Developer Experience

Anders Nylund

School of Science

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

Espoo TBA

Supervisor

Prof. Pirjo Professor

Advisor

Dr Alan Advisor

Aalto University
School of Science

Copyright © 2019 Anders Nylund

Aalto University
School of Science

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

Author Anders Nylund		
Title Developer Experience		
Degree programme Computer, Communication and Information Sciences		
Major Software and Service Engineering		**Code of major** SCI3043
Supervisor Prof. Pirjo Professor		
Advisor Dr Alan Advisor		
Date TBA	**Number of pages** 36	**Language** English

Abstract
The abstract in english

Keywords Developer Experience, Software Projects

Aalto University
School of Science

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

Författare Anders Nylund		
Titel Developer Experience		
Utbildningsprogram Computer, Communication and Information Sciences		
Huvudämne Software and Service Engineering		**Huvudämnets kod** SCI3043
Övervakare Prof. Pirjo Professori		
Handledare TkD Alan Advisor		
Datum TBA	**Sidantal** 36	**Språk** Engelska

Sammandrag
Sammandrag på svenska

Nyckelord Nyckelord på svenska, temperatur

Preface

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

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

Otaniemi, Date to te announced

Anders Nylund

Contents

Thesis dictionary

DX	Developer Experience
UX	User Experience
IDE	Integrated Development Environment
IM	Intrinsic Motivation
EM	Extrinsic Motivation
HCI	Human Computer Interaction
API	Application Programming Interface
SLR	Systematic Literature Review
MLR	Multivocal Literature Review
GL	Grey Literature
SE	Software Engineering
MSECO	Mobile Software Ecosystem

1 Introduction

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

Developing and creating software is a social activity that requires both technical and social skills from the developers. Deep technical skills and understanding is required to be able to implement the wanted artifact or end product. However software engineering is a highly social activity, and therefore it has been noted that human factors are the most important when regarding software development performance [2].

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

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

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

In this study we will focus on understanding what DX is and make an attempt to map out good concrete practices on how a software company can improve the DX in their projects.

1.1 Motivation

At the time of writing, a quick search with the keyword *"Developer Experience"* on google.com gives as a result mostly articles on how framework and library authors should consider their user's (developer's) experience with using the product (tool, library, framework). Also, performing searches with the same *"Developer Experience"* keyword on known libraries of conference papers like Google Scholar and IEEExplore, the content and topic of the results vary much. This shows that there might not be a common and well known definition of what DX is.

In some research the term *Developer Experience* with the abbreviation of *DE*^{x} is used [5], in some other research the term *Programmer eXperience* and abbreviation *PX* is used [12], and finally maybe the most common abbreviation is *DX*. This shows that there is still some ambiguity to the terms and definitions in academic research. Additionally, most results when searching with the term *Developer Experience* gives results about the experience and knowledge level of a developer in e.g. terms of years working in the field of software development or amount of contribution, and not the hedonic and pragmatic experience of participating in development work.

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

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

1.2 Research problem and questions

Easterbrook [3] encourage practitioners to document and reason the selection process of the research problem and questions, the philosophical stance, and the selected research methods e.g the research protocol. They encourage this because other researchers can then understand and interpret the study and possibly replicate the study.

During this study the research problem and questions have evolved and been modified while more understanding and knowledge about the research topic has been created and accumulated. The starting point of the study was to understand how DX is linked to software project outcomes. However, this was noted to be too vague, difficult to measure and difficult to research. The current state and understanding of DX does not allow to research correlation and causality of DX to software project outcomes as defined in [3].

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

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

To analyze the research questions, the categorization, classification, and guidelines from [3] will be used.

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

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

Table 1: The research questions

out the absence of definitions. This question

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

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

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

1.3 Scope and focus

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

1.4 Structure of the thesis

This will be finalized later

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

2 Background

A software project is a project where a group of people share a common goal what can for example be to create a product or service. In a software project there is a developer or multiple developers that have the responsibility of implementing the technical product itself. The developers are the ones writing the executable source code for the program or service, so that it can by it's functions and features achieve the requirements set to it.

Developer Experience and its related terms have been studied and researched relatively little at the moment of writing. A literature review of the term *"Programmer Experience"* studied 73 articles that matched their search criteria [12]. 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.

A doctoral thesis titled "Software Developer Experience: Case Studies in Lean-Agile and Open Source Environments" in 2015 coined the term Developer Experience.

Developer experience can be divided into three different sub areas – cognitive (How developers perceive the development infrastructure), affective (How developers feel about their work), and conative (How developers see the value of their contribution) [5]. In a study it was also concluded that the cognitive part of DX is also addressed via intention and affect [11].

2.1 Programmer Experience

Programmer Experience (PX) can be defined as *The result of the intrinsic motivations and perceptions of programmers about the use of development artifacts* [12]. A programmer can be seen as person who gives exact instructions on how a program should behave and function. PX is based on the study mainly related to the programming environment, but also programming codes and Application Programming Interfaces (API).

2.2 Developer Experience

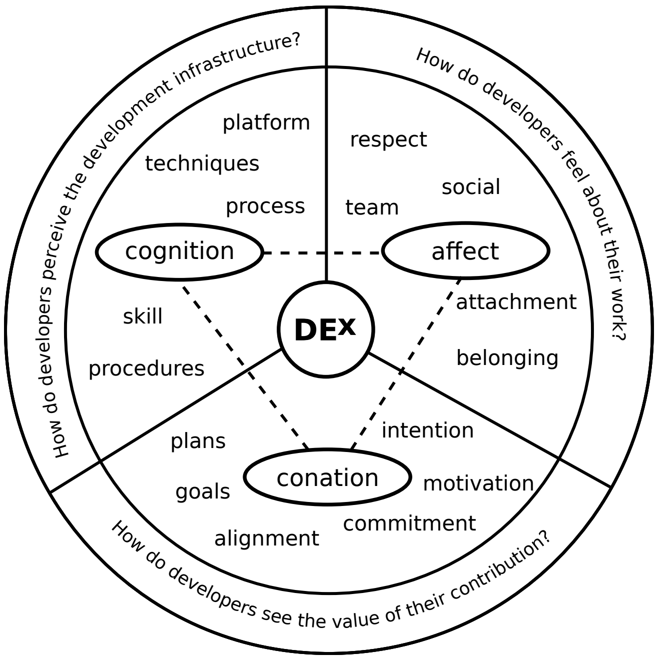
A developer is a person with a bigger responsibility than a programmer. If a programmer is following instructions, requirements, and guidelines, the developer is also finding out what the instructions, requirements and guidelines should be (find other source than https://devskiller.com/programmer-vs-developer/). Therefore DX is also considering more of the surrounding context than what PX is considering.

Developer Experience (DX) is a bigger construct than PX. DX includes also the motivation of developers, and not only the artefacts like the programming environments [12]. 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

The Developer Experience can be divided into 2 different environments, a social and a technical environment [4]. This thesis might focus more on the technical environment.

The developer experience 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 [4]. The research in this thesis will use a longer time-frame of developer experience.

Figure 1: Conceptual framework of Developer Experience [5]


How do developers perceive the development infrastructure?
How do developers feel about their work?
How do developers see the value of their contribution?
platform
techniques
process
cognition
skill
procedures
respect
social
team
affect
attachment
belonging
DEX
plans
goals
alignment
conation
intention
motivation
commitment

2.3 Intrinsic Motivation

Intrinsic Motivation (IM) is the motivation that is enabled by someone enjoying their own work, i.e. the motivation is originating from the work itself. Extrinsic Motivation (EM) is motivation that stems from the outcomes of the work performed [11] (Self-determination theory. Handbook of theories of social psychology).

2.4 Performance Alignment Work

2.5 Happiness of developers

Happiness of developers have been reported have direct consequences to the themselves, process and the product [8]

2.6 Selection of tools

Perceived choice is a perception of that the choice has already been made [11]. Selecting tools in software development projects is in a crucial role, as it can significantly improve the Developer Experience in software projects.

One study of Integrated Development Environment (IDE), and how it is connected with state of flow, intrinsic motivation, and user experience reveal that if the developers have a high perception of choice, the also are overall more satisfied with the tools [11]. They also concluded that if the selected tools are selected without their input, (they perceive it chosen already), the developers will have a worse developer experience with it, as e.g. their frustration with the tool will be more common.

There has been a study on the Developer Experience of IDEs [10]. 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 [10], the survey in the study produced answers that were most pragmatic, but not hedonic. This could show that most of the developers are practical, and not feeling based. This has also been proven [1]. This might also be a reason why Developer Experience has not gotten that much attention yet, as big part of people in software engineering are *"Introverts"*. Software engineers are also more logical thinkers than feeling based. As Developer Experience is focusing on the feelings and subjective opinions about things, it might be a difficult topic to research.

2.7 Flow state

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.

[13] studied how an IDE worked in a collaborative environment and it's developer experience.

2.8 Team, community, and collaboration

Entering an ecosystem: The hybrid OSS landscape from a developer perspective

2.9 User Starting Experience

0 to 200 or *Time to Hello World*

3 Research methods

During the study, the research problem and the research questions have evolved as the understanding and comprehension of the problem has improved. Because of this the planned research methods and the goals with these have also evolved and been reshaped when new information have been acquired.

From the introduction and motivation in sections 1 and 1.1, it was determined that DX is a novel and abstract construct. The concepts and definitions of it are multiple, and vary both in research and in industry.

Easterbrook [3] define a set of guidelines for empirical research in software engineering. They argue that selecting a clear research question, an explicit philosophical stance, and an appropriate research method are key when researching in the context of SE. They also note that many research projects in SE fail in defining the stance of the research, which then further complicates all aspects of the research including its interpretation, validity, replicability etc.

3.1 Research approach

Based on the research problem and research questions, this thesis will take an exploratory approach. A research with an exploratory approach considers questions that address the existence of a phenomena, try to describe something [3].

3.2 Research philosophy

According to Easterbrook [3], making explicit decisions on the research methods is key, and by doing that the research becomes more beneficial for practitioners and other researchers. Therefore they also argue that explicitly defining every step of the research should be done.

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

3.3 Research method selection

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

3.3.1 MLR

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

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

3.3.2 Workshop

To address the empirical study of the software company it was also seen that exploratory research methods was the best fit to answer the research questions. Workshops are a collaborative way to gather data, where

Workshops are used by practitioners in various ways.

3.3.3 Survey

4 Multivocal literature review

Traditionally, in SLRs the reviewed literature consists only of literature that is formally published, and thats motivation of publishing is the publication in itself, e.g. publications in journals and conferences. Material that is produced with commercial interests and informally published material and publications are not considered in SLR [7].

MLRs, are a way to include grey literature into SLRs [6]. 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."* [14]

The Prague Definition of grey literature is strict and therefore not allowing e.g. blog posts to be used on MLRs. However, a specific guideline for including grey literature in literature reviews has been created [7]. This guideline is based on the guidelines on how to perform SLR in SE [9].

4.1 The motivation behind a MLR

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

In software engineering practitioners constantly produce valuable literature in e.g. technical reports or blog posts, but this material is not considered in SLRs. This has been identified as a problem, and there's been a call for MLRs in SE [6].

An SLR would include only the academic papers, and therefore it might not be sufficient to only focus on that. In a MLR the GL should provide a current perspective and fill in the gaps of academic and formal literature [7].

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

A SLR has been conducted on the concept *Programmer Experience* [12]. This SLR will be used to guide this MLR.

4.2 The review protocol of the MLR

DX is a novel concept, and therefore there has been little formal research on the topic. Based on the different levels of literature, white, grey, and black [7] (find correct reference), DX could be seen even to be in the category of black literature.

All data of the MLR can be found **here**. The data collection is done with Google Sheets and is based on the example shown in [7].

4.2.1 Research questions

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

4.2.2 Search process

The search is performed as a manual search by the author from various libraries to gather the academic literature. For grey literature, the Google search engine (https://www.google.com) will be used. To limit the amount of grey literature, only the first 2 pages of searches from google.com will be included. Only 91 percent of participants in a study went to the second page of google results page [15]. Therefore it's appropriate to include only the most relevant results from the google search.

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

Table 2: Academic literature sources for the MLR

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

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

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

Search string for both academic and grey literature will be **"developer experience"**. Because of an ambiguous definition of DX, only one search string is uses.

This assures that all relevant publications are included. Including more words in the search string or creating a more complex search string would require a better understanding of DX, that the author does not have at this moment. Also, including other search strings would bias the search result.

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

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

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

4.2.3 Inclusion criteria

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

4.2.4 Exclusion criteria

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

4.2.5 Quality assessment

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

4.2.6 Data collection and analysis

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

- The source
- Year of publication
- Classification of paper
  - Type of research
  - Scope (Research trends or specific research question)
- Main software engineering topic area

- The author(s) and affiliation (organisation and country)
- Research question/issue
- Definition of Developer Experience
- Point of interest towards Developer Experience (context)
- Summary of paper

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

4.3 Data Collection

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

4.4 Data Analysis

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

4.4.1 Definition of DX

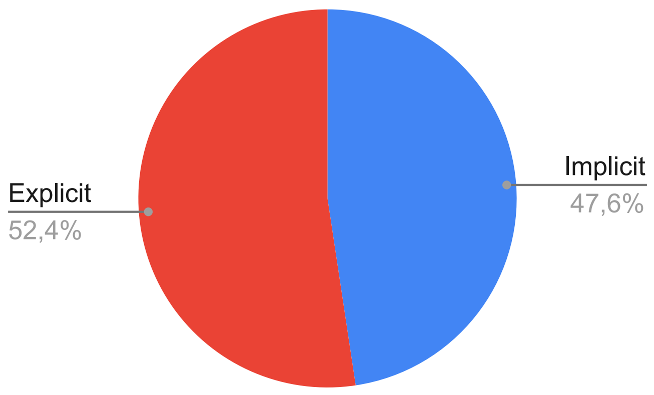
The most common definition of DX in academic papers is the concept and definition of in [5]. This definition is at the moment the only stated definition of DX in the formal literature. There is also further derivations on this concept and definition, but nothing that is revolutionizing or that takes another viewpoint.

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

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

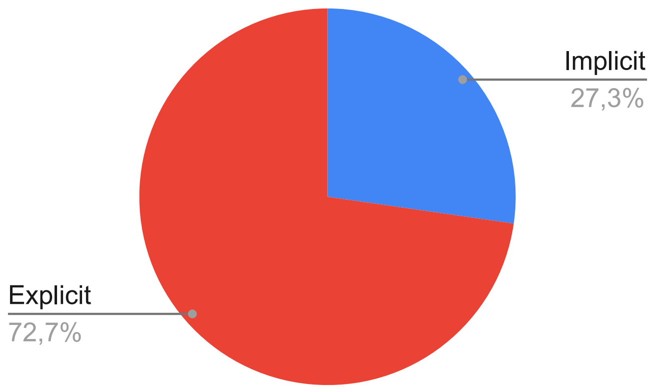
Implicit definition of DX means that the author doesn't give any definition or explanation of what DX is. The definition can often however be inferred from the context of the paper.

Figure 2: Implicit vs. explicit definition of Developer Experience in academic literature



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

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



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

4.4.2 Context of DX

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

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

Table 3: Different contexts that emerged during the literature review

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table 4: Academic and grey literature

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

Table 5: Academic literature

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

Table 6: Grey literature

Figure 4: Combined

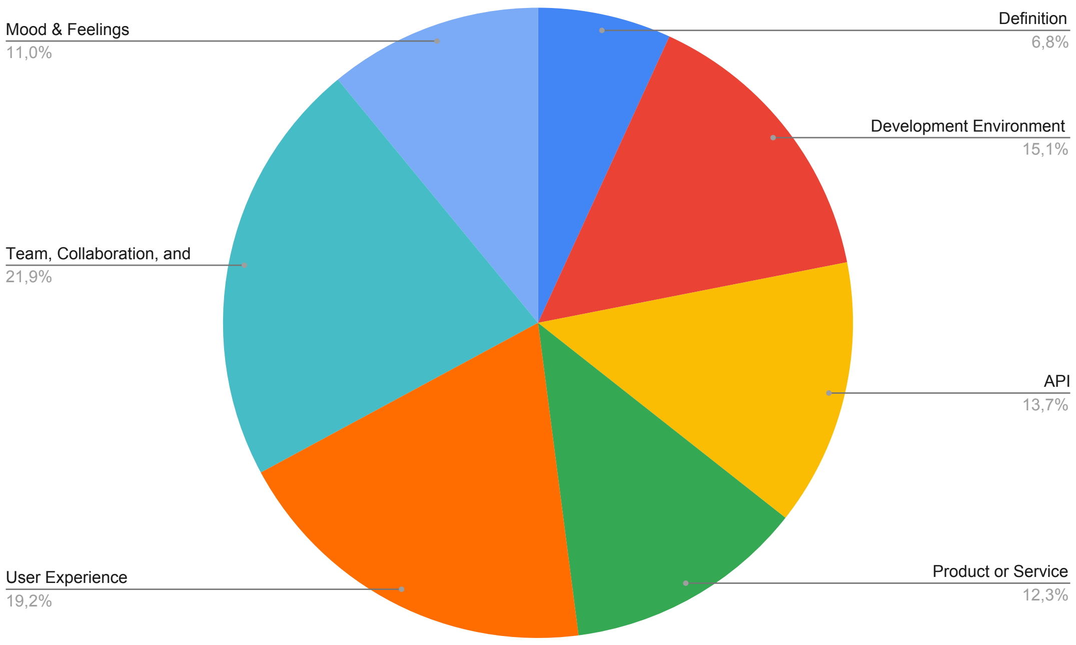


Figure 5: Academic

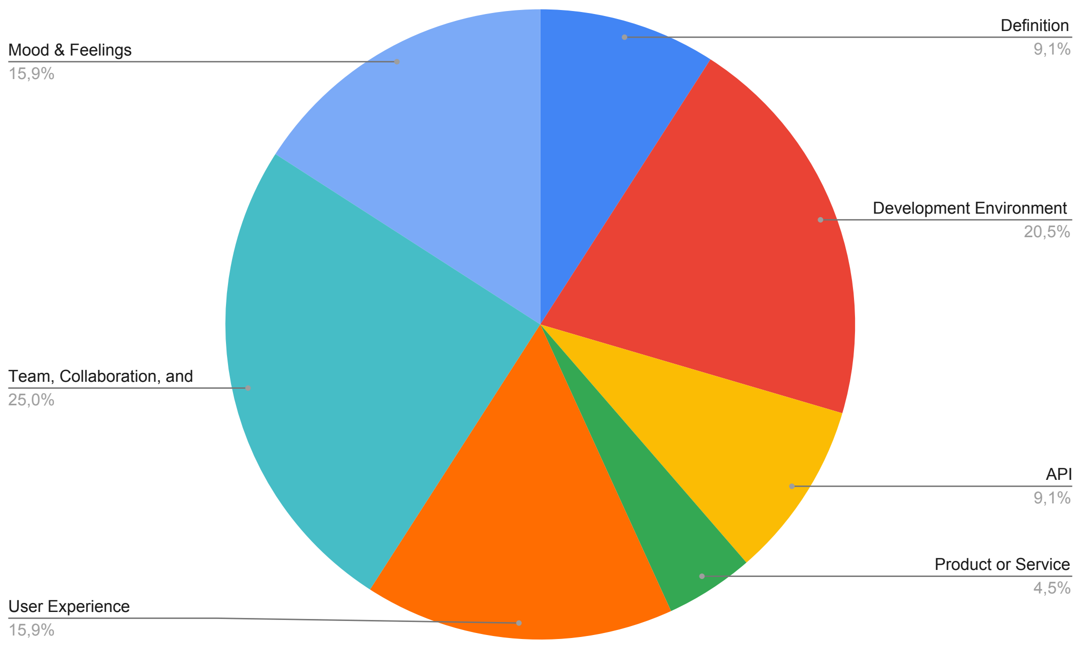
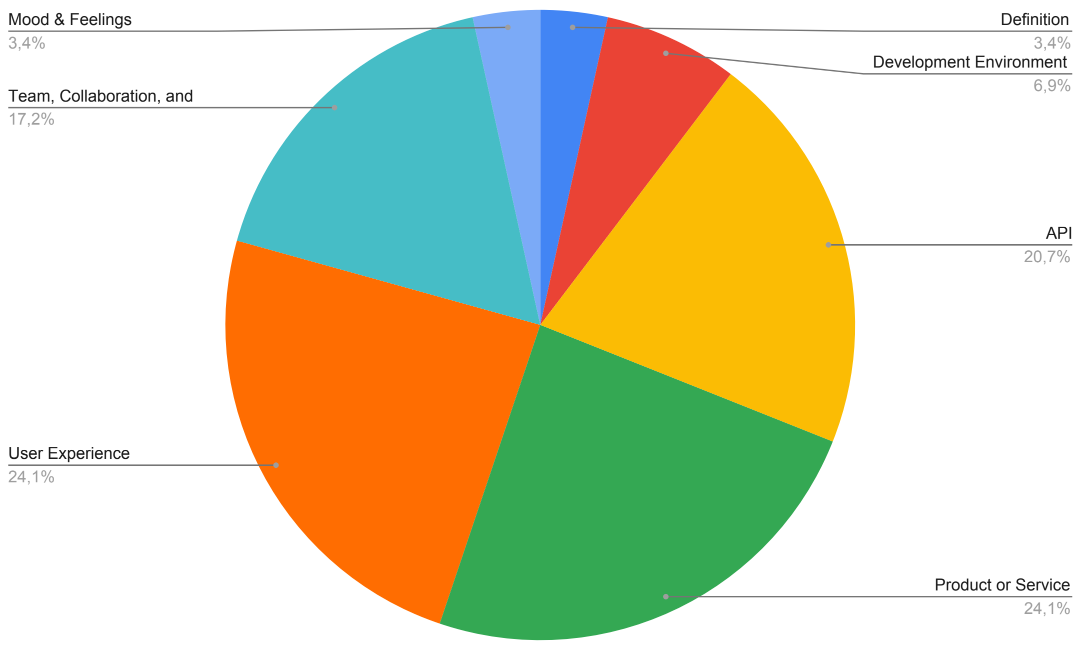


Figure 6: Grey



4.5 Validity of search results

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

5 Exploratory research

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

Possible approaches for the practical part of the study:

– Based on the this article, create a set of guidelines and good practices of DX, and interview and/or measure how well established the application of the practices are

– Conduct interviews studying the development environments of different projects. *"When starting a new project, be upfront that you want to tailor your tools and methods for collaboration"*(uxdesigndx)

– Workshop to map current practices and viewpoints on DX

– Interviews with individual developers on their viewpoint of DX

5.1 Workshop

5.2 Survey (or interview)

6 Results

Answer the research questions and problem.

6.1 Validity of results

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

7 Summary

8 Conclusions

9 Possible references

SLR and MLR

1. K. M. Benzies, S. Premji, K. A. Hayden, and K. Serrett, "State-of-the-Evidence Reviews: Advantages and Challenges of Including Grey Literature,"Worldviews on Evidence-Based Nursing, vol. 3, pp. 55-61, 2006.

2. Rucinski, Taryn. (2015). The Elephant in the Room: Toward a Definition of Grey Legal Literature. Law library journal. 107. 543.

3. Joachim Schöpfel. Towards a Prague Definition of Grey Literature. Twelfth International Conference on Grey Literature: Transparency in Grey Literature. Grey Tech Approaches to High Tech Issues. Prague, 6-7 December 2010, Dec 2010, Czech Republic. pp.11-26.

4. B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software engineering," in "EBSE Technical Report," 2007, vol. EBSE- 2007-01.

5. V. Garousi, M. Felderer, Experience-based guidelines for effective and efficient data extraction in systematic reviews in software engineering, in: International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 2017, pp. 170–179.

6. Lotto, L. S. (1986). Qualitative Data Analysis: A Sourcebook of New Methods: Matthew B. Miles and A. Michael Huberman. Educational Evaluation and Policy Analysis, 8(3), 329–331. https://doi.org/10.3102/01623737008003329

7. Using argumentation theory to analyse software practitioners' defeasible evidence, inference and belief Author links open overlay panel. Austen Rainer

Development tools

1. Murphy,G.C.,Kersten,M.,Findlater,L.:HowareJavasoftwaredevelopersusingtheElipseIDE? Softw. IEEE 23(4), 76–83 (2006)

2. 17. Muslu,K.,Brun,Y.,Holmes,R.,Ernst,M.D.,Notkin,D.:Speculative analysis of integrated development environment recommendations. ACM SIGPLAN Not. 47(10), 669–682 (2012)

3. Kersten, M., Murphy, G.C.: Using task context to improve programmer productivity. In: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT 2006/FSE-14), pp. 1–11. ACM, New York, NY, USA (2006)

4. The Impact of "Cosmetic" Changes on the Usability of Error Messages Tao Dong, Kandarp Khandwala May 2019 CHI EA '19: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems

5. Designing a live development experience for web-components Jens Lincke, Patrick Rein, Stefan Ramson, Robert Hirschfeld, Marcel Taeumel, Tim Felgentreff October 2017 PX/17.2: Proceedings of the 3rd ACM SIGPLAN International Workshop on Programming Experience

6. Are Software Developers Just Users of Development Tools? Assessing Developer Experience of a Graphical User Interface Designer. Kati Kuusinen Conference paper First Online: 23 August 2016

Software Development

1. Capretz, L.F., Ahmed, F.: Making sense of software development and personality types. IT Prof. 12(1), 6–13 (2010)

2. An exploratory study on the influence of developers in technical debt Reem Alfayez, Pooyan Behnamghader, Kamonphop Srisopha, Barry Boehm May 2018 TechDebt '18: Proceedings of the 2018 International Conference on Technical Debt

Soft skills

1. empty

Developer mood

1. Graziotin, D., Wang, X., Abrahamsson, P.: Happy software developers solve problemsbetter: psychological measurements in empirical software engineering. PeerJ2(1), e289(2014)

2. Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H.: Motivation in software engineering: A systematic literature review. IST 50, 860–878 (2008)

3. D. Graziotin, Consequences of unhappiness while developing software, in Proc. 2nd Int. Workshop Emotion Awareness Softw. Eng., May 2017, pp. 42–47.

4. On the Unhappiness of Software Developers Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, Pekka Abrahamsson June 2017 EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering

Flow state and distractions. Sociability and Social Support

1. Ryan, R.M., Mims, V., Koestner, R.: Relation of reward contingency and interpersonal context to intrinsic motivation: a review and test using cognitive evaluation theory. J. Pers. Soc. Psychol. 45, 736–750 (1983)

2. 27.Oehlberg, L., Ducheneaut, N., Thorton, J.D., Moore, R.J.,Nickell, E. 2006. Social TV: Designing for D istributed, SocialTelevision Viewing. In Proc. Euro iTV'06. (2006), 251-259

3. Leont'ev, A. N. Activity, consciousness, and personality. Prentice-Hall Press, (1978)

4. Perlow, L.A., The time famine: Toward a sociology of work time. Admin. Science Quarterly, 44, (1999), 57-81

User Experience

1. Hassenzahl, M., Tractinsky, N.: User experience - a research agenda. Behav. Inf. Technol. 25(2), 91–97 (2006)

2. Henrique Henriques, Hugo Lourenço, Vasco Amaral, and Miguel Goulão. 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 (MODELS '18). ACM, New York, NY, USA, 200-210. DOI: https://doi.org/10.1145/3239372.3239387

3. Ferreira, J. , Sharp, H. and Robinson, H. (2011), User experience design and agile development: managing cooperation through articulation work. Softw: Pract. Exper., 41: 963-974. doi:10.1002/spe.1012

Other

1. Kansala, M. and Tuomivaara, S. (2013). Do Agile Principles and Practices Support the Well-being at Work of Agile Team Members? In Proceedings of the 8th International Conference on Software Engineering Advances (ICSEA 2013), pages 364–367.

2. Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6):1213–1221.

3. Journal of Systems and Software Volume 140, June 2018, Pages 32-47 Journal of Systems and Software What happens when software developers are (un)happy Author links open overlay panel Daniel Graziotina Fabian Fagerholm bcXi-aofengWangd PekkaAbrahamssone Show more https://doi.org/10.1016/j.jss.2018.02.041

4. Teamwork quality and project success in software development: A survey of agile development teams Author links open overlay panelYngveLindsjørnaDag I.K.Sjøbergab TorgeirDingsøyrbc Gunnar R.Bergersen Tore Dybåa

5. Gass, O., Meth, H., Maedche, A.: PaaS characteristics for productive software development: an evaluation framework. Internet Comput. IEEE 18(1), 56–64 (2014)

6. Deci, E., Ryan, R.M.: Self-determination theory. Handbook of theories of social psychology. SAGE, Los Angeles (2012). ISBN 9780857029607

Workshops

1. Sanne Akkerman and Arthur Bakker. Boundary Crossing and Boundary Objects. Review of Educational Research, 81(2):132–169, 2011. ISSN 08919976. doi: 10.3102/0034654311404435.

2. Riitta Smeds, Rita Lavikka, Miia Jaatinen, and Antero Hirvensalo. Interventions for the co-creation of inter-organizational business process change. In Proceedings of the International Conference on Advances in Product Management Systems, pages 11–18. Springer, 2015. ISBN 9783319227580. doi: 10.1007/978-3-319-22759-7 2.

3. Frank Lyman. Think-Pair-Share: An expanding teaching technique. Maa- Cie Cooperative News, 1(1):1–2, 1987.

4. Guillermo E Leon de la Barra, De Barra, Mario B Leon de la Barra, and Ana M Urbina. Creative Problem Solving Workshops for Engineering Stu- dents. In Proceedings of the 27th Annual Conference. Frontiers in Ed- ucation: Teaching and Learning in an Era of Change, volume 3, pages 1428–1430. IEEE, 1997. ISBN 0780340868.

5. Haridimos Tsoukas. A Dialogical Approach to the Creation of New Knowl- edge in Organizations. Organization Science, 20(6):941–957, 2009. ISSN 1047-7039. doi: 10.1287/orsc.1090.0435.

6. Guillermo E Leon de la Barra, De Barra, Mario B Leon de la Barra, and Ana M Urbina. Creative Problem Solving Workshops for Engineering Stu- dents. In Proceedings of the 27th Annual Conference. Frontiers in Ed- ucation: Teaching and Learning in an Era of Change, volume 3, pages 1428–1430. IEEE, 1997. ISBN 0780340868.

7. David Coghlan. Action research: Exploring perspectives on a philosophy of practical knowing. Academy of Management Annals, 5(1):53–87, 2011. ISSN 19416520. doi: 10.1080/19416520.2011.571520.

8. Lotte S Luscher and Marianne W Lewis. Organizational Change and Managerial Sensemaking: Working Through Paradox. Academy of Management Journal, 51(2):221–240, 2008. ISSN 00014273. doi: 10.1103/PhysRevLett. 108.187601.

References

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

[2] T. DeMarco and T. Lister. *Peopleware: productive projects and teams.* Addison-Wesley, 2013.

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

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

[5] F. Fagerholm and J. Münch. Developer experience: Concept and definition. *CoRR*, abs/1312.1452, 2013.

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

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

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

[9] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering, 2007.

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

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

[12] J. Morales, C. Rusu, F. Botella, and D. Quinones. Programmer experience: A systematic literature review. *IEEE Access*, PP:1–1, 05 2019.

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

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

[15] A. J. van Deursen and J. A. van Dijk. Using the Internet: Skill related problems in users' online behavior. *Interacting with Computers*, 21(5-6):393–402, 06 2009.