

Impact of Developer Experience in the outcome of Software Projects

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

Copyright © 2019 Anders Nylund



Author Anders Nylund		
Title Impact of Developer Experience in the outcome of Software Projects		
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 28	Language English

Abstract

Your abstract in English. Keep the abstract short. The abstract explains your research topic, the methods you have used, and the results you obtained.

The abstract text of this thesis is written on the readable abstract page as well as into the pdf file's metadata via the \thesisabstract macro (see above). Write here the text that goes onto the readable abstract page. You can have special characters, linebreaks, and paragraphs here. Otherwise, this abstract text must be identical to the metadata abstract text.

If your abstract does not contain special characters and it does not require paragraphs, you may take advantage of the abstracttext macro (see the comment below).

Keywords Developer Experience, Software Projects

Författare Anders Nylund

Titel Impact of Developer Experience in the outcome of Software Projects

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** 28 **Språk** Engelska

Sammandrag

Sammandrag på svenska. Try to keep the abstract short. Abstract explains your research topic, the methods you have used, and the results you obtained.

Nyckelord Nyckelord på svenska, temperatur

Preface

I want to thank Professor Pirjo Professori and my instructor Dr Alan Advisor for their good and poor guidance.

Otaniemi, Date to te announced

Anders Nylund

Contents

Abstract	3
Abstract (in Swedish)	4
Preface	5
Contents	6
Thesis dictionary	8
1 Introduction	9
1.1 Motivation	9
1.2 Research problem and questions	10
1.3 Scope and focus	10
1.4 Structure of the thesis	11
2 Background and literature review	12
2.1 Programmer Experience	12
2.2 Developer Experience	12
2.3 Intrinsic Motivation	13
2.4 Performance Alignment Work	13
2.5 Happiness of developers	13
2.6 Selection of tools	13
2.7 Flow state	14
2.8 Team, community, and collaboration	14
3 Research material and methods	15
4 Multivocal literature review	16
4.1 The motivation behind a MLR	16
4.2 The review protocol of the MLR	17
4.2.1 Research questions	17
4.2.2 Search process	17
4.2.3 Inclusion criteria	18
4.2.4 Exclusion criteria	18
4.2.5 Quality assessment	18
4.2.6 Data collection and analysis	18
4.3 Data Collection	19
4.4 Data Analysis	19
4.4.1 Definition of DX	19
4.4.2 Context of DX	19
4.5 Validity of search results	20
5 Interviews and case studies	21

6	Results	22
6.1	Validity of results	22
7	Summary	23
8	Conclusions	24
9	Possible references	25
	References	28

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 engineering and development is a complex practice that requires both technical and social skills. Compared to other engineering professions, software engineering is still in early stages. The best practices are still evolving, new ideas are coming and previous ideals 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 required features. Software products are however used by people, and therefore it is a requirement to understand how social interactions work. Software is also most of the times

Software developers are in an interesting role where they are both creators and designers when they write the logic that makes up the product. Meantime they are also users of tools that they use to create the product. Developers using a software product that helps in their creative design work will create an user experience for them. 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.

In recent academic research and internet articles a concept called Developer Experience (DX) has gotten traction. DX is a term that explains how developers experience their development environments, both technically and socially. The same way as User Experience (UX) is considering the user of a system or tool, DX can be seen as the experience that developers have as users of a system. Here the system however includes the tools, frameworks, processes that the developer is the user of when developing software.

DX is more prevalent and therefore 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.

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). However, that is only one viewpoint on DX, as it also includes the feelings and perceptions of the developers. In some research the term *Developer Experience* with the abbreviation of DE^x is used, and in some other research the term *Programmer eXperience* and abbreviation PX is used [3] [10]. This shows also 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.

For the author the DX means having a low friction and easy setup with their own development environment. They want to have an environment that is lightweight, fast, and easy to use. It should have a short cycle of feedback i.e. when making a change to the source code it should be immediately reflected in the output. This might be the reason why they like to develop for the web, as the tools are often quick and have a fast feedback cycle. The environment should perform tasks automatically as building, reporting errors. The frontend JavaScript framework React and the tools supporting it are a great examples of excellent DX. The tools are intuitive and guide the developer in making the right things. After all learning new technologies is not about solving new problems, but it's about solving the same old problems more efficiently, faster, easier i.e. with a better DX.

1.2 Research problem and questions

Research problem: 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?

- RQ 1** What is the difference in the definition of Developer Experience in academic literature and in industry literature?
- RQ 2** What aspects of Developer Experience are currently being considered in software projects? What aspects of Developer Experience do developers see as valuable?
- RQ 3** What is included in replicable practices and techniques that can be utilized to create a good Developer Experience in software project teams?

Table 1: The research questions

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. ...???

2 Background and literature review

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 [10]. 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) [3]. In a study it was also concluded that the cognitive part of DX is also addressed via intention and affect [9].

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* [10]. 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 [10]. 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 [2]. This thesis might focus more on the technical

environment.

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 [9] (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 [6]

2.6 Selection of tools

Perceived choice is a perception of that the choice has already been made [9]. 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 [9]. 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 [8]. 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 [8], 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.

[11] 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

3 Research material and methods

What material will be used in the research and what methods/methodologies will be used to study the problem. What kind of approach to research will be used in the thesis.

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

4 Multivocal literature review

Traditionally, in SLRs the reviewed literature consists only of literature that is formally published, and that's 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 [5].

MLRs, are a way to include grey literature into SLRs [4]. 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." [12]

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 [5]. This guideline is based on the guidelines on how to perform SLR in SE [7].

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 [4].

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 [5].

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* [10]. 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 [5] (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 [5].

4.2.1 Research questions

The foundation for the MLR is the first research question [RQ1](#) (What is the difference in the definition of Developer Experience in academic literature and in industry literature?). 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.

IEEEExplore	(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 be

- 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 used. 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 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 through the data points that emerged during the collection of the data

4.4.1 Definition of DX

A majority of the conference papers base their definition of DX on [3]. This definition is at the moment the only stated definition of DX in the formal literature. There are however some other derived definitions of DX in the formal literature.

In grey literature the definition of DX is loosely based, and no sources of definition is given in the articles.

4.4.2 Context of DX

From the analysis of the material, there is a clear indication that there are different viewpoints to Developer Experience.

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.

0 to 200 / Time to Hello World

User Starting Experience

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

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 Interviews and case studies

Define what kind of practical approach to the study subject is going to be taken

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
- An interview 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"

–

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. Benzie, 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.: How are Java software developers using the Eclipse IDE? *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 problems better: psychological measurements in empirical software engineering. *PeerJ* 2(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., Thornton, J.D., Moore, R.J., Nickell, E. 2006. Social TV: Designing for Distributed, Social Television 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

References

- [1] L. F. Capretz. Personality types in software engineering. *International Journal of Human-Computer Studies*, 58(2):207 – 214, 2003.
- [2] F. Fagerholm. *Software Developer Experience: Case Studies in Lean-Agile and Open Source Environments*. PhD thesis, University of Helsinki, 2015.
- [3] F. Fagerholm and J. Münch. Developer experience: Concept and definition. *CoRR*, abs/1312.1452, 2013.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering, 2007.
- [8] 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.
- [9] 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.
- [10] J. Morales, C. Rusu, F. Botella, and D. Quinones. Programmer experience: A systematic literature review. *IEEE Access*, PP:1–1, 05 2019.
- [11] 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.
- [12] 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.