimmerli, C., & Norrie, M. (2013). Informing the design of new mobile development methods and tools. In *Chi'13 extended abstracts on human factors in computing systems* (pp. 283–288). ACM.

Ollis, G. (2019). *Helping developers to help each other: A technique to facilitate understanding among professional software developers.* (Doctoral dissertation, Bournemouth University).

Oran, A. C. (2017). A set of artifacts and models to support requirements communication based on perspectives. *ACM SIGSOFT Software Engineering Notes*, *41*(6), 1–5.

Ozkaya, I. (2019). The voice of the developer. *IEEE Software*, *36*(5), 3–5.

Palviainen, J., Kilamo, T., Koskinen, J., Lautamäki, J., Mikkonen, T., & Nieminen, A. (2015). Design framework enhancing developer experience in collaborative coding environment. In *Proceedings of the 30th annual acm symposium on applied computing* (pp. 149–156). SAC '15. doi:10.1145/2695664.2695746

Pinsonneault, A., Barki, H., Gallupe, R. B., & Hoppen, N. (1999). Electronic brainstorming: The illusion of productivity. *Information Systems Research*, *10*(2), 110–133. doi:10.1287/isre.10.2.110. eprint: https://pubsonline.informs.org/doi/pdf/10.1287/isre.10.2.110

Pinter, C., Lasso, A., & Fichtinger, G. (2019). Polymorph segmentation representation for medical image computing. *Computer methods and programs in biomedicine*, *171*, 19–26.

Reaktor. (2019a). Reaktor careers. Retrieved October 27, 2019, from https://www.reaktor.com/careers/

Reaktor. (2019b). Reaktor home. Retrieved October 27, 2019, from https://www.reaktor.com/

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

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

Romano, S., Scanniello, G., Fucci, D., Juristo, N., & Turhan, B. (2018). The effect of noise on software engineers' performance. In *Proceedings of the 12th acm/ieee international symposium on empirical software engineering and measurement* (p. 9). ACM.

Al-Samarraie, H., & Hurmuzan, S. (2018). A review of brainstorming techniques in higher education. *Thinking Skills and Creativity*, *27*, 78–91. doi:https://doi.org/10.1016/j.tsc.2017.12.002

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* (pp. 11–26).

Scupin, R. (1997). The kj method: A technique for analyzing data derived from japanese ethnology. *Human organization*, 233–237.

Silva, J., Barišic, A., Amaral, V., Goulão, M., Tezel, B. T., Alaca, O. F., ... Kardas, G. (n.d.). Comparing the usability of two multi-agents systems dsls: Sea_ml++ and dsml4mas study design.

Talouselämä. (2019). Ilmapiiri ei synny sanelemalla. Retrieved October 27, 2019, from https://www.talouselama.fi/uutiset/ilmapiiri-ei-synny-sanelemalla/fe137d60-5f49-3690-a6f2-3d3eab5fce54

Tan, W., Fan, Y., Ghoneim, A., Hossain, M. A., & Dustdar, S. (2016). From the service-oriented architecture to the web api economy. *IEEE Internet Computing*, *20*(4), 64–68.

van Deursen, A. J., & van Dijk, J. A. (2009). Using the Internet: Skill related problems in users' online behavior. *Interacting with Computers*, *21*(5-6), 393–402. doi:10.1016/j.intcom.2009.06.005. eprint: http://oup.prod.sis.lan/iwc/article-pdf/21/5-6/393/2267736/iwc21-0393.pdf

Vilkko, N. (2016). *Työskentely ketterässä tiimissä ja työn imun kokemus; working in an agile team and the experience of work engagement* (G2 Pro gradu, diplomityö). Retrieved from http://urn.fi/URN:NBN:fi:aalto-201702142323

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering* (p. 38). Citeseer.

Zhang, R., & Xie, G. (2018). Toward understanding iot developers in chinese startups. In *Proceedings of the 22nd international conference on evaluation and assessment in software engineering 2018* (pp. 181–186). ACM.

A Resources of the Multivocal Literature Review

Table A1: Scientific articles

1.	Fagerholm and Münch (2013)	Developer experience: concept and definition
2.	Kuusinen et al. (2016)	Flow, Intrinsic Motivation, and Developer Experience in Software Engineering
3.	Nebeling et al. (2013)	Informing the Design of New Mobile Development Methods and Tools
4.	Henriques, Lourenço, Amaral, and Goulão (2018)	Improving the Developer Experience with a low-code process modelling language
5.	Awdren Fontão, Bonifácio, Santos, and Dias-Neto (2018)	Mobile Application Development Training in Mobile Software Ecosystem: Investigating the Developer eXperience
6.	Palviainen et al. (2015)	Design framework enhancing developer experience in collaborative coding environment
7.	Graziotin et al. (2017c)	Unhappy Developers: Bad for Themselves, Bad for process, and Bad for Software Product
8.	Awdren Fontão, Dias-Neto, and Viana (2017)	Investigating factors that influence developers' experience in mobile software ecosystems
9.	Graziotin et al. (2017b)	On the Unhappiness of Software Developers
10.	Graziotin et al. (2017a)	Consequences of Unhappiness While Developing Software
11.	Graziotin et al. (2018)	What happens when software developers are (un)happy
12.	Fagerholm et al. (2014)	How do software developers experience team performance in Lean and Agile environments?

13.	Fagerholm et al. (2015)	Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments
14.	Chatley (2019)	Supporting the Developer Experience with Production Metrics
15.	Awdren Fontão, Ekwoge, Santos, and Dias-Neto (2017)	Facing up the primary emotions in Mobile Software Ecosystems from Developer Experience
16.	Murphy et al. (2018)	API Designers in the Field: Design Practices and Challenges for Creating Usable APIs
17.	Pinter, Lasso, and Fichtinger (2019)	Polymorph segmentation representation for medical image computing
18.	Dong and Khandwala (2019)	The Impact of "Cosmetic" Changes on the Usability of Error Messages
19.	Mäenpää et al. (2017)	Entering an ecosystem: The hybrid OSS landscape from a developer perspective
20.	Kati Kuusinen (2015)	Software Developers as Users: Developer Experience of a Cross-Platform Integrated Development Environment
21.	Fagerholm and Pagels (2014)	Examining the Structure of Lean and Agile Values Among Software Developers
22.	Miranda et al. (2018)	Improving the Usability of a MAS DSML
23.	Fontão, Santos, Dias-Neto, et al. (2016)	MSECO-DEV: Application Development Process in Mobile Software Ecosystems
24.	Awdren Fontão, Santos, and Dias-Neto (2015)	Research Opportunities for Mobile Software Ecosystems
25.	Myers and Stylos (2016)	Improving API usability
26.	Macvean, Church, Daughtry, and Citro (2016)	API Usability at Scale
27.	Kati Kuusinen (2016)	Are Software Developers Just Users of Development Tools? Assessing Developer Experience of a Graphical User Interface Designer
28.	Kärpänoja, Virtanen, Lehtonen, and Mikkonen (2016)	Exploring Peopleware in Continuous Delivery

29.	Ekwoge et al. (2017)	Tester Experience: Concept, Issues and Definition
30.	Romano, Scanniello, Fucci, Juristo, and Turhan (2018)	The effect of noise on software engineers' performance
31.	Ivo, Guerra, Porto, Choma, and Quiles (2018)	An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms
32.	de Lima Fontão, Dias-Neto, and Santos (2017)	Towards a Guideline-Based Approach to Govern Developers in Mobile Software Ecosystems
33.	Morales et al. (2019)	Programmer eXperience: A Systematic Literature Review
34.	Claussen and Halbinger (2019)	The Role of Pre-Innovation Platform Activity for Diffusion Success: Evidence from Consumer Innovations on a 3D Printing Platform
35.	Oran (2017)	A Set of Artifacts and Models to Support Requirements Communication Based on Perspectives
36.	Nazário, Guerra, Bonifácio, and Pinto (n.d.)	Detecting and Reporting Object-Relational Mapping Problems: An Industrial Report
37.	Zhang and Xie (2018)	Toward Understanding IoT Developers in Chinese Startups
38.	Silva et al. (n.d.)	Comparing the Usability of two Multi-Agents Systems DSLs: SEA_ML++ and DSML4MAS Study Design
39.	Ollis (2019)	Helping developers to help each other: a technique to facilitate understanding among professional software developers

Table A2: Grey literature

1.	The Best Practices for a Great Developer Experience (DX)	https://hackernoon.com/the-best-practices-for-a-great-developer-experience-dx-9036834382b0
2.	Developer Experience (DX) — Devs Are People Too	https://hackernoon.com/developer-experience-dx-devs-are-people-too-6590d6577afe
3.	Great Developer Experiences and the People Who Make Them	https://medium.com/apis-and-digital-transformation/great-developer-experiences-and-the-people-who-make-them-b97b544caba9

4.	Developer Experience (DX) | Heroku	https://www.heroku.com/dx
5.	Contributing to a great developer experience (DX) as a designer	https://uxdesign.cc/contributing-great-developer-experience-designer-e1f497b0fb4
6.	Developer eXperience. How I missed it before?	https://dev.to/stereobooster/developer-experience-how-i-missed-it-before-47go
7.	What is Developer Experience? - EveryDeveloper	https://everydeveloper.com/developer-experience/
8.	Building the Developer Experience (DX) From the Ground Up	https://blog.argoproj.io/building-the-developer-experience-dx-from-the-ground-up-8254d50457f5
9.	API Developer Experience (DX) Resources	https://www.moesif.com/blog/api-guide/api-developer-experience/
10.	4 Must-Read Articles on Developer Experience	https://www.kennethlange.com/4-must-read-articles-on-developer-experience-dx/
11.	Workflows for the New Developer Experience	https://thenewstack.io/workflows-for-the-new-developer-experience/
12.	APIs for Humans: The Rise of Developer Experience	https://www.hellosign.com/blog/the-rise-of-developer-experience
13.	Effective Developer Experience (DX) | UX Magazine	https://uxmag.com/articles/effective-developer-experience
14.	What is API Developer Experience and Why It Matters	https://www.infoq.com/news/2015/10/api-developer-experience/
15.	Developer Experience: What and Why – The AppsLab	http://theappslab.com/2017/04/04/developer-experience-what-and-why/
16.	What exactly is Developer Experience? - APIMATIC	https://blog.apimatic.io/what-exactly-is-developer-experience-1646b813df14
17.	Developer experience - Sanity.io	https://www.sanity.io/developer-experience
18.	4 APIs Doing Developer Experience Really Well | Nordic APIs |	https://nordicapis.com/4-apis-doing-developer-experience-really-well/