A multivocal literature review on developer experience

Anders Nylund

School of Science

Thesis submitted for examination for the degree of Master of
Science in Technology.
Espoo 18.2.2020

Supervisor

Ph.D Fabian Fagerholm

Advisor

Ph.D Fabian Fagerholm

Aalto University
School of Science

**Aalto University
School of Science**

Author Anders Nylund		
Title A multivocal literature review on developer experience		
Degree programme Computer, Communication and Information Sciences		
Major Software and Service Engineering		**Code of major** SCI3043
Supervisor Ph.D Fabian Fagerholm		
Advisor Ph.D Fabian Fagerholm		
Date 18.2.2020	**Number of pages** 67+5	**Language** English

Abstract

Developer experience is a concept that has emerged from the notion of User experience. Developer experience differs from user experience that it is considering developers instead of users. However, developers can be seen as users of software development tools. The concept of developer experience has initiated new ways of thought and practices for software development. Even if there has been previous research on the topic of developer experience, there are still open questions and vague definitions on what developer experience is and what it means in practice. To better understand the definition of developer experience this thesis' focus is on understanding the current state of research of developer experience. The study was performed as multivocal literature review where articles written by both academics and practitioners were analysed. The results from the review indicate that developer experience can be seen as a way to understand the feelings and perceptions of the developers in a software development context. Developer experience is studied both in the academics and the industry, where they both focus on improving the developers experiences when developing software. The difference between the academics and practitioners is that academics are focused on understanding the underlying phenomena of developer experience, while practitioners have seen a lot of different opportunities and business value in studying and improving developer experience.

Keywords	Developer Experience, Multivocal Literature Review, Software Developer, Software Engineering

Författare Anders Nylund		
Titel En multivocal litteraturöversikt på utvecklarupplevelse		
Utbildningsprogram Computer, Communication and Information Sciences		
Huvudämne Software and Service Engineering	**Huvudämnets kod** SCI3043	
Övervakare Ph.D Fabian Fagerholm		
Handledare Ph.D Fabian Fagerholm		
Datum 18.2.2020	**Sidantal** 67+5	**Språk** Engelska

Sammandrag

Utvecklarupplevelse är ett begrepp som har framträtt från definitionen av användarupplevelse. Utvecklarupplevelse skiljer sig från användarupplevelse från att utvecklarupplevelse är förknippad med utvecklare som både utvecklar nya produkter och tjänster men också som också agerar som användare av verktyg och hjälpmedel till utveckling. Användarupplevelse fokuserar endast på personer som använder produkter och tjänster. Konceptet av utvecklarupplevelse har resulterat till nya tankesätt och metoder inom mjukvaruutveckling. Fastän det har gjorts forskning inom ämnet av utvecklarupplevelse, så finns det ännu öppna frågor om utvecklarupplevelse och vad det innebär. Existerande definitioner på utvecklarupplevelse är till viss del vaga. För att bättre förstå definitionen av utvecklarupplevelse, fokuserar detta diplomarbete på att förstå den nuvarande statusen på forskning av utvecklarupplevelse. Studien i detta diplomarbete utfördes som en litteraturöversikt, där artiklar skrivna av både akademiker och experter inom industrin analyserades. Resultaten från litteraturöversikten indikerar att utvecklarupplevelse kan ses som ett sätt att förstå utvecklarnas känslor och uppfattningar i mjukvaruutveckling. Utvecklarupplevelse studeras både inom akademin och industrin, där båda fokuserar på att förbättra utvecklarnas upplevelser när man utvecklar programvara. Skillnaden mellan akademiker och industrin är att akademiker är inriktade på att förstå de underliggande fenomenen av utvecklarupplevelse, medan industrin har sett många olika möjligheter med att skapa affärsvärde av att förstå och förbättra utvecklarupplevelse.

Nyckelord Utvecklarupplevelse, Multivocal litteraturöversikt, Mjukvaruutvecklare, Mjukvaruutveckling	

Preface

I want to thank everyone that have shown interest toward this thesis. I am a bit surprised over how much others have been interested in the topic of developer experience. From the beginning to the very end I've had discussions about the topic with my friends and colleagues, and these discussions have been greatly helpful and they are appreciated. The discussions and different opinions have helped me to validate the selection of the topic, and to further iterate on the research problem and the presented research questions. I want to also thank my advisor Fabian Fagerholm for supporting me during the thesis and giving me advice on how to continue and tackle the problems that I have faced. Finally I want to thank Reaktor for giving me the opportunity to work on this thesis. My colleagues at Reaktor have showed interest and provided good insights into the thesis.

Otaniemi, 18.2.2020

Anders Nylund

Contents

Thesis dictionary

API	Application Programming Interface
DX	Developer Experience
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
OSS	Open Source Software
PAW	Performance Alignment Work
SE	Software Engineering
SEO	Search Engine Optimization
SLR	Systematic Literature Review
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 relatively 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 understanding and skills are required to be able to implement the required product, product or artifact. However software engineering is also a highly social activity, and therefore it has been noted that human factors are among the most important when regarding software development performance (DeMarco & Lister, 2013).

Software developers are in an interesting role. They are creators and designers when they write the code and design the logic that makes up the software, but also in the meantime they are users of tools that they utilize in their craft. 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 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 emerged. 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 everything around the developer, and everything that affects the software development practice.

This study aims to build on the understanding of what phrases like *"I love using this framework as it has such a good developer experience!"*, and *"The conventions of the project were confusing and caused a really bad developer experience"* mean. Focus of this study is on building on the understanding and definition of DX on a broad level.

1.1   Motivation

At the time of writing (February 2020), 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 could indicate 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 of 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 that is created when developer participates in software
development work.

Law, Roto, Hassenzahl, Vermeeren, and Kort (2009) conducted a comprehensive
research on the notion of UX. UX is as a well defined and understood concept and
term significantly more mature than what DX is, 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 understood definition.

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 the
developer's side of things and 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 potential for many different kinds of gains in 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 philo-
sophical stance, and the selected research methods e.g the research protocol. They
encourage this because other researchers can then better understand and correctly
interpret the study and possibly replicate the study.

During this study the research problem and questions evolved and were modified
while more understanding and knowledge about the research topic were created and
got accumulated. The starting point of the study was to understand how DX is linked
to software project outcomes. However, this was proven 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: What is the definition and aspects of Developer Experience, and how do they differ between scientific literature and literature written by practitioners?

RQ 1	What objects and entities have been studied with respect to developer experience?
RQ 2	What methods have been used to study developer experience?
RQ 3	What is known about factors that improve or worsen developer experience?
RQ 4	From what contexts is developer experience looked at?
RQ 5	What are the definitions given to developer experience?

Table 1: The research questions

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

The research problem is a Description and Classification question, but also a Descriptive-Comparative question. The comparison initiated by the problem helps to better understand the definition of DX, and creates the ground for this thesis. The answers to this problem will help to understand the phenomenon better, but also point out the absence of definitions.

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*. The nature of these selected research questions will guide the research and guide the selection of the used methods and techniques.

RQ1 prompts to study the different objects that can affect the DX. This question was created from the theoretical framework presented by Fagerholm (2015, p. 70) (Figure 3).

RQ2 studies the different research methods that have been used in both grey and scientific literature. Answering this could help to understand what might work and what might not work when studying and researching DX.

RQ3 continues on RQ1, and further evolves the focus of the problem into trying to understand what makes the DX better or worse on the possible different object and entities of DX.

RQ4 is asked because it was noted when reading the preliminary literature that different articles all take their own viewpoint and context from which DX is looked at. It is important to understand what parts of DX is being studied and why some specific results have been achieved. Scientific and grey literature do presumably have different contexts of DX that they are studying.

Finally, RQ5 is an ambitious questioning that tries to summarize the previously presented questions. There are multiple different definitions to UX, e.g. (DIS, 2009), (Law et al., 2009), (Hassenzahl & Tractinsky, 2006). RQ5 is searching for a concise definition like there exists for UX.

1.3   Scope and focus

The scope of the thesis changed remarkably during the research process. 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. This change of the scope is discussed further in Section 3.1.

The scope of this thesis is relatively narrow. There is a lot of open questions about the concept of DX, and the thesis is only scratching the surface of the current state and understanding of the definition of DX. Therefore this thesis focuses on getting a broad view on the research problem. It is acknowledged that understanding the whole concept DX is difficult, and that there are no definitive answers when trying to define what DX is about. This thesis doesn't give any clear and well defined definition of DX, and there are no "right" or "wrong" answers presented in the thesis.

However, despite the difficult topic, this thesis' aim is to lay groundwork in looking into the current state of DX and its definition in the current literature.

1.4   Structure of the thesis

The structure of the thesis is as follows: Section 1. presents and motivates the problem in this thesis. A traditional literature review on the subject of developer experience is presented in Section 2. Section 3. discusses the different options of research methods, motivates the selected ones, and presents the research protocol of the multivocal literature review. Section 4. presents the results of the thesis. Section 5. continues on the analysis of the results and answer the research questions. Finally, Section 6. draws conclusions on the presented results of the thesis.

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 based on the initial literature research, and they create an outline for the multivocal literature review. 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 3.3.

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 to development are *programmer experience*, and finally *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 therefore brand experience is a more broad concept than UX. *Product experience* is narrower than 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

2.1 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 is. Interestingly, this relates exactly to the situation of what the understanding and definition of DX is at the moment.

Today, UX is still a more of a subjective than objective measure. However, in the field of user-centered design there has been interest towards finding a common definition of UX. One example of this is DIS (2009), the ISO 9241-210 standard, where an attempt to standardize human-centered design is done. This standard includes also a definition of UX. DIS (2009) is an attempt of standardization from the point of view of the industry. The fact that this standard has been implemented and

exists shows that there has been ambiguity of what the definitions of human-centered design and UX are.

On top of the ISO 9241-210 standard, Mirnig, Meschtscherjakov, Wurhofer, Meneweger, and Tscheligi (2015) have done a quasi-formal analysis on the definition of UX given in DIS (2009). They go into great detail and break down the definition word by word, and by that try to create a better understanding of the given definition and the whole concept of UX.

As DX can be seen as a derivative of UX, it can be seen that defining DX is not a trivial task. However, a well chosen definition can bring great value, as it creates a common and shared understanding.

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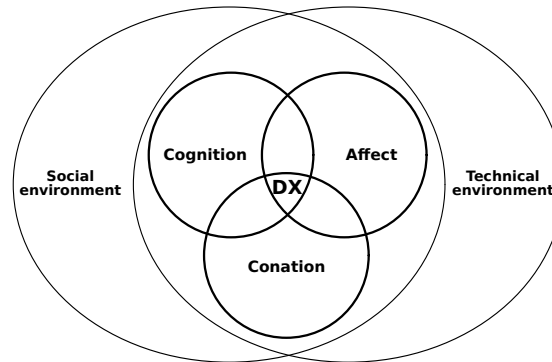
Figure 1: The Developer Experience Concept (Fagerholm, 2015)

Current scientific literature on the definition of DX are few in numbers. DX has however a comprehensive and detailed definition by Fagerholm (2015) that build upon Fagerholm and Münch (2013). Fagerholm's (2015) 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.

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

Fagerholm and Münch (2013) takes an approach from psychology, and divide 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.

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

The technical environment, including programming languages, infrastructure, processes, techniques, plans, diagrams etc., is also part of the DX. A developer is interacting with these artifacts and that generates an experience. Activities with these artifacts are both experienced as an individual but also as a group. Timewise, 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, 2015; Figure 3)

Figure 3: Theoretical Framework of Developer Experience (Fagerholm, 2015)

DX can be seen as important to the practice of software development and engineering 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 affects the DX, so that the platform or environment can be designed to be aligned with the developers using it (Fagerholm & Münch, 2013).

The theoretical framework by Fagerholm (2015) presented in Figure 3 is a presentation of the activities of developers in a individual and social environment, and how the experiences arise. The framework includes aspects as experience *objects, formations, influencers, content, progression, behaviour outcome* and *object outcome.* These different aspects can be used to study DX from a wide variety of different viewpoints.

2.3 Programmer Experience

In this thesis a software developer is defined as 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.

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. They also concluded that 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. Morales et al. (2019) conclude that PX is *"the result of the intrinsic motivations and perceptions of programmers about the use of development artifacts"*.

However, there is ambiguity around the definition of PX. HPI Software Architecture Group (2019) use the term *"Programming Experience"*, and their definition of PX resemble the one of DX. They are not only talking about the technical details of programming, but also about aspects as the live domain, understanding the requirements, visualization, immediate experience, and joy related to the experience.

2.4   Starting Experience

Murphy, Kery, Alliyu, Macvean, and Myers (2018) discuss about "User Starting Experience" mostly from the point of view of developers using API's and products. However, this shouldn't be limited or restricted to only APIs, programming languages, or frameworks. The concept of starting experience could be used in the measurement of introducing or adopting anything new related to software development e.g. introducing a new development technique or process to a software development team.

Murphy et al. (2018) introduce the concepts of *0 to 200* and *Time to Hello World.* These both concepts are discussing the approachability of different artifacts. 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 i.e. the directors or managers. 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.

The term *0 to 200* originates 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. The term *Time to Hello World* has its grounds in the introduction and adoption of a new programming language, where often the first task is to print or log the text "Hello World" to the output channel e.g. the console or terminal. Adopting the basic structures and building block of the programming language prove how easy it is to get stared and going with the programming language, and therefore can be a measurement of the approachability of the programming language.

3 Research methods

Easterbrook et al. (2008) define a set of guidelines for empirical research in software engineering. Despite that the guidelines are aimed for empirical studies, they are used in this literature review study where they are applicable. 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, and replicability.

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.

3.1 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 a pragmatic approach. A constructivist approach sees the problem in a way where the human context is always present, and that it is an important part of the research and the study. An approach of **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 a controlled and isolated environment (Easterbrook et al., 2008).

3.2 Research approach and method

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 or that try to describe something (Easterbrook et al., 2008). The definition of DX given by Fagerholm and Münch (2013) and Fagerholm (2015) function as the basis of the study, but they are only giving the abstract concept to build upon. This concept and framework has not yet been widely taken into practice in empirical research. From Fagerholm (2015) the theoretical framework was also used for guiding the research.

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 Multivocal Literature Review (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. An MLR takes into account grey literature, and therefore includes the perspectives of the current state of the topic in the industry. It also allows to answer exploratory questions where the existence of a subject or topic is abstract or vague. To be able to investigate and define the definition of DX the study had 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.

Other research approaches that could have been suitable for this thesis were e.g. exploratory case studies, interview studies, or even observational studies at a case company following a developers day to day work. They all could have been conducted as studies with an exploratory approach. However, the problem with selecting one of these methods would have been the little background knowledge about the topic and the phenomena of DX what would have made it difficult to get a broad view of the topic. It would have required some assumptions, and a lock down into some specific viewpoint of what DX is. Therefore, and because of the current situation with the research of DX, made it difficult to study DX with empirical research techniques. Instead, a comprehensive literature review was seen as a better approach in this study. A MLR gave a broad view of the current state of DX. It allowed to avoid the lock down of a specific viewpoint and approach to DX.

The main goal of a MLR is to get objective viewpoints and results on the topic that is being studied. MLR's, and systematic literature reviews in general, are utilized to tackle the problem of researcher bias. However, there still needs to be some underlying principle that guides the MLR. 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)

For the present researcher taking an approach that enabled an open mindset allowed to question suitable research questions. This approach was the most suitable in this thesis, both from the viewpoint of the research topic and researchers own skills, competency, and way of thought. Therefore the techniques presented in Morse et al. (2016) are also utilized in this research in all parts of it including the material selection, but also the analysis of the results.

3.3 Multivocal literature review

Traditionally in Systematic Literature Reviews (SLR) in Software Engineering (SE), 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 published in non-scientific

forums without peer review e.g. informal material and publications, are not considered in SLRs (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)

In the citation above (The Prague Definition of grey literature), the definition of grey literature is strict and therefore does not allow 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).

3.3.1 The motivation behind a Multivocal Literature Review

DX is an novel and abstract concept and framework, and therefore there has been little formal research on the topic. Garousi et al. (2019) classify literature to three different levels. White, grey, and black. White literature has high credibility and includes books, magazines, government reports, and white papers. The second level, grey literature, includes e.g. news articles, presentations, and wiki articles. Black literature has the lowest credibility of the types. Blogs, emails, and tweets are classified as black literature. Based on the these three levels of literature, DX could be seen even to be in the category of black literature. DX can be seen to still be mainly on the level of ideas, concepts, and thoughts.

As pointed by Fagerholm (2015), the framework they present can be used to guide inquiries into DX. The goal was to create an understanding of the definition of DX from the selected 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. A researcher conducting a traditional literature review, where material is picked based on and with help of some non-systematic procedure, is likely going to be biased by the researchers own expertise and experiences. Because, if there is no rules to follow when picking articles, the researcher is going to pick them by their own interest and what they personally find relevant e.g. the study is going to be affected by the subjective opinions of the researcher. To avoid researcher bias, a better approach for the literature is a systematic literature review (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. An SLR is also suitable to study DX, as DX can be seen as a subjective concept of the developer.

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). A SLR would include only the scientific papers, and therefore it might not be sufficient to only focus on that. In a MLR the grey literature 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 grey literature can provide insights about the field of SE, and especially about DX.

3.3.2 The review protocol of the Multivocal Literature Review

All data of the MLR can be found from Nylund (2020). The data collection is based on the example shown in Garousi et al. (2019). Kitchenham and Charters (2007) point out that following a review protocol is crucial when performing a SLR. The following sections are a brief overview of the review protocol used in this thesis.

3.3.2.1 Search process

The search process was divided into a database search and followed by 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.

Database search Initially the search of scientific literature was not systematic and the very first iteration of the search was performed as a manual search by the author from various libraries to gather the scientific literature. These non-systematic searches for literature helped to plan, design, and focus the research and the MLR. The selected databases and libraries of sources of scientific literature are listed in Table 2.

Wohlin (2014) and Kitchenham and Charters (2007) 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 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.

Garousi et al. (2019) also mention that SLRs in general are helpful when there is a huge amount of studies on a specific topic. In these cases, doing a SLR helps to index the studies, which then also helps further studies on the same topic.

Other possible libraries and databases that could have been used are listed in Table 3. Google scholar does not provide advanced search that have the required search options, especially search based on author keyword. Also, ScienceDirect och Scopus, do not have the required advanced search features. Advanced search by

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

author keyword was a search criteria and because of that these libraries and databases were excluded from the initial search for scientific sources.

For gathering grey literature, the Google search engine (https://www.google.com) was used. Because of a limited amount of time available for the research, a decision to only select the 2 first result pages of Google was made. 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.

Search string for both scientific and grey literature was **"developer experience"**. As the aim of the MLR review was to get an overview of the definition of DX, only that specific search string was used. This was an attempt to assure that the 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 on the scientific articles, 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 their study might be related to a concept called DX.

Searching by author keyword gave search 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 or any other search strategy. There is no exact data gathered on this, but based on the manual unstructured test searches that was performed, the amount of unrelated articles was significantly higher than compared to when searching by author keyword. In full text searches, the consequent words "Developer Experience" appear in the content of the article, and therefore they also appear in the search results. By using author keyword, this limits the search to only cases where the author has deliberately

selected the keyword. 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.

Snowballing The results from the database search were good, and the relevancy of the found sources were satisfactory. However there was a suspicion that the amount of found results were too few. 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 round of backward and forward snowballing resulted in 5 articles from backward and 14 articles from forward snowballing.

Snowballing was not used on grey literature.

3.3.2.2   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.

3.3.2.3   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.

3.3.2.4   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 novelty of the selected 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 was included, despite its quality.

3.3.2.5 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, the inclusion/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 4.

3.3.2.6 Data analysis

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. They 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.

Data collection was 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 on data attribute 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 analysis was used. Iterating on the categories by removing, adding, and combining them, new categories and patterns emerged.

Qualitative data analysis was performed one attribute at a time. Each data point was copied to its own Google Sheet tab 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 with affinity diagramming is to visually arrange statements that are written on cards or notes. Next step is to group things that relate to each other 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 interpretations from the articles.

4 Results

The Multivocal Literature Review (MLR) performed on the definition of DX resulted in interesting results that are 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.

During the analysis, two different groups of researchers that study DX was found. A group of researchers in Brazil has to a large extent researched the DX in the context of Mobile Software Ecosystems (MSECO) (Fontão, Santos, Dias-Neto, et al., 2016; Fontão, Bonifácio, Santos, & Dias-Neto, 2018; Fontão, Dias-Neto, & Viana, 2017; Fontão, Ekwoge, Santos, & Dias-Neto, 2017; Fontão, Santos, & Dias-Neto, 2015). 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 (Graziotin, Fagerholm, Wang, & Abrahamsson, 2017a, 2017b, 2017c, 2018). To this group is also linked other studies related to the DX of IDEs.

In Table 4 the numbers of included and excluded sources is presented. The number of excluded sources is relatively high compared to the included sources.

	Scientific literature	Grey literature	
Included	40	19	59
Excluded	50	2	52
	90	21	111

Table 4: Number of sources analyzed for the MLR

4.1 The object or 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 RQ1. 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.

4.1.1 Object or entity 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. The reason behind this is most likely because of 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
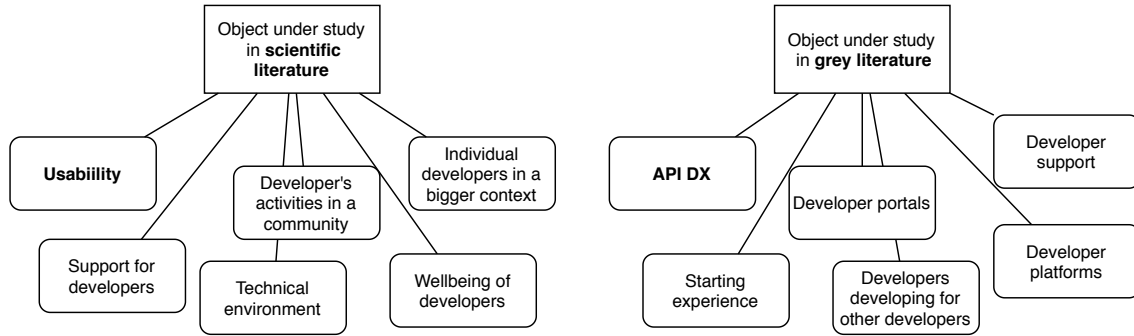
Figure 4: The object or entity of DX under study

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, well-being 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 is often 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, but also bigger as e.g. users of a specific software development platform.

Well-being of developers was discussed in multiple articles. Mainly this was a discussion topic of the group of researchers studying the happiness and unhappiness of developers (Graziotin et al., 2017a, 2017b, 2017c, 2018). 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 well-being of developers.

Individual developers in a bigger context was not a self-evident object under study. Mäenpää, Fagerholm, Munezero, Kilamo, Mikkonen, et al. (2017) (Connection between community context and the individual developer's experience) and Fagerholm and Pagels (2014) (Lean/Agile values) 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.

The above mentioned objects and entities under study show that they all have in common the "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 well-being of developers is seen important and different viewpoints on how this can be improved was found in the articles.

4.1.2   Object or entity 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 widely discussed topic in the industry. There has lately been discussions about a new term called *API Economy* Tan, Fan, Ghoneim, Hossain, and Dustdar (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 adaptability.

Developer portals are websites and documentation libraries, mostly of APIs, where information and examples aimed towards developers can be found. 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 counterintuitive to use. Popular software libraries and frameworks are putting effort into creating a good DX, that will benefit the developers using them.

Object or entity under study in scientific literature

Support for developers	Fontão, Dias-Neto, and Viana (2017) Fontão et al. (2016) Fontão et al. (2018)
Technical environment	Kuusinen, Petrie, Fagerholm, and Mikkonen (2016) Kuusinen (2016) Morales et al. (2019)
Developer's activities in a community	Fontão et al. (2015) Palviainen et al. (2015) Oran (2017) Linna et al. (2018) Claussen and Halbinger (2019)
Wellbeing of developers	Graziotin et al. (2018) Graziotin et al. (2017a) Fontão, Ekwoge, et al. (2017) Graziotin et al. (2017c) Graziotin et al. (2017b)
Individual developers in a bigger context	Mäenpää et al. (2017) Fagerholm and Pagels (2014)

Object or entity under study in grey literature

Starting experience	EveryDeveloper (n.d.)
API DX	Apigee (2019)
Developer portals	Rangel (n.d.) Sandoval (2017) De Simone (2015)
Developers developing for other developers	Jarman (2017) Heroku (2019) Vilrokx (2017) Baker (2017) Sanity (n.d.) Dhide (2017)
Developer support	moesif (n.d.) Skogberg (2019)
Development platforms	Magazine (2011) Riggins (2018)

Table 5: Object under study

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 recruiting developers to their platforms. 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.

4.1.3 Comparison between scientific literature and grey literature in the object or entity 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 is also focusing more on the feelings and moods of developers, while only a small part of the grey literature focused on this.

4.2 Research methods that have been used to study DX

A list of research methods used to study DX can be found in Table 7. This section addresses RQ2 From the nature of the research topic in the studies, it is apparent that most of the research methods are exploratory in nature.

4.2.1 Research methods used to study DX in scientific literature

Mostly the results of the studies were based on literature review and some qualitative methods like interviews, surveys, and diary studies. There were also some design science researches and case studies. Case studies and research designs are probably

Interviews
Survey
Literature review / Systematic mapping
Study of related approaches
Trainings / Experiments / Diary studies
Design research
Case study
Expert experience
Repository mining

Table 6: Different research methods used to study DX
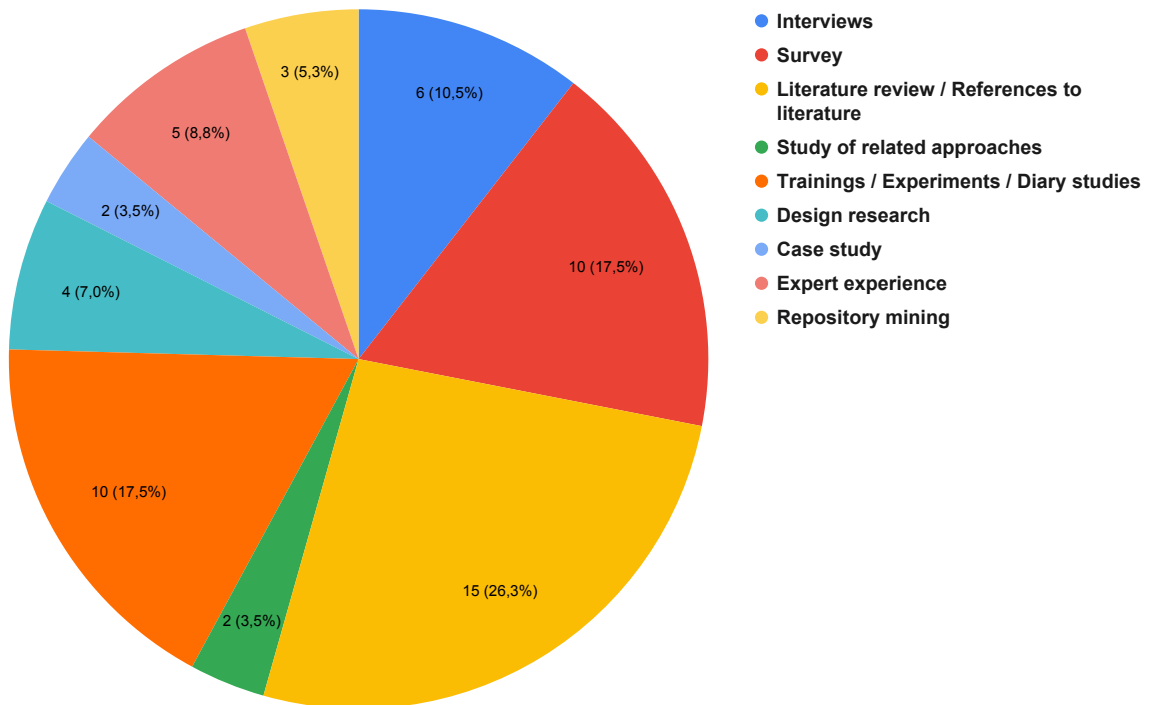

Figure 5: Research methods of Developer Experience in scientific literature

Research methods used to study DX in scientific literature

Interviews   Henriques, Lourenço, Amaral, and Goulão (2018)
                                    Palviainen et al. (2015)
                                    Murphy et al. (2018)
                                    Kärpänoja, Virtanen, Lehtonen, and Mikkonen
                                    (2016a)
                                    Zhang and Xie (2018)
                                    Ollis (2019)

Survey   Kuusinen et al. (2016)
                                    Graziotin et al. (2017c)
                                    Graziotin et al. (2017b)
                                    Graziotin et al. (2017a)
                                    Dong and Khandwala (2019)
                                    Kuusinen (2015)
                                    Fagerholm and Pagels (2014)
                                    Miranda et al. (2018)
                                    Kuusinen (2016)

Literature review / References   Fagerholm and Münch (2013)
to literature   Henriques et al. (2018)
                                    Fontão, Dias-Neto, and Viana (2017)
                                    Chatley (2019)
                                    Pinter, Lasso, and Fichtinger (2019)
                                    Mäenpää et al. (2017)
                                    Fagerholm and Pagels (2014)
                                    Fontão et al. (2016)
                                    Fontão et al. (2015)
                                    Myers and Stylos (2016)
                                    Ekwoge, Fontão, and Dias-Neto (2017)
                                    Romano, Scanniello, Fucci, Juristo, and Turhan
                                    (2018)
                                    Linna et al. (2018)
                                    de Lima Fontão, Dias-Neto, and Santos (2017)
                                    Morales et al. (2019)
                                    Oran (2017)

Study of related approaches   Fagerholm and Münch (2013)
                                    Henriques et al. (2018)

Trainings / Experiments / Diary studies	Nebeling, Zimmerli, and Norrie (2013) Fontão et al. (2018) Dong and Khandwala (2019) Miranda et al. (2018) Fontão et al. (2016) Ivo, Guerra, Porto, Choma, and Quiles (2018) de Lima Fontão et al. (2017) Nazário, Guerra, Bonifácio, and Pinto (n.d.) Zhang and Xie (2018) Silva et al. (n.d.)
Design research	Henriques et al. (2018) Pinter et al. (2019) Dong and Khandwala (2019) Oran (2017)
Case study	Fagerholm et al. (2014) Fagerholm et al. (2015)
Expert experience	Mäenpää et al. (2017) Myers and Stylos (2016) Macvean, Church, Daughtry, and Citro (2016) Kärpänoja, Virtanen, Lehtonen, and Mikkonen (2016b) Nazário et al. (n.d.)
Repository mining	Fontão, Ekwoge, et al. (2017) de Lima Fontão et al. (2017) Claussen and Halbinger (2019)

Research methods used to study DX in grey literature

Literature review / References to literature	De Simone (2015) Nazr (2018)

Expert experience	Jarman (2017)
	Baker (2017)
	Apigee (2019)
	Heroku (2019)
	Skogberg (2019)
	EveryDeveloper (n.d.)
	Dhide (2017)
	moesif (n.d.)
	Lange (2016)
	Riggins (2018)
	Rangel (n.d.)
	Magazine (2011)
	De Simone (2015)
	Vilrokx (2017)
	Nazr (2018)
	Sandoval (2017)

Table 7: Research methods

popular because they provide academics a way to collaborate with practitioners. This is common especially in software engineering, as the industry might be leading the way ahead the academic research.

Some data repository mining was also performed to find out patterns in developer behaviour. This type of research is interesting and had been enabled by the large developer communities that provide opportunities for data mining and analysis. Examples of these are StackOverflow and GitHub.

Experiments, trainings and observational studies were also a common research method. These were utilized in studies were the focus was on different objects of was on the usability of like the IDE or the programming language utilized control groups and experiments.

4.2.2 Research methods in used to study DX grey literature

In the grey literature the research methods are all expert experience reports where practitioners from the industry report their experiences and lessons from them on blogs and other websites. However, the grey literature articles were quite heavy in references, where the authors were used either someone else's writing or their own writing to base their statements.
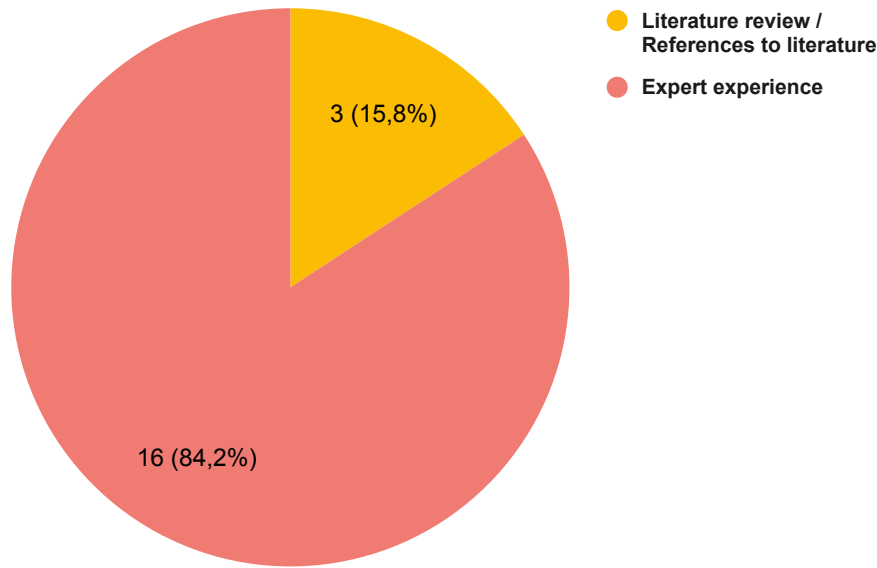
Figure 6: Research methods of Developer Experience in grey literature

4.3 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. This section answers RQ3. 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.
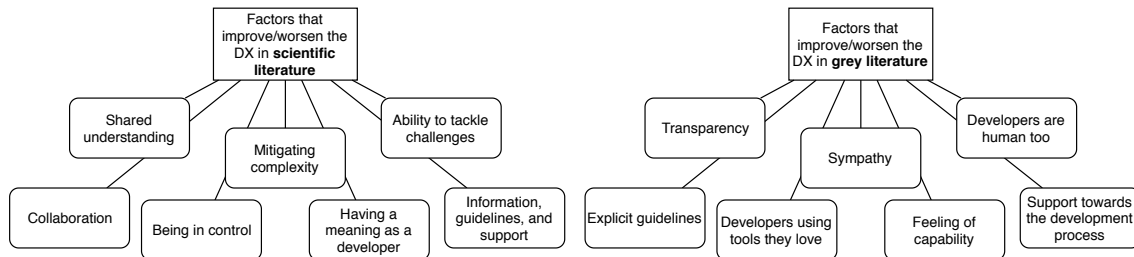

Figure 7: Factors that improve or worsen the DX

4.3.1 Factors that improve or 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.*

Factors that improve or worsen the DX in scientific literature

Shared understanding	Fagerholm et al. (2014)
	Fagerholm et al. (2015)
	Murphy et al. (2018)
	Fontão et al. (2016)
Collaboration	Palviainen et al. (2015)
	Mäenpää et al. (2017)
	Kärpänoja et al. (2016a)
	Romano et al. (2018)
	Ollis (2019)
	Oran (2017)
	Graziotin et al. (2017b)
	Linna et al. (2018)
Mitigating complexity	Fontão, Ekwoge, et al. (2017)
	Nebeling et al. (2013)
	Miranda et al. (2018)
Being in control	Kuusinen (2015)
	Silva et al. (n.d.)
	Myers and Stylos (2016)
	Macvean et al. (2016)
Ability to tackle challenges	Pinter et al. (2019)
	Ivo et al. (2018)
	Zhang and Xie (2018)
	Graziotin et al. (2018)
Having a meaning as a developer	Fagerholm and Pagels (2014)
	Fontão, Dias-Neto, and Viana (2017)
Information, guidelines, and support	de Lima Fontão et al. (2017)
	Myers and Stylos (2016)
	Macvean et al. (2016)
	Claussen and Halbinger (2019)
	Chatley (2019)
	Nazário et al. (n.d.)
	Henriques et al. (2018)
	Fontão et al. (2018)

Factors that improve or worsen the DX in grey literature

Transparency	Riggins (2018)
	Sandoval (2017)
	Baker (2017)

Explicit guidelines	Jarman (2017) Nazr (2018)
Developers using tools they love	Heroku (2019) Apigee (2019) stereobooster (2018) Dhide (2017) De Simone (2015)
Sympathy	Skogberg (2019)
Feeling of capability	EveryDeveloper (n.d.) Sanity (n.d.)
Developers are human too	moesif (n.d.) Rangel (n.d.)
Support towards the development process	Magazine (2011) Vilrokx (2017)

Table 8: Factors that improve or worsen the Developer Experience

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 an 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 (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.

4.3.2 Factors that improve or 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.

4.3.3 Comparison of scientific 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.

4.4   Explicit and 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 helps us to answer RQ4. 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.
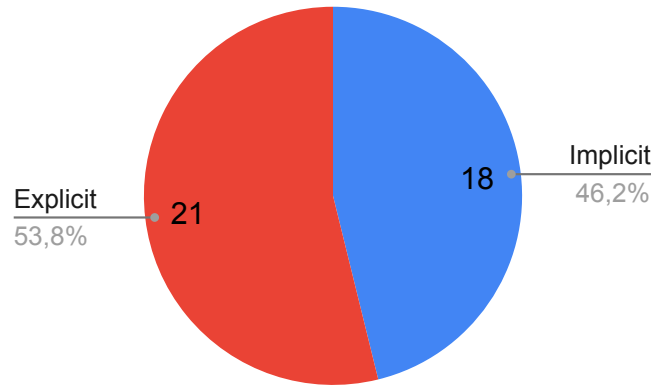

Figure 8: Implicit vs. explicit definition of Developer Experience in scientific literature

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 are things that can be taken for granted, and therefore do not require additional clarification or definition of what they are. However, DX might not yet be at that point yet and therefore it could be seen that it is required to give a definition of it.

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.
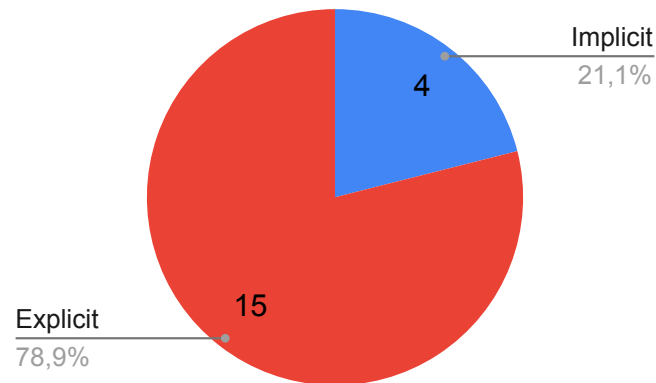
Figure 9: Implicit vs. explicit definition of Developer Experience in grey literature

Implicit or explicit definition of DX in scientific literature

Implicit definition	Nebeling et al. (2013)
	Henriques et al. (2018)
	Graziotin et al. (2017c)
	Graziotin et al. (2017b)
	Graziotin et al. (2017a)
	Graziotin et al. (2018)
	Chatley (2019)
	Murphy et al. (2018)
	Pinter et al. (2019)
	Dong and Khandwala (2019)
	Miranda et al. (2018)
	Fontão et al. (2015)
	Myers and Stylos (2016)
	Macvean et al. (2016)
	Ivo et al. (2018)
	Claussen and Halbinger (2019)
	Silva et al. (n.d.)
	Ollis (2019)

Explicit definition

Fagerholm and Münch (2013)
Kuusinen et al. (2016)
Fontão et al. (2018)
Palviainen et al. (2015)
Fontão, Dias-Neto, and Viana (2017)
Fagerholm et al. (2014)
Fagerholm et al. (2015)
Fontão, Ekwoge, et al. (2017)
Mäenpää et al. (2017)
Kuusinen (2015)
Fagerholm and Pagels (2014)
Fontão et al. (2016)
Kuusinen (2016)
Kärpänoja et al. (2016b)
Ekwoge et al. (2017)
Romano et al. (2018)
Linna et al. (2018)
de Lima Fontão et al. (2017)
Morales et al. (2019)
Oran (2017)
Nazário et al. (n.d.)
Zhang and Xie (2018)

Implicit or explicit definition of DX in grey literature

Implicit definition

Apigee (2019)
Heroku (2019)
moesif (n.d.)
Sanity (n.d.)

Explicit definition

Jarman (2017)
Baker (2017)
Skogberg (2019)
stereobooster (2018)
EveryDeveloper (n.d.)
Dhide (2017)
Lange (2016)
Riggins (2018)
Rangel (n.d.)
Magazine (2011)
De Simone (2015)
Vilrokx (2017)
Nazr (2018)
Sandoval (2017)

Table 9: Implicit or explicit definition of Developer Experience

4.5 Context of the studies

This section sets the ground to be able to answer RQ4. From the analysis of the material, there is a clear indication that there are different viewpoints and contexts to the research and study of DX. The different contexts that they studies are made in emerged from the analysis are listed in Table 10. The context were completely unknown before starting the MLR. One source address the topic of DX from multiple contexts and viewpoints. The context can be seen as the environment of where DX is studied in, but also the motivation why the paper or article has been written. In some cases the context is clear, e.g the API of a product, but sometimes it can be more subtle. Some cases the paper or article doesn't have any specific context, but focuses more on defining DX and in these cases the context can be seen as "giving a definition to DX". In software engineering the topics researched and studied can often be seen from multiple different viewpoints based on a plethora of different things. The same is true about DX.

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

Table 10: 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 without any specific viewpoint. 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
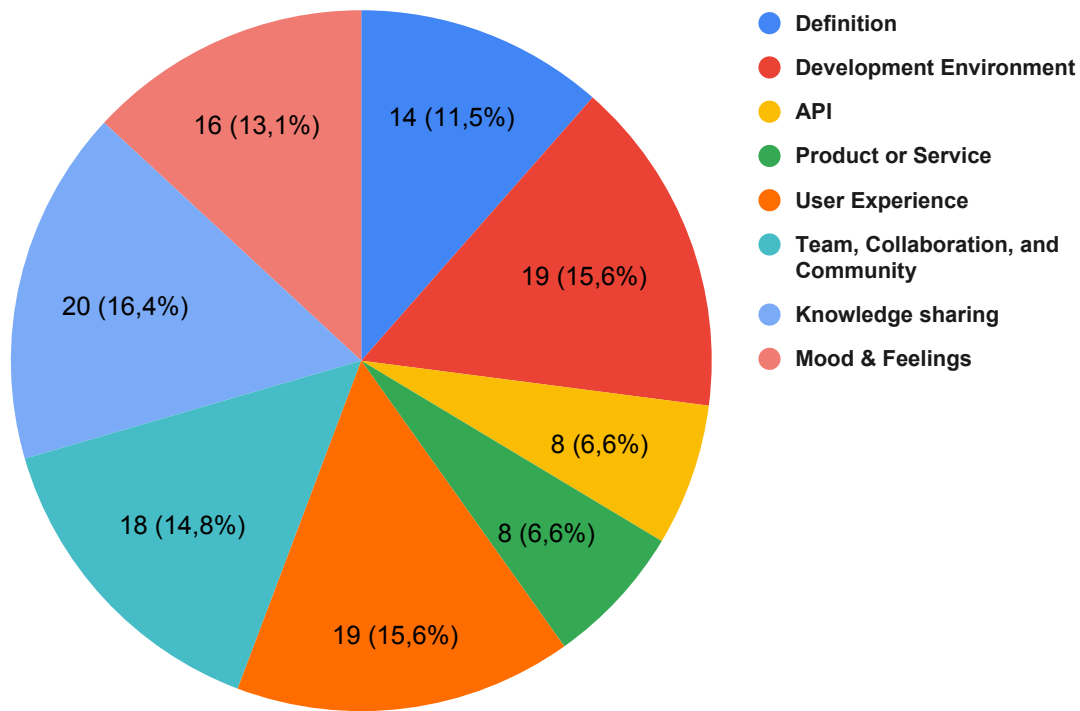
Figure 10: Context of Developer Experience in scientific literature

Context of the studies in scientific literature

Definition

Fagerholm and Münch (2013)
Kuusinen et al. (2016)
Fontão et al. (2018)
Palviainen et al. (2015)
Fontão, Dias-Neto, and Viana (2017)
Murphy et al. (2018)
Mäenpää et al. (2017)
Fagerholm and Pagels (2014)
Fontão et al. (2016)
Kuusinen (2016)
Kärpänoja et al. (2016b)
Ekwoge et al. (2017)
Romano et al. (2018)
Morales et al. (2019)

Development Environment	Fagerholm and Münch (2013) Kuusinen et al. (2016) Nebeling et al. (2013) Fontão et al. (2018) Palviainen et al. (2015) Fontão, Dias-Neto, and Viana (2017) Chatley (2019) Pinter et al. (2019) Dong and Khandwala (2019) Mäenpää et al. (2017) Kuusinen (2015) Fontão et al. (2016) Kuusinen (2016) Kärpänoja et al. (2016b) Ekwoge et al. (2017) Linna et al. (2018) Morales et al. (2019) Nazário et al. (n.d.) Zhang and Xie (2018) Ollis (2019)
API	Nebeling et al. (2013) Murphy et al. (2018) Pinter et al. (2019) Dong and Khandwala (2019) Myers and Stylos (2016) Macvean et al. (2016) Morales et al. (2019) Ollis (2019)
Product or service	Murphy et al. (2018) Pinter et al. (2019) Miranda et al. (2018) Fontão et al. (2016) Myers and Stylos (2016) Macvean et al. (2016) Kuusinen (2016) Claussen and Halbinger (2019)

User Experience	Fagerholm and Münch (2013)
	Nebeling et al. (2013)
	Henriques et al. (2018)
	Murphy et al. (2018)
	Pinter et al. (2019)
	Dong and Khandwala (2019)
	Kuusinen (2015)
	Miranda et al. (2018)
	Fontão et al. (2015)
	Myers and Stylos (2016)
	Macvean et al. (2016)
	Kuusinen (2016)
	Kärpänoja et al. (2016a)
	Ekwoge et al. (2017)
	Ivo et al. (2018)
	Morales et al. (2019)
	Claussen and Halbinger (2019)
	Nazário et al. (n.d.)
	Silva et al. (n.d.)
Team, Collaboration, & Community	Fagerholm and Münch (2013)
	Nebeling et al. (2013)
	Fontão et al. (2018)
	Palviainen et al. (2015)
	Fontão, Dias-Neto, and Viana (2017)
	Graziotin et al. (2018)
	Fagerholm et al. (2014)
	Fagerholm et al. (2015)
	Chatley (2019)
	Mäenpää et al. (2017)
	Fagerholm and Pagels (2014)
	Fontão et al. (2016)
	Kärpänoja et al. (2016a)
	Claussen and Halbinger (2019)
	Oran (2017)
	Nazário et al. (n.d.)
	Zhang and Xie (2018)
	Ollis (2019)

Knowledge sharing	Fagerholm and Münch (2013)
	Fontão et al. (2018)
	Palviainen et al. (2015)
	Fontão, Dias-Neto, and Viana (2017)
	Fagerholm et al. (2014)
	Fagerholm et al. (2015)
	Chatley (2019)
	Fontão, Ekwoge, et al. (2017)
	Murphy et al. (2018)
	Mäenpää et al. (2017)
	Fagerholm and Pagels (2014)
	Fontão et al. (2016)
	Myers and Stylos (2016)
	Macvean et al. (2016)
	de Lima Fontão et al. (2017)
	Claussen and Halbinger (2019)
	Oran (2017)
	Nazário et al. (n.d.)
	Zhang and Xie (2018)
	Ollis (2019)
Mood & Feelings	Fagerholm and Münch (2013)
	Graziotin et al. (2017c)
	Graziotin et al. (2017b)
	Graziotin et al. (2017a)
	Graziotin et al. (2018)
	Fagerholm et al. (2014)
	Fagerholm et al. (2015)
	Fontão, Ekwoge, et al. (2017)
	Fontão et al. (2015)
	Kuusinen (2016)
	Kärpänoja et al. (2016a)
	Ekwoge et al. (2017)
	Morales et al. (2019)
	Nazário et al. (n.d.)
	Zhang and Xie (2018)
	Ollis (2019)

Context of the studies in grey literature

Definition	stereobooster (2018)
	Magazine (2011)
	De Simone (2015)
	EveryDeveloper (n.d.)
	Nazr (2018)

Development Environment	stereobooster (2018) Riggins (2018)
API	Jarman (2017) Apigee (2019) stereobooster (2018) EveryDeveloper (n.d.) moesif (n.d.) Lange (2016) Rangel (n.d.) Magazine (2011) De Simone (2015) Vilrokx (2017) Nazr (2018) Sanity (n.d.) Sandoval (2017)
Product or service	Baker (2017) Apigee (2019) Heroku (2019) EveryDeveloper (n.d.) Dhide (2017) moesif (n.d.) Lange (2016) Rangel (n.d.) Magazine (2011) De Simone (2015) Vilrokx (2017) Nazr (2018) Sanity (n.d.)
User Experience	Jarman (2017) Baker (2017) Apigee (2019) stereobooster (2018) Dhide (2017) moesif (n.d.) Lange (2016) Rangel (n.d.) Magazine (2011) De Simone (2015) Vilrokx (2017) Nazr (2018) Sandoval (2017)

Team, Collaboration, & Community	Heroku (2019)
	Skogberg (2019)
	stereobooster (2018)
	moesif (n.d.)
	Riggins (2018)
	Magazine (2011)
	Vilrokx (2017)
	Sanity (n.d.)
Knowledge sharing	Jarman (2017)
	Apigee (2019)
	Skogberg (2019)
	Dhide (2017)
	Riggins (2018)
	Magazine (2011)
	De Simone (2015)
	Vilrokx (2017)
	Nazr (2018)
	Sandoval (2017)
Mood & Feelings	Skogberg (2019)
	EveryDeveloper (n.d.)
	Rangel (n.d.)
	De Simone (2015)

Table 11: Different contexts of Developer Experience in the literature

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
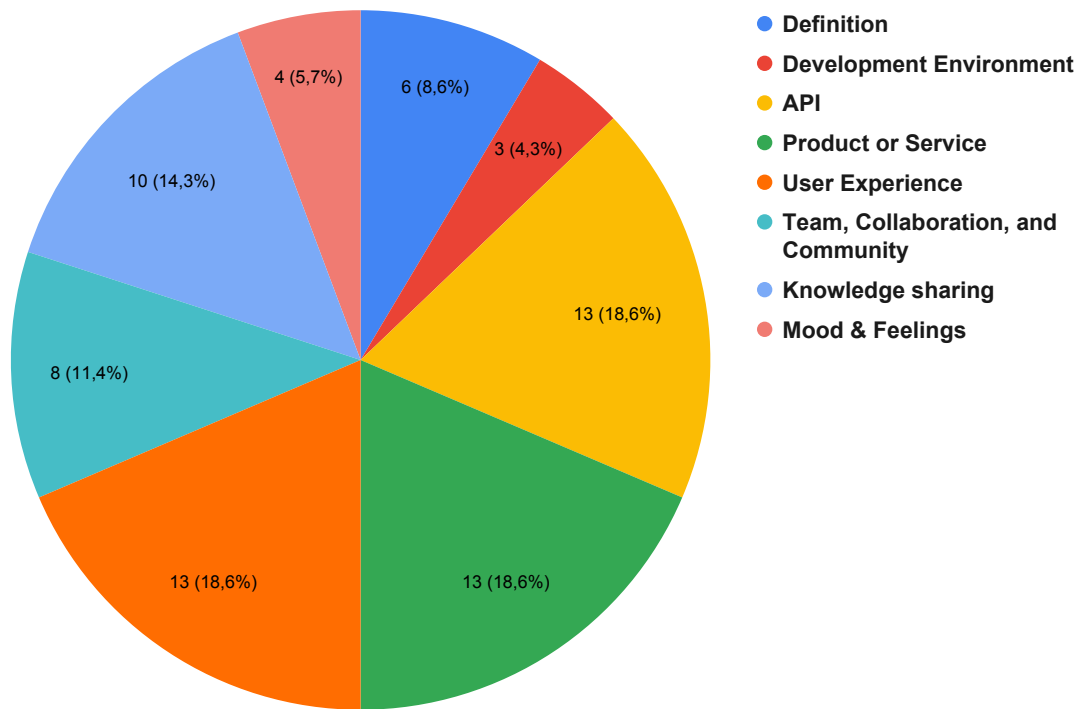
Figure 11: Context of Developer Experience in grey literature

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 (Baker, 2017), 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.

4.6   Categories of definitions of DX

Finally to answer RQ5, an attempt to understand the definition of DX was made. This was based on the summaries of 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. The definitions of DX was also a data point in the data collection form, and each article's short definition of DX was interpreted. The subsequent sections analyze the definitions on both the scientific and grey literature, and categorizes the definitions into bigger contexts.
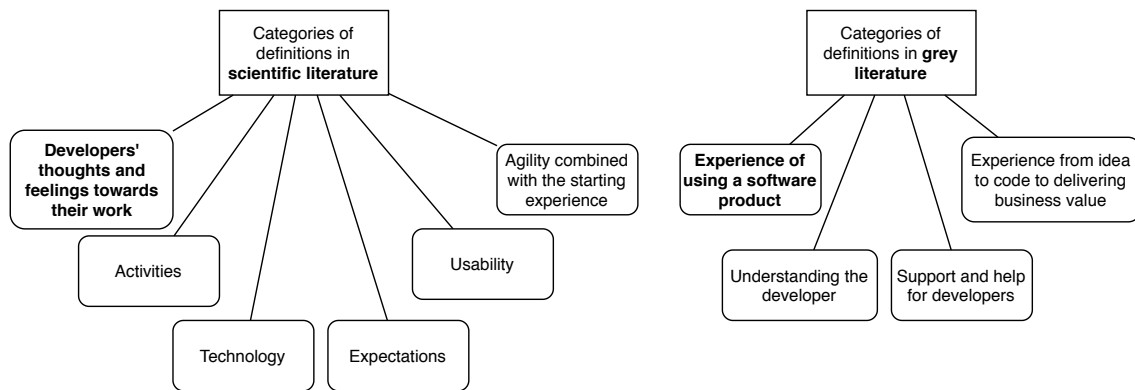

Figure 12: Categories of definitions of DX

4.6.1   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 (February 2020) 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 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 et al. (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 et al. (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"* (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.

4.6.2   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.

> "Developer Experience (DX) is the equivalent of User Experience (UX) when it comes to a developer" (Jarman, 2017)

The category of *understanding the developer* consisted of the following definition:

> "DX design is about understanding the context of use, understanding what developers need to complete their tasks, underlying technology, integration points, and focusing on how developers feel while using a product or services." (Dhide, 2017).

There seems to be a consensus that understanding the developer is important when creating a good developer experience.

Experience from idea to code to delivering business value was a category that was created from a quote in Riggins (2018) where they write about a talk on platforms and DX of them. This category included also viewpoints on the starting experience, and about how developers understand what is possible and what is not.

Support and help for developers was the final category of definitions in grey literature. This category had definitions that focused on making the developers life as easy as possible, and about creating a seamless experience for developers. Improving and optimizing how developer get their work done allows to create a better DX. This category emerged from the articles that discussed about third party library, framework, and tool authors (developers) that have other developers as their users or customers.

> "... Developer Experience (DX) — which is all about using User Experience (UX) techniques to make life easier for third-party developers calling your public APIs ..." (Lange, 2016)

4.6.3 Difference between the definition of DX in scientific and grey literature

There can be seen both differences and similarities between the definitions of DX in the scientific and grey literature. They both define DX as the developer's experience of developing software and interacting with software development artifacts. They both also define providing a good DX is related to understanding the developer and showing sympathy towards them. Their difference is that in grey literature the definitions lean more toward the usability of products and services, while the definitions in scientific literature take a more holistic approach of defining DX.

4.7 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 a wide variety of different viewpoints. The different data points that were collected showed interesting results about the different aspects of DX, both in scientific and grey literature. There were both similarities and differences between them.

One main finding from the MLR, that applies for both scientific and grey literature, is that in software engineering in general there has happened a change in the empathy and sympathy shown and expressed towards developers. Almost all articles analyzed were discussing about DX in a humane and compassionate tone. 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. Kauppinen et al. (2009) discuss about customer value creation. Even though the study is in the field of Requirements Engineering (RE), the similarities can be seen. They state that practitioners have previously focused more on feature creation i.e. a *"inside-out"* approach of looking at RE where the features and other development ideas came from the development team themselves. However, they see that better customer value can be created by taking an *"outside-in"* approach, where the customer is understood, and that the features and development ideas come from them.

4.8 Experience influencers

Fagerholm (2015) note that there are experience influencers in the formation of DX. This section discusses the most notable experience influencers that were found in this study both in the background literature check and the MLR. Each subsection discusses a particular experience influencer in no particular order.

4.8.1 Selection of tools

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

Kuusinen et al. (2016) studied Integrated Development Environment (IDE), and how they are connected with state of flow, intrinsic motivation, and user experience. Their findings reveal that if the developers have a high perception of choice, they 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 DX with it, as e.g. their frustration with the tool will be more common.

There has been a study on the Developer Experience of IDEs (Kuusinen, 2015). However, the study concentrated on the UX of the selected IDE that was studied. When selecting an IDE it is also important to consider what the other developers in the team or organization is using or what other would prefer to use.

There can be situations when two different developers use a different IDE, and therefore also the experience can be completely different between them. At the most extreme the 2 IDEs are not compatible with each other as their files related to the project are different. An example of this is Eclipse and IntelliJ IDEA as Java IDEs.

In a study of IDEs (Kuusinen, 2015), the survey about IDEs in the study produced answers about the IDE that were most pragmatic, but not hedonic. This could show that most of the developers are practical, and not feeling based. Capretz (2003) studied the personalities in software engineering and found out that the majority of software developers are of type ISTJ (Introversion, Sensing, Thinking, Judgement) on the Myers-Briggs Type indicator. This might also be a reason why DX has not received that much attention yet, as big part of people in software engineering are *"Introverts"*. This doesn't mean that they do not have feelings, but that they keep them more to themselves or that they are not that well aware of their feelings. Software engineers might also think, even if they might not be, that they base their reasoning more on logic than feelings. As Developer Experience is focusing on the feelings and subjective opinions about things, it might be a difficult topic to research.

4.8.2   Organization and project onboarding

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

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

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

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

4.8.3   Motivation

According to Kuusinen et al. (2016) 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) studied how IM, state of flow, and UX relate to the DX of software developers using IDE. They found out that these properties were significant predictors of a good DX in the IDE.

4.8.4 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.

4.8.5 Happiness of developers

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

4.8.6 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.

4.8.7 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 et al., 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.

5 Discussion

Developer experience is an umbrella term that captures a big part how developers feel and perceive their work. Current definition of DX in the industry is largely based on practitioners own experiences. In academic writings and papers, the topic of DX is studied relatively little, and the existing articles are not well known by the research community.

Surprisingly the most used research method in scientific articles was literature reviews and systematic mappings. Even if the topic is novel and there is not much studies on it, the studies are basing their findings on previous work. One explanation to this could that DX is an umbrella term (hypernym) of multiple different hyponyms, e.g. UX, usability, and aspects of psychology. Based on the research methods, there can be seen that trainings, experiments, and diary studies were utilized a lot. This shows that the academics were closely collaborating with practitioners from the industry and students at universities.

5.1 Answering the research questions

The research problem (What is the definition and aspects of Developer Experience, and how do they differ between scientific literature and literature written by practitioners?) prompted to perform a Multivocal literature review where both scientific and grey literature was studied to gain a better understanding of DX.

RQ1 (What objects and entities have been studied with respect to developer experience?) focused on finding about the different objects or entities of DX that are studied. The most obvious object under study in scientific literature was the technical artifacts that the developer interacts with in their everyday work tasks. Also, the most common artifact under study was programming languages and IDEs i.e. development tools. Other objects under study were the processes and methods used, the work environment, and developer's mood. In grey literature the most studied object of DX were APIs and their usability and other development tools.

RQ2 (What methods have been used to study developer experience?) questioned the research methods used in the research of DX. The most apparent research method was literature reviews in scientific articles, and expert experience reports in grey literature. In scientific literature empirical studies were also common along with surveys.

Answering RQ3 (What is known about factors that improve or worsen developer experience?) was possible based on the results of the analysis of the data collected. There were multiple different factors that improve or worsen the DX. In scientific literature were multiple factors, e.g. shared understanding, collaboration, and mitigating complexity, that seemed to improve DX. Common with all these factors were that almost all of them were non-technical, and more social factors, if following the division into technical and social factors by Fagerholm (2015) presented in Figure 1.

RQ4 (From what contexts is developer experience looked at?) was presented because it was noted that different articles all take their own viewpoint and context

from which DX is looked at. Scientific articles are focused more on looking at DX from the context the development environment, the definition DX, Knowledge sharing, and Team, Collaboration and Community. However grey literature focused mostly on DX from the context of APIs, products and services, and UX.

RQ5 (What are the definitions given to developer experience?) asked about the different definitions of DX found in the literature. The most notable and recurring concise definition is provided by Fagerholm and Münch (2013) as *"the developers feelings and perceptions about their work"*. The definitions of DX in the grey literature to a large degree fall under this same definition given by Fagerholm and Münch (2013). However, there is the apparent business aspect visible in the grey literature where the point of view is gaining business value by improving DX of products and services.

5.2 Validity of the results

The selection of the search strategy might have caused some relevant search results to have been excluded from the results. There were attempts to try to mitigate this risk, but it could not be assured that this was the case. Because of the novelty of the topic, there are most likely a large amount of articles and writings that discuss about the topic of DX implicitly. However, as a researcher it is almost impossible to try to also include these articles into the research and study.

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.

Other noteworthy limitations to the research was the search process and decisions made to finding the sources of the MLR. These decisions include the decision to only include 2 first pages of Google search results, collecting scientific articles from 4 databases were used to find scientific articles, and finally, only one round of snowballing was performed on the scientific sources. They all are limitations to the search process, and might have affected the search results.

5.3   Future research

This study took an exploratory approach into understanding the phenomenon of DX, especially the current state of the literature in it.

One possible future research approach could be to find possible cause-effect relations regarding to DX. One example of this could be to find out the relation between practicing good developer experience and the effect of that on software projects. One question to answer could be e.g. *"What are the aspects of Developer Experience that are utilized in practice and have potential of being replicable in different teams of a software consulting company?".* Unanswered questions in this approach are *"what is developer experience?", what are the aspects of developer experience? "what does utilizing in practice mean?, "what does having potential mean?" "what does replicable mean?".* To be able to study research problems like this there needs to be more ground work done in the field of study of DX.

In conclusion, future research could focus on any aspect of DX in practice. How are practitioners considering DX? What could the benefits of focusing on DX be? As it was found out in this thesis, there could be huge potential in understanding DX and how it affect the practice of software development. We have seen the evolution from Human Computer Interaction (HCI) to User Experience (UX). It will be interesting to see to what extent DX is taken.

6 Conclusions

From the research and foundations of Human Computer Interaction (HCI), Usability, and User Experience (UX) has a relatively new concept of Developer Experience (DX) emerged. While UX considers an user and their experience of using a service or product, DX considers the developer's experience of acting as an user of software development artefacts while developing software.

Traditionally developers have been seen as an asset or resource in the software development practice, where they are the labour and work force while developing and creating software products and services. The developer's opinions, feelings, and voice have not been heard well enough.

The concept of DX can be seen as a revolution to this. First, this means that the voice of the developer is being heard. Second, the developers are also being understood. DX is a way to measure otherwise difficult, complicated, and often abstract things. It is about understanding and showing sympathy towards the developer. With this understanding the environment that developers work in can be improved, including both the social and technical factors.

References

Apigee. (2019). Great developer experiences and the people who make them. Retrieved September 10, 2019, from https://medium.com/apis-and-digital-transformation/great-developer-experiences-and-the-people-who-make-them-b97b544caba9

Baker, J. (2017). Developer experience (dx) - devs are people too. Retrieved September 10, 2019, from https://hackernoon.com/developer-experience-dx-devs-are-people-too-6590d6577afe

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 Simone, S. (2015). What is api developer experience and why it matters. Retrieved September 10, 2019, from https://www.infoq.com/news/2015/10/api-developer-experience/

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.

Dhide, R. (2017). Building the developer experience (dx) from the ground up. Retrieved September 10, 2019, from https://blog.argoproj.io/building-the-developer-experience-dx-from-the-ground-up-8254d50457f5

DIS, I. (2009). 9241-210: 2010. ergonomics of human system interaction-part 210: Human-centred design for interactive systems. *International Standardization Organization (ISO). Switzerland.*

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., 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.

EveryDeveloper. (n.d.). What is developer experience? - everydeveloper. Retrieved September 10, 2019, from https://everydeveloper.com/developer-experience/

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.

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., 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., Dias-Neto, A., & 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., Ekwoge, O. 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., 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.

Heroku. (2019). Developer experience (dx) | heroku. Retrieved September 10, 2019, from https://www.heroku.com/dx

HPI Software Architecture Group. (2019). Programming experience 2019 (px/19) workshop. Retrieved November 22, 2019, from http://programming-experience.org/px19/

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. (2016a). Exploring peopleware in continuous delivery. In *Proceedings of the scientific workshop proceedings of xp2016* (p. 13). ACM.

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

Kauppinen, M., Savolainen, J., Lehtola, L., Komssi, M., Tohonen, H., & Davis, A. (2009). From feature development to customer value creation. In *2009 17th ieee international requirements engineering conference* (pp. 275–280). IEEE.

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. (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. (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.

Lange, K. (2016). 4 must-read articles on developer experience. Retrieved September 10, 2019, from https://www.kennethlange.com/4-must-read-articles-on-developer-experience-dx/

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.

Linna, J., Keskinen, T., Sarangi, A. K., Turunen, M., Hakulinen, J., & Dash, S. (2018). Open service innovation ecosystem for public transportation. In *Proceedings of the 25th its world congress.*

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.*

Magazine, U. (2011). Effective developer experience (dx) | ux magazine. Retrieved September 10, 2019, from https://uxmag.com/articles/effective-developer-experience

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.

Mirnig, A. G., Meschtscherjakov, A., Wurhofer, D., Meneweger, T., & Tscheligi, M. (2015). A formal analysis of the iso 9241-210 definition of user experience. In *Proceedings of the 33rd annual acm conference extended abstracts on human factors in computing systems* (pp. 437–450). ACM.

moesif. (n.d.). Api developer experience (dx) resources. Retrieved September 10, 2019, from https://www.moesif.com/blog/api-guide/api-developer-experience/

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.

Nazr, S. (2018). What exactly is developer experience? - apimatic. Retrieved September 10, 2019, from https://blog.apimatic.io/what-exactly-is-developer-experience-1646b813df14

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.

Nylund, A. (2020). A multivocal literature review on developer experience. figshare. Retrieved from https://figshare.com/articles/A_multivocal_literature_review_on_developer_experience/11873790/1

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

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

Rangel, F. (n.d.). Apis for humans: The rise of developer experience. Retrieved September 10, 2019, from https://www.hellosign.com/blog/the-rise-of-developer-experience

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

Riggins, J. (2018). Workflows for the new developer experience. Retrieved September 10, 2019, from https://thenewstack.io/workflows-for-the-new-developer-experience/

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.

Sandoval, K. (2017). 4 apis doing developer experience really well | nordic apis |. Retrieved September 10, 2019, from https://nordicapis.com/4-apis-doing-developer-experience-really-well/

Sanity. (n.d.). Developer experience - sanity.io. Retrieved September 10, 2019, from https://www.sanity.io/developer-experience

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.

Skogberg, A. (2019). Contributing to a great developer experience (dx) as a designer. Retrieved September 10, 2019, from https://uxdesign.cc/contributing-great-developer-experience-designer-e1f497b0fb4

stereobooster. (2018). Developer experience. how i missed it before? Retrieved September 10, 2019, from https://dev.to/stereobooster/developer-experience-how-i-missed-it-before-47go

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.

Vilrokx, M. (2017). Developer experience: What and why – the appslab. Retrieved September 10, 2019, from http://theappslab.com/2017/04/04/developer-experience-what-and-why/

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

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 et al. (2018)   Improving the Developer Experience with a low-code process modelling language

5. Fontão et al. (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. 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.	Fontão, Ekwoge, et al. (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 et al. (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.	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 et al. (2016)	MSECO-DEV: Application Development Process in Mobile Software Ecosystems
24.	Fontão et al. (2015)	Research Opportunities for Mobile Software Ecosystems
25.	Myers and Stylos (2016)	Improving API usability
26.	Macvean et al. (2016)	API Usability at Scale
27.	Kuusinen (2016)	Are Software Developers Just Users of Development Tools? Assessing Developer Experience of a Graphical User Interface Designer
28.	Kärpänoja et al. (2016b)	Exploring Peopleware in Continuous Delivery
29.	Ekwoge et al. (2017)	Tester Experience: Concept, Issues and Definition
30.	Romano et al. (2018)	The effect of noise on software engineers' performance

31.	Linna et al. (2018)	Open service innovation ecosystem for public transportation
32.	Ivo et al. (2018)	An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms
33.	de Lima Fontão et al. (2017)	Towards a Guideline-Based Approach to Govern Developers in Mobile Software Ecosystems
34.	Morales et al. (2019)	Programmer eXperience: A Systematic Literature Review
35.	Claussen and Halbinger (2019)	The Role of Pre-Innovation Platform Activity for Diffusion Success: Evidence from Consumer Innovations on a 3D Printing Platform
36.	Oran (2017)	A Set of Artifacts and Models to Support Requirements Communication Based on Perspectives
37.	Nazário et al. (n.d.)	Detecting and Reporting Object-Relational Mapping Problems: An Industrial Report
38.	Zhang and Xie (2018)	Toward Understanding IoT Developers in Chinese Startups
39.	Silva et al. (n.d.)	Comparing the Usability of two Multi-Agents Systems DSLs: SEA_ML++ and DSML4MAS Study Design
40.	Ollis (2019)	Helping developers to help each other: a technique to facilitate understanding among professional software developers

Table A1: Scientific articles

1.	Jarman (2017)	https://hackernoon.com/the-best-practices-for-a-great-developer-experience-dx-9036834382b0

71

2. Baker (2017) https://hackernoon.com/developer-experience-dx-devs-are-people-too-6590d6577afe

3. Apigee (2019) https://medium.com/apis-and-digital-transformation/great-developer-experiences-and-the-people-who-make-them-b97b544caba9

4. Heroku (2019) https://www.heroku.com/dx

5. Skogberg (2019) https://uxdesign.cc/contributing-great-developer-experience-designer-e1f497b0fb4

6. stereobooster (2018) https://dev.to/stereobooster/developer-experience-how-i-missed-it-before-47go

7. EveryDeveloper (n.d.) https://everydeveloper.com/developer-experience/

8. Dhide (2017) https://blog.argoproj.io/building-the-developer-experience-dx-from-the-ground-up-8254d50457f5

9. moesif (n.d.) https://www.moesif.com/blog/api-guide/api-developer-experience/

10. Lange (2016) https://www.kennethlange.com/4-must-read-articles-on-developer-experience-dx/

11. Riggins (2018) https://thenewstack.io/workflows-for-the-new-developer-experience/

12. Rangel (n.d.) https://www.hellosign.com/blog/the-rise-of-developer-experience

13. Magazine (2011) https://uxmag.com/articles/effective-developer-experience

14. De Simone (2015) https://www.infoq.com/news/2015/10/api-developer-experience/

15. Vilrokx (2017) http://theappslab.com/2017/04/04/developer-experience-what-and-why/

16. Nazr (2018) https://blog.apimatic.io/what-exactly-is-developer-experience-1646b813df14

17. Sanity (n.d.) https://www.sanity.io/developer-experience

| 18. | Sandoval (2017) | https://nordicapis.com/4-apis-doing-developer-experience-really-well/ |

Table A2: Grey literature