Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Dongmei Gao

Measuring Developers' Episodic Experience of Low-Code Development Platforms

Master's Thesis
Espoo, September 26, 2022

Supervisor: Prof. Fabian Fagerholm
Advisor: M.Sc. (Tech) Tomi Laakso

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: Dongmei Gao

Title:
Measuring Developers' Episodic Experience of Low-Code Development Platforms

Date:	September 26, 2022	**Pages:**	63 + 5
Major:	Human-Computer Interaction	**Code:**	SCI3097
Supervisor:	Prof. Fabian Fagerholm		
Advisor:	M.Sc. (Tech) Tomi Laakso		

Developer Experience (DX) is being emphasized in recent research, either on the topic of understanding its concepts or measurements. It refers to how developers think and feel as they work towards a goal in the software development environment. Episodic experience is the term used to describe the experience that arises over a period of minutes or hours. Just as the user experience is based on the use of the product, DX is often linked to the development tools. However, measuring developers' episodic experience when using software development products for a particular task in a given situation has not been specifically studied. In addition, Low-code Development Platforms (LCDPs) are becoming popular in recent years with the aim of allowing users to build applications easily and efficiently. The users of LCDP can be seen as special developers, with or without professional programming experience. This thesis presents a questionnaire to measure developers' episodic experience of a specific LCDP with various experience items based on literature review and Delphi study with experts knowledgeable about DX. Observational task-based test was used to validate how it correlated with different aspects of the platform. The role of prior backgrounds in this context was also investigated. Using descriptive and statistical methods, I found significant differences in the prediction of tool quality and task difficulty across experience items. I also compared these items in groups with similar backgrounds. The results of the study indicated the experience of programming, LCDP, and design showed various impacts on the performance and feelings of developers. Based on the results of the study, the questionnaire with experience items is able to make some recommendations for the design of LCDPs. The thesis also provides some insights for further research in this area.

Keywords: episodic experience, developer experience, low-code development platform, experience measurement, software engineering

Language: English

Acknowledgements

I wish to thank my supervisor PhD Fabian Fagerholm for his guidance during the whole process of the thesis. He expressed interest in my initial topic and suggested a good direction for my research. He answered all my questions patiently and gave me very helpful advice. I also thank my advisor Tomi who helped me better understand the low-code development tool that used in this study. Besides, I would like to thank all the experts involved in the Delphi study who showed interest in the subject and were happy to communicate and share their experiences. Finally, I am grateful to my friends who were doing their thesis at the same time. We often discussed together and learned from each other about research ideas and methods. And more importantly, we always encouraged each other when we encountered any difficulties.

Espoo, September 26, 2022

Dongmei Gao

Abbreviations and Acronyms

DX	Developer Experience
HCI	Human-Computer Interaction
UX	User Experience
IDE	Integrated Development Environment
UEM	User Experience Measurement
OAG	Online Application Generator
LCDP	Low-code Development Platform
API	Application Programming Interface
UI	User Interface
EUD	End-User Development
SE	Software Engineering
ICC	Intraclass Coefficient Correlation
DEXI	Developer Experience Scale

Contents

Chapter 1

Introduction

Software development is a product development process that builds a software system or software part of a system in response to changing user requirements. The software development life cycle has also evolved through history. In recent years, agile software development — occurring in short intervals and software releases are made to capture small incremental changes, has been applied in more and more technology companies (Ruparelia, 2010).

Software developers are a significant role in the software development context. Developers have dual roles as both a producer of a software product and a user of a development tool such as an Integrated Development Environment (IDE). Understanding DX is necessary because it can better support software developers in their activities and ultimately improve their well-being (Kuusinen, 2015). Research in Human-Computer Interaction (HCI) and user experience (UX) fields also provide a lot of ground work to support the study of DX. Measuring developer experience can be difficult, not only because it is a subjective feeling, but also because it requires the researcher to know the developers, to be familiar with their working environment, to know how the developers behave in their work, how they interact with the development tools, what their goals are and what their attitude towards their work is. To improve developer experience, various development tools are designed to facilitate their work efficiency and reduce the repeated work. Customised settings in IDEs give developers the freedom to choose their own preferences. The agreed code specification also helps them to be more comfortable with teamwork, reducing the work to merge code.

Just as users become experts in the products they typically use and then user experience gradually changes, developers also develop a long-term relationship with the development tools they are used to. DX varies, from their earliest awareness to their use over a long period of time. We use episodic experience to describe how developers feel after a single event or given task.

Low-code development platforms enable their end users to build applications visually by interaction with the app builder interface. End users are only required to write little or no code during the development process. For the latter case, we also call the platform a no-code platform. SAP AppGyver, used in this study, is the world's first professional no-code platform. LCDPs have attracted public attention because of its efficiency and convenience. Users of LCDPs are a special group of developers who may have diverse backgrounds or even no programming experience. This thesis strives to explore what this kind of developer perceives when they perform a task on a LCDP.

1.1 Motivation

Most of previous research directly applied user experience measurements (UEMs) to measure DX. However, due to their different roles and characteristics, the influencing factors of the DX and those of the end-user experience vary greatly. For example, developers might be forced to use a specific development tool since they are working for their employers. Lack of freedom to choose what they like may affect their motivation. Moreover, collaboration is also an inevitable factor during the development cycle since software development is a very team oriented work. Thus, selection of the most suitable influencers in terms of the characteristics of the developers and the tools used as well as context is the main topic of this thesis.

Considerable research has been undertaken concerning DX using the development infrastructure for a period of time. Developers share their experiences and opinions through interviews, questionnaires and other research methods. This type of research is more oriented towards the overall DX of the infrastructure. It works well in determining the direction of the product design and development. If we are interested in testing a particular feature of the product, especially if it is new, we might be more inclined to know more about DX for this single feature. **Episodic experience**, which refers to a single event that occurs under certain circumstances (Fagerholm, 2015), is advantageous for limiting the scope of the testing. The development tool or platform can make targeted improvements based on developers' feedback. We believe software development will benefit from the identification of concrete problems.

So far, a universal approach measuring DX has not been agreed. Researchers selected applicable methods for diverse scenarios. Nevertheless, most developers in previous research refer to traditional developers, with professional programming knowledge and implementing the development by

writing source code. One category of developer is rarely mentioned and recognized — **citizen developers**, developers with little or no software development background (Oltrogge et al., 2018). They build the application on the top of online application generators (OAGs) which are a type of web based software that enable users to create various types of applications. On the one hand, this kind of platform decreases the cost of development and maintenance, and on the other hand, developers can focus more on creation rather than on complex repetitive tasks. Although Oltrogge et al. (2018) addressed the security of OAGs, a survey (Lebens et al., 2022) related to the usage of OAGs indicated that they had been already prevalent across all sizes of companies. Meanwhile, the proliferation of LCDP, a kind of OAGs, caters to the needs of users or organizations and has inspired more citizen developers.

Therefore, this thesis is concerned with episodic experience of developers including traditional and citizen developers using a LCDP.

1.2 Research Problem and Questions

The main goal of this thesis is to explore **how to measure developers' episodic experience of low-code development platforms.** The most common and convenient way of measuring instrument is questionnaire. Using a questionnaire as the research method has several benefits. For example, it is easy to collect a large amount of information quickly, and respondents have the anonymity and autonomy to choose whether or when to fill out the questionnaire (Gillham, 2008). In one real usage scenario, the questionnaire displays automatically after developers doing a task on the platform. Developers are free to fill in it or not. To develop a questionnaire, the first step was deciding experience factors that might be related to both episodic experience and LCDP. Thus, our first research question is as follows:

RQ 1 Which experience factors are associated with the developers' episodic experience of LCDPs?

In addition, the impact of the prior background is also striking for episodic experience and LCDPs. For using the platform for the first time in a given situation, developers do not have enough time to get familiar with it. The learning cost is still a problem for most of LCDPs. Developers' prior experience determines how much effort and time they need to invest in learning the platform. And during this process, their feelings and performance might be greatly affected. This leads to our second research question:

RQ 2 How does background affect developers' episodic experience of LCDP?

The last research question is about how this thesis could contribute to further research into this topic:

RQ 3 What insights can be derived from this study for further research into episodic experience and LCDPs?

Currently the related research is insufficient so our research can be regarded as a kind of trial. We expected that our findings could help to have a more holistic knowledge on both episodic experience and LCDP, provide some insights and expose some potential problems to delve deeper into this topic in the future. To answer these research questions, the thesis research was organized in four stages shown in Figure 1.1.

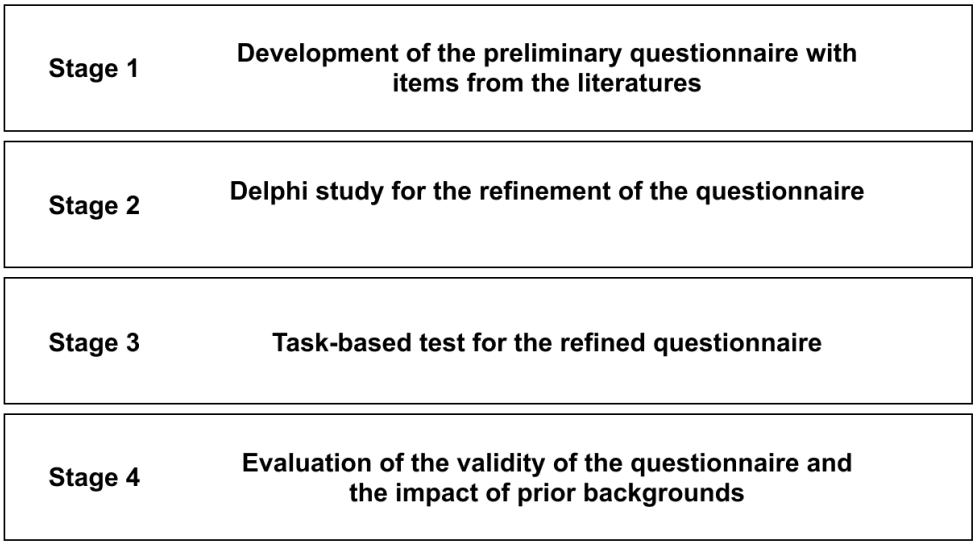

Figure 1.1: Research flow for the thesis work.

1.3 Scope and Focus

This thesis aims to create and validate a questionnaire for measuring developer experience of LCDPs. **Developers** in this study refer to people who

build applications on LCDPs, with minimum prerequisites for experience with office tools. They might be traditional developers or citizen developers. Diverse background is the unique characteristic of users of LCDPs. The office tool requirement is to ensure that developers have a certain level of learning and software adaptability. But to understand the impact of background and prior knowledge on episodic experience, participants are grouped by programming, design and LCDP experience. The study also seeks to reveal the relationship between personal background and developer experience. The verification test is run on a specific LCDP which has common features as general LCDPs. The goal is to explore the influencing factors that impact developers involved in a specific task on the LCDP and provide insights for the design and development of LCDPs. Due to the specificity of LCDPs and time limitation, it is difficult to verify the applicability of the questionnaire to other platforms or tools. Future studies may address the performance of one or several influencing factors across multiple platforms.

1.4 Structure of the Thesis

The structure of the thesis is as follows: Chapter 1 presents and motivates the problem in this thesis. A literature review on the subject of developer experience and low-code development platform is shown in Chapter 2. Chapter 3 discusses the process of developing a questionnaire from the preliminary questionnaire to the final improved version. Chapter 4 conducts user testings to validate the questionnaire and analyzes the result. Chapter 5 answers research questions, reveals the limitation of the study and looks ahead to the future work. Finally, Chapter 6 presents a summary of the results of the thesis.

Chapter 2

Background

In this section, the literature review focuses on a series of relevant topics, including user experience, developer experience, and LCDPs. Distinguishing between UX and DX is crucial. They are interrelated but also to some extent independent in concepts and frameworks. Both relate to the use of or interaction with products or services. DX places more emphasis on the users' role as developers. Since this thesis is about measuring instruments, the literature review also consists of multiple methods for user research as well as evaluating DX. Most of the user research methods are applicable to the developer's experience, but a few appropriate adjustments are needed to make based on scenarios. The evaluation of DX is discussed in Section 2.3. Moreover, Section 2.4 introduces the concepts and characteristics of LCDPs by comparing with traditional software development and also reviews current research on LCDPs.

2.1 User Experience

The International Organization for Standardization (2018) defines user experience as "the user's perceptions and responses that result from the use and/or anticipated use of a system, product or service (ISO, 2018)". As an essential part of the HCI domain, a vast amount of research have been investigating the concept of UX, but without an agreement on its definition and evaluation.

Law et al. (2009) conducted a survey to collect opinions about the nature and scope of UX from academia and industry. Participants were asked to choose the definition that they agree with most within five definitions. Respondents' subjective opinions on UX emphasized perceptions and reactions to the product which were consistent with the ISO 9241 standards. The au-

thors also suggested clarification of "anticipated use" in the definition since 14% respondents didn't understand it exactly. According to their views, UX is dynamic, context-dependent, subjective and focuses on the interaction between a person and a user interface (UI). Although respondents owned different backgrounds, such as nationality or working years, it only caused limited variation in agreement on definition choices.

Hassenzahl and Tractinsky (2006) advocated advancing our understanding of UX from empirical UX research. They concluded three significant facets of UX and anticipated potential related challenges for future research. Figure 2.1 presents the ideas of each perspective as well as the crossover with other perspectives. The first perspective is to promote system quality with

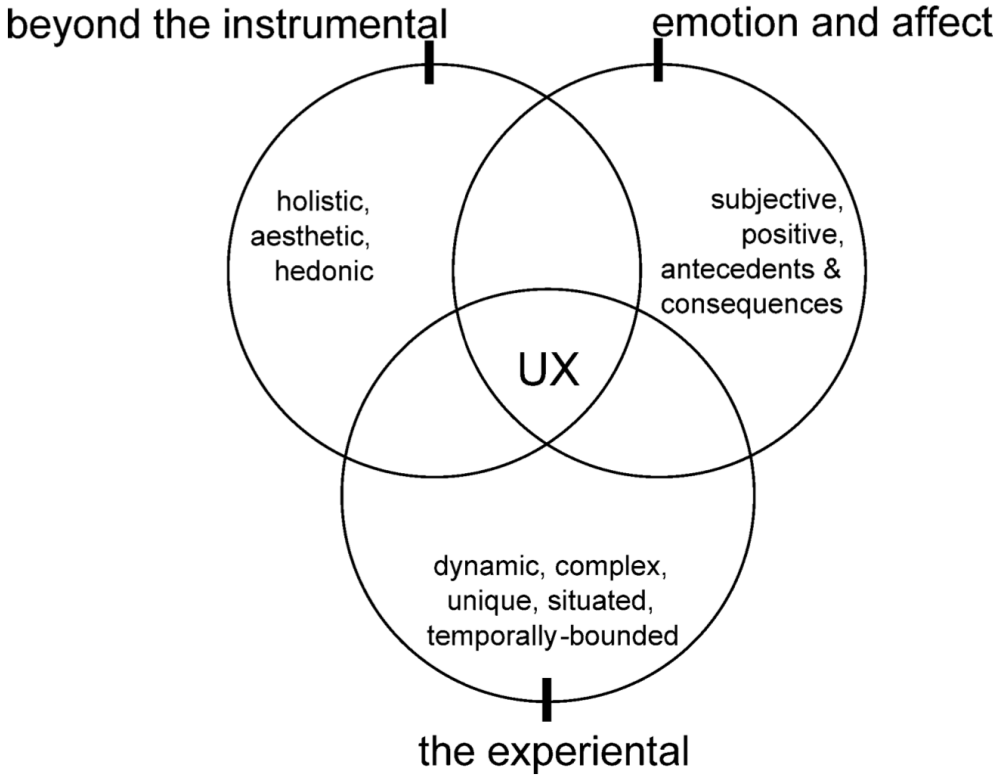

Figure 2.1: Three facets of UX (Hassenzahl and Tractinsky, 2006).

non-instrumental factors, such as aesthetics, intimacy, or hedonic aspects. Next, affect and emotions play crucial roles in UX, and concentrate on acquiring positive emotional experiences. Finally, the nature of experience is emphasized. It can be complex and unique, or subjective and dynamic.

2.1.1 User Research

Knowing and improving UX of products or services rely on user research. When users use a product or service, they might feel enjoyable and appreciated. But some may not realize that these benefits come from long or short term user research that eventually translates the results of UX into design and application. On the other hand, the lack of user research usually leads end-user frustration. The role of user research may seem less obvious, but it actually affects the entire production process of the product. In the book *Observing the User Experience A Practitioner's Guide to User Research* authored by Goodman and Kuniavsky (2012), a cross-industry study conducted by the consulting firm Accenture, 57% deemed that "inability to meet customer needs" led to the failures of new products or services. 50% complained about "the lack of a new or unique customer-perceived value proposition". Therefore, products need to be closely aligned with user needs. User research is one way to connect products and their audiences and measure user experience of products or services.

It is no longer a one-size-fits-all world. Culture differences always make migration of products difficult (Kumar and Whitney, 2003). In the past, western products usually appeared in foreign countries simply by changing their language. But now, user research has become a necessary strategy before big companies open up new markets.

2.1.2 User Research Methods

A vast number of research methods have emerged in the field of user experience. Determining when to use which methods is one of the main questions for many projects. Rohrer (2014) created a 3-dimensional framework from three axes, as shown in Figure 2.2, to help better understand how to select methods: 1) Attitudinal vs. Behavioral, 2) Qualitative vs. Quantitative, 3) Context of Use. Different approaches depending on the nature and conditions of the project.

The author also presented another time-dimension aspect to select methods. Regardless of the methods used, they should meet the explicit and implicit project requirements as much as possible. If we consider the process of product development, different methodologies can be applied to user research at different stages of development. For example, in the initial stage of the product development, we seek new design opportunities. Typical methods such as field studies, diary studies or data mining can be useful for inspiration. After we decide on the design direction, card sorting, participatory design, paper prototype and usability studies can help designers refine the

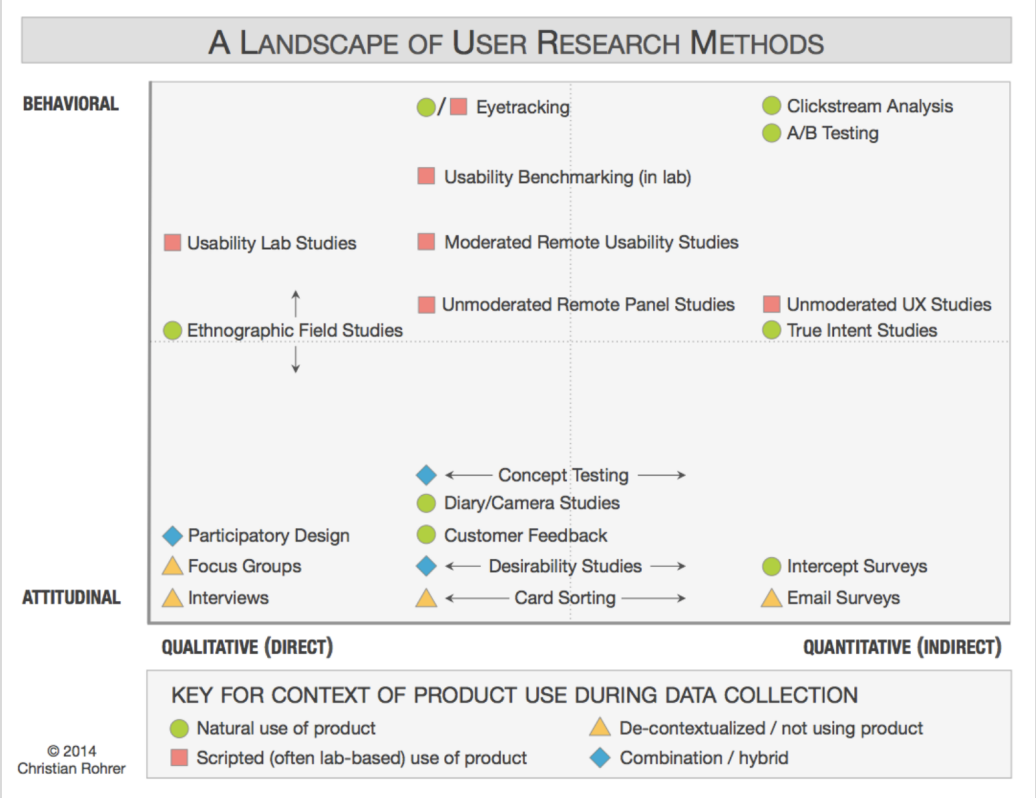

Figure 2.2: A landscape of user research methods (Rohrer, 2014).

design and improve the usability. On the finally assessment phase, quantitative methods serve as a summary, including usability benchmarking, online assessments, surveys and A/B testing.

There are currently some flourishing examples of application of user research methods. For example, an European project (eCAALYX) aimed at improving the quality of life of elderly people with chronic diseases, and the final product is a TV user interface.(Nunes et al., 2010). The user research methods adopted in this project consisted of the literature review and some informal interviews. The study discussed how perception, cognition, mental and psychosocial aspects change with age, respectively. Then the information collected and analyzed in the user research was incorporated into the creation of the personas. The purpose of personas was to deepen the understanding of the problems, goals and needs of the end-users. They described the causes and characteristics of patients with different diseases. Furthermore, another user research that also occurs in the healthcare industry focused on the design of a new medical device (Martin et al., 2012). By

conducting the open-ended semi-structured interview, the study suggested the fundamental change of the concept of the medical device and identified safety precautions when developing the device. The user research eventually contributed to saving the time and cost of producing and marketing inputs. Diary study is also widely used during user research. One of examples was reported by Czerwinski et al. (2004) about how people interweave multiple tasks with interruptions. Participants wrote down the task complexity, task duration and other related information when they switched to different tasks. The main findings eventually gave insights to guide the design of software task-management tools that can assist the workers in multi-tasking. Overall, user research is essential to improving the user experience of a product, and it plays a major role in the entire product life cycle.

2.2 Developer Experience

In the software development process, not only do developers' skills and allocated time matter, but their motivations also play an important role (Kaltio and Kinnula, 2000). Kuusinen (2015) also mentioned software development requires developers not only technical skills to create the software but also social skills to collaborate with team members and stakeholders. Baddoo et al. (2006) stated in their case study that intrinsic factors, such as autonomy and sense of achievements, is one of essential motivators of software developers. They also confirmed that good software developers with characteristics like proactive and performing well in teamwork had a positive impact on software projects. Moreover, developer happiness is assumed to be positively correlated with the success of a project. However, Linberg (1999) revealed in some cases developers feel satisfied with their job even if the project is a failure by the traditional definition. They may have their own judgement of success. As a result, the research into the definition and measurement of developer experience (DX) has emerged.

Influenced by the UX concept, Fagerholm and Münch (2012) described developer experience as "a means for capturing how developers think and feel about their activities within their working environments".

In more detail, Fagerholm and Münch (2012) analyzed the DX from the focus areas and end goal. The prescriptive process models rather than user interface design, build the basic layer. Descriptive process models and adaptive process models contribute to the usability and efficiency of a product. For specific context, understanding the process-product relationship facilitates the most appropriate use. Finally, the entire experience is determined by the software developers and their activities in the environment. They

also proposed a conceptual framework to conclude developer experience from three main dimensions - cognition (development infrastructure), affect (feeling about work) and conation (the value of one's own contribution). Figure 2.3 depicts the framework.

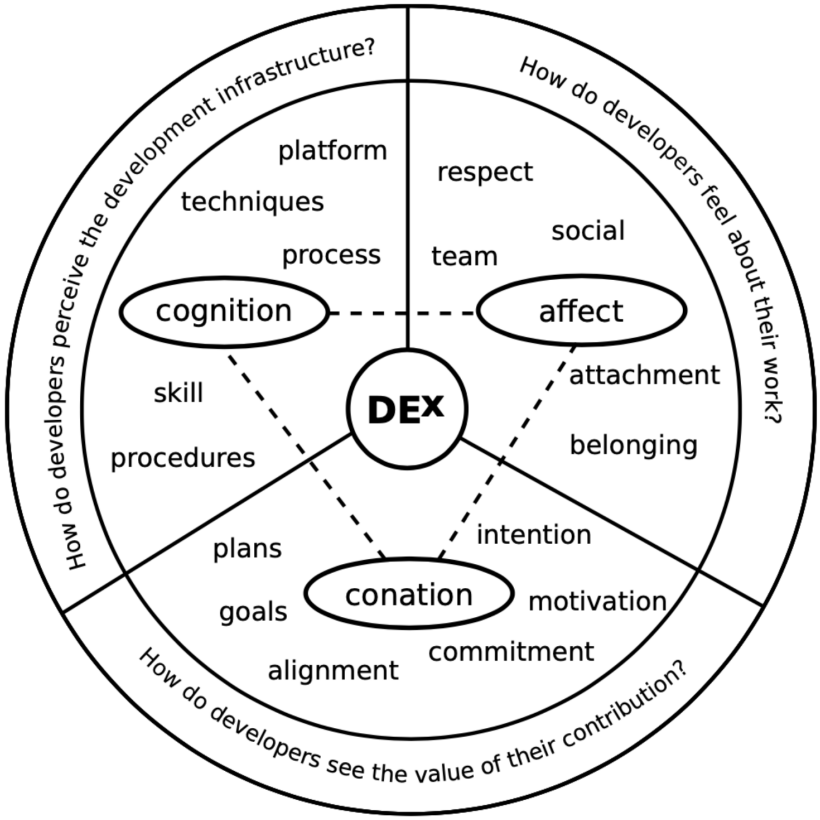

Figure 2.3: The conceptual framework of DX (Fagerholm and Münch, 2012).

On top of Fagerholm and Münch (2012)'s framework, Lee and Pan (2021) extended the explanation of three dimensions in study of measuring DX toward the use of a Deep Learning (DL) platform. The "cognition" is the rational basis provided by the DL platform, "affection" is explained as the emotional state, and "conation" is about the developer's tendency to use the DL platform voluntarily.

However, after reviewing a large amount of literature, Nylund (2020) found the definition of DX is still vague. To better conclude what DX means in practice, he performed a multivocal literature review study to analyze articles related to developer experience from publications and grey literature. This study listed all related contexts of developer experience, such as de-

velopment environment, collaboration or mood. Finally, Nylund concluded the results that DX is a consequence of the feelings and perceptions of the developers in a software development context. Fagerholm (2015) categorized developer experience in his theoretical framework (Figure 2.4) into immediate experience, episodic experience and cumulative experience. Immediate experience is generated by the first impression, episodic experience emerges during a specific duration and accumulated experience happens after a long-term usage. Episodic experience is advantageous for testing a particular feature of the product. The development tool or platform make targeted improvements based on developers' feedback. Control and analysis of the relevant external and internal conditions are essential in testing since many unique factors affect the experience during this time, such as time pressure, task difficulty, developer commitment, etc.

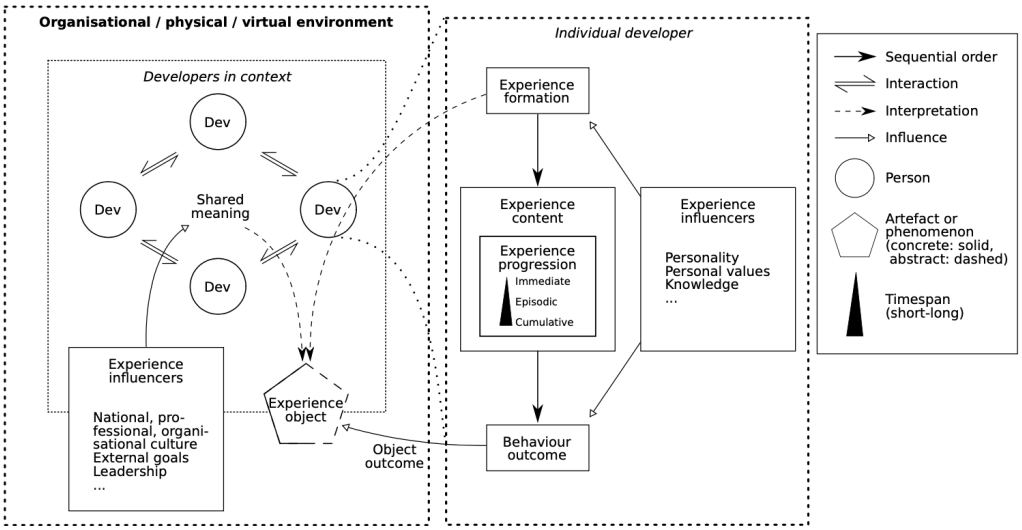

Figure 2.4: Theoretical Framework of Developer Experience (Fagerholm, 2015).

2.3 Developer Experience Evaluation

In addition to the understanding of DX, substantial research have also attempted to evaluate the DX in various development contexts, such as policies or platforms. Being a sub-branch of UX, researchers of DX often apply UX approaches directly. However, in some cases such as the creation of new software tools, the existing UX approaches are not particularly appropriate.

Tuomi et al. (2021) built a framework for evaluating and adapting usability evaluation methods (UEM) for DX evaluation. By identifying assumptions, identifying conflicts and adaptation, a given UEM approach can be adapted into a measuring instrument for DX. For example, the original survey item "I would imagine that most people would learn to use this system very quickly" can be describe as "I would imagine that most software developers would learn to use this software very quickly".

The conceptual framework designed by Fagerholm and Münch mentioned in the previous section can also be used for evaluation. For example, Lee and Pan (2021) identified a series of questionnaire items which are sub-constructs of cognitive, emotional, and behavioral factors of DX, to evaluate developer experience toward a DL platform. They applied multiple statistical methodologies to assess the reliability and validity of the proposed measurement model, such as Delphi survey which was intended to determine the final survey based on preliminary items. Descriptive statistics and exploratory factor analysis are also widely performed to verify the data values. The results of this study proved that the evaluation of DX of a DL platform can start with three perspectives: the value developers perceived from the platform, affection and willingness to use the platform.

Mikkonen (2016) and Kuusinen (2016) created a survey to evaluate DX of Qt Creator and Vaadin Designer respectively. Survey contents were extracted from the Short Dispositional Flow State Scale (SDFS-2), parts of the Intrinsic Motivation Inventory (IMI), the short version of the AttrakDiff-2 (SAD-2) and the Developer Experience Scale (DEXI). They finally analyzed the correlation between survey results, and the overall UX and assessment and overall ability of target tool to fulfill the needs of the respondent. Based on the findings, both studies suggested practical ways to design corresponding tools, including practical design of features.

Ichario and Maarek (2020) studied the impact of terms of service and privacy policies on the DX of Web API via ground theory and card sorting methods. The results call for the establishment of standards and guidelines to improve the DX and avoid risk to API users. IDE is also one of the most popular tools that are evaluated because it provides relevant plug-ins to support developers' programming efforts. Kline et al. (2002) examined DX of IDE for Java via heuristic and psychometric methods and discovered the issue of program learnability and visibility that affect the DX.

2.4 Low-Code Development Platforms

In recent years, software development has become more and more flexible. Implementing a simple application doesn't need much effort anymore. Building automatic tools to allow non-professional people to create software systems is gradually becoming popular. The Low-Code Development Platform (LCDP) is such a tool. Here is the definition from Waszkowski (2019):

> A set of tools for programmers and non-programmers. It enables quick generation and delivery of business applications with minimum effort to write in a coding language and requires the least possible effort for the installation and configuration of environments, and training and implementation(Waszkowski, 2019).

Waszkowski also pointed out three main features of the LCDP: databases, business processes and user interface. The LCDP simplifies the development process, reduces the learning costs and visualizes the coding environment in order to allow developers spend less time on coding and focus on their goals. Dahlberg (2020) interviewed six participants with prior experience with the LCDP and asked their opinions and feelings about the platform. The positive findings include increasing productivity, more attention to the goals and common understanding between customers and developers. In contrast, the negative findings show compressed creativity, limited freedom and the safety risk for information and collaboration.

Sahay et al. (2020) compared several LCDPs to identify the typical functionalities and features. The goal is to increase the understanding and help users select the most appropriate platforms to meet their own needs. They discovered that the LCDP was built in two major ways: UI to Data and Data to UI. They also discussed four characteristics that have a large impact on DX: 1) interoperability, creating standards and a friendly ecosystem in this domain in order to mitigate issues caused by closed sources; 2) extensibility, allowing developers to customize capabilities to reduce the design constraints; 3) learning curve, which is still less intuitive for citizen developers and 4) scalability, running on the cloud to manage intensive computing.

2.5 Developers Using Low-code Development Platforms

LCDP is one of forms of End-User Development (EUD) which is a new research topic emerging from software engineering (SE) and HCI. EUD is defined as follows:

> "A set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact"(Lieberman et al., 2006).

Developers using LCDPs to create applications own diverse backgrounds, such as educations, skills or preferences. They rely on LCDPs to achieve their creative goals. They do not need to go through conventional development cycles to build applications from 0 to 1 as traditional developers do. They take advantage of the convenience of the platform and spend less effort on the infrastructure. The infrastructure and the environment of LCDP have a significant impact on developer experience. The main target of this kind of platforms is supposed to be citizen developers (Sahay et al., 2020) who have less background in software development. However, due to the complex technique and lack of tutorials of these platforms, their adoption from citizen developers is affected. Faced with the ever-changing needs of world, many companies are abandoning lengthy development processes, opting for low-code development and focusing on innovation to save time and labour. In this case, more and more professional developers prone to use LCDPs and their background helps them get starting quickly.

To support the diversity of end users, another developer group of LCDPs called end-user developers plays a significant role. End-user developers of LCDPs refer to those are responsible to develop and adapt the platform in order to mitigate the complexity of the platform and allow citizen developers to build their own applications. They focus more on low-level details, creating components and simplifying logic for end users.

2.5.1 Differences to Traditional Development

Compared to traditional development platform, the LCDP provides different experience for developers. Traditional development platforms are primarily for professional developers. Although IDEs have provided various encapsulated functions to reduce developers workload, developers still need to code from start to finish. Most LCDPs can build user interface via drag-and-drop and process data automatically. However, they also have many limitations depending on the platform. LCDPs have little customization since the features are designed by the platform. Developers can only make use of existing features. For example, some developers may prefer a free layout, placing components wherever they want on the interface. But the platform they use might only support top and down layout. Learning what kind of functions they can implement is an essential step before getting started.

Another great difference between two types of development is developers' background. Traditional development requires developers to have proficient code skills. More and more frameworks are emerging to make work easier, so developers must constantly learn and update their knowledge. Instead, LCDPs are committed to engaging more and more citizen developers as their users. To achieve this, they have been simplifying the development process. As long as you have basic computer skills, such as sending emails and typing, you can easily develop your own applications. Table 2.1 shows the differences among coding knowledge, development, customization, deployment, agility and maintenance.

Property	**Low-code Development**	**Traditional Development**
Coding Knowledge	Basic knowledge required	Advanced coding skills and knowledge of multiple web framework
Development	Rapid 1-3 weeks	Slow Around 2-8 months
Customization	Very limited customization	Tailored to meet your specific requirements
Deployment	Fast but limited platforms	Free to deploy to any and all platforms.
Agility	- More agile - Make changes quickly. - New releases will be quicker and any errors can be backtracked and resolved immediately	- Changes can be slow - Can be agile by following agile development practices
Maintenance	- Easy maintenance comparatively. - Automatically updated - Minimal resources in the regular maintenance	- Tough to regularly update and maintain - A dedicated team to update the code each time

Table 2.1: Low Code vs Traditional Development (Patel, 2021).

2.5.2 DX of Low-Code Platforms

Measuring and improving the DX of LCDP have also attracted the interest of researchers. For example, LCDP usually adopt Domain-Specific Languages (DSLs) which are visual modelling languages to hide low-level details about

developing and publishing applications. Henriques et al. (2018) attempted to improve the DX of a low-code platform, by developing a new version of DSLs for process modelling. They used mixed methods of interviews, "Physics of Notation", System Usability Scale (SUS) and the NASA Task Load index (TLX) for design and evaluation during the study. For the usability of the new model, they got satisfactory results: the semantic transparency, the correctness of responses and the SUS score increased while the TLX score decreased. At present, studies on the DX of LCDPs are scarce. On the one hand, LCDP is still a new concept. Companies and studies are working to make more people aware. They start with presenting its definition, advantages and disadvantages, and comparison with traditional development platforms. Doubts remain about its trustworthiness. A few citizen developers may still find it hard to learn and quickly give up. But we believe as more and more people become aware of LCDPs, developers' learning costs and experience will become more of a concern.

Chapter 3

Development of Questionnaire

The aim of this thesis is to measure developer experience of Low-code development platforms. We decided to use a questionnaire to achieve it. The development of the questionnaire consists of two stages. Firstly, we collected related question items from a stack of literature. They were selected based on characteristics of LCDP and the DX framework. Then Delphi study was used to gather opinions from a virtual panel of experts. They helped us to nuance and refine the content of the questionnaire.

3.1 Development of Preliminary Questionnaire

We did four steps to generate the initial questionnaire: 1) gather question items from existing studies and widely used surveys, 2) classify and categorize all question items, 3) select measurement dimensions that match the goal of our study, 4) determine the format of question items. Collecting question items help to understand the perspectives and methods of questionnaire design in current research.

3.1.1 Question Items Collection

We selected items with regard to their widespread use and relevance to DX. The study by Nylund (2020) is a multivocal literature review on DX which gathered research related to DX as much as possible. We started from this study and mainly chose factors from sections "Factors that improve/worsen the DX", "Experience influencers" and "Contexts of DX". The total was 17, encompassing a holistic view of the DX. The list of D influencers includes *mood & feelings*, *project onboarding*, *performance alignment work*, etc. This study facilitated our progress to search for literature associated with DX.

A more recent study conducted by Lee and Pan (2021) aimed to evaluate the sub-constructs of DX. Their survey was based on the conceptual framework of DX which is adapted from that of Fagerholm and Münch (2012) — Cognitive, Emotional and Behavioral. All question items are sub-constructs of these three facets. We extracted 20 items from their final results which nomological validity has already be proved, for example, *provides convenience to developers*, *provides developers with a variety of add-on options*, *the interaction between the platform and the developer is clear* and others.

Developer Experience Scale (DEXI) is also a significant material to measure DX. Its word-pairs were selected from the dataset of a meta-study, highlighting the features of software developmentKuusinen (2016). It contains 1 item for measuring general quality, 5 items for pragmatic quality and 6 items for hedonic quality. Because of the commonalities between UX and DX and the greater depth of UX research, in addition to these questionnaires directly relating to DX, other widely used surveys were considered : 1) the Short Dispositional Flow State Scale (SDFS-2) (Jackson et al., 2008), 2) Intrinsic Motivation Inventory (IMI) (Ryan, 1982), 3) Short AttrakDiff-2 (SAD-2) (Hassenzahl et al., 2010).

Short Dispositional Flow State Scale (SDFS-2). Flow is one of the key competencies that positive psychology has identified. Many research addressed its importance with regard to experiences, such as Jackson et al. (2008) and Mikkonen (2016). It is defined by full concentration on what one is doing, therefore one gains a positive subjective experience (Jackson et al., 2008). SDFS-2 survey contains 9 dimensions of the state of flow. And each dimension is described in detail in a complete sentence, for example, "I feel I am competent enough to meet the high demands of the situation".

Intrinsic Motivation Inventory (IMI). It is a multidimensional measurement of a subject's experience with an experimental task from a psychometric perspective. The original one is lengthy and repetitive, thus McAuley et al. (1989) presented several typical ones of all items by their experiment about the basketball free-throw shooting game. Mikkonen (2016) used the shorter one in their study, selecting 8 items from 4 subscales: Interest/enjoyment (3), Perceived competence (3), Effort/Importance(1) and Perceived choice (1).

AttrakDiff-2 (SAD-2). Hassenzahl et al. (2010) found a clear relationship between need fulfilment and pleasurable experiences with technology. They also categorized experiences by the main need they fulfil (i.e. 2 items for general measurement, 4 items for practical measurement and hedonic measurement separately). Kuusinen (2016) adopted this survey to their study to assess the DX of a particular IDE, Qt Creator. Since DEXI was inspired by SAD-2, they have the same formats and measurement dimensions.

Each item is in word-pair format.

3.1.2 Consolidation and Classification

After collecting items from existing surveys, there were 110 items in total. However, we found that there were many repeated dimensions although they might be in different formats. For example, both the item "I enjoy the tool very much" in IMI survey and the item "enjoyable - Unenjoyable" in DEXI scale express the happiness of users when they were using the tool. Therefore, we put similar items together to achieve de-duplication. The work was done on Miro, a visual collaboration platform. It makes communicating and presenting the result easily. Besides, we sorted items by the type of experience. It might be an initial impression of the platform, such as attractive appearance (immediate experience). It can also refer to the episodic experience occurs in a certain situation and the cumulative experience which is the result of prolonged or repeated usage of the platform over a period of time (Fagerholm, 2015). This study mainly focused on items related to episodic experience. But some items that apply to both episodic experience and cumulative experience were temporarily placed in episodic experience.

Under the episodic experience category, we continued to subdivide the items by cognition, affect and conation (Fagerholm and Münch, 2012). The previous definitions of these three categories are more general. To better accommodate the characteristics of episodic experiences and LCDPs, we have interpreted them in a more specific way for this study. Cognition refers to reasonable basis providing competitiveness, efficacy, value or problem solving for the developer; affect describes developers' emotion in a novel situation or given task while conation is the result or value obtained by a developer using the platform to complete a task. Moreover, since software development is also a social activity (DeMarco and Lister, 2013), some items were categorised as social impact which shows an aligned value across members in a team.

For greater clarity, the items under each subcategory were re-categorised using different labels. Cognition subcategory consists 7 labels: control, process, information, help, complexity, features and quality. For example, *Support for developers* belongs to help while *Limited - Extensive / Add-on options* is included in features. Both the cognition and affect subcategories include only one label, goal achievement and emotional experience respectively. We also simplified complex or long items to make it easier to be identified. For instance, *I have a good idea while I am performing about how well I am doing* becomes *Performance awareness.* At the end of classification, we reduced the number of candidate items to 35. Figure 3.1 shows the final result.

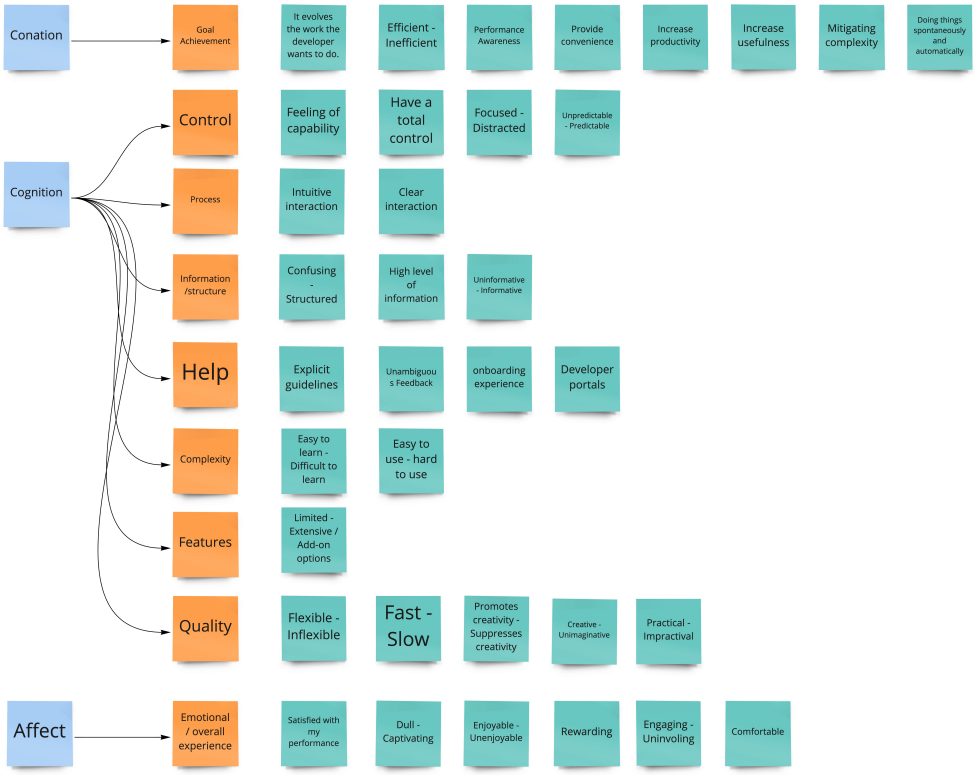

Figure 3.1: Consolidation and classification result of episodic experience items. Blue sticky notes: subcategories; orange sticky notes: labels under each subcategory; green sticky notes: individual experiences.

3.1.3 Selection and Formatting

We continued to reduce the number of items and leave the most relevant ones. The strategy of selection was based on the importance of using the platform. When developers use the platform, their primary concern is to accomplish their goals. Therefore completion is the first item. In addition to completion, efficiency is also significant, especially for LCDPs. Developers expect to create the application fast and easily. Well-being of developers was also an essential consideration in improving the DX. We were concerned about developers' feelings during the process. Furthermore, the quality of the platform also had a significant impact on the quality of work completed and the experience. Complexity, interaction, structure and support were finally taken into account.

It is also essential that all items are formatted consistently. We applied

the word-pairs format and 7-point scale between two antonyms because it was easy to read. Word-pair format takes participants less time to read these items. The reaction within seconds of the task being completed is the most realistic. Numerous literature discussed the selection of the points of scale. For example, the psychometric literature suggests that the more points on the scale the better, but diminishing returns occur at around 11 points (Nunnally, 1975) and Alwin (1997) also proved questions with more categories were both more reliable and more effective. However, Colman et al. (1997) stated 5-point and 7-point scale had little difference compared by empirical methods. In this regard, we decided to adopt 7-point scale as the format for our questionnaire. The questionnaire starts with a question: "What do you think of the experience of the process?" which leads participants to review their experience. Table 3.1 presents the preliminary questionnaire.

What do you think of the experience of the process?	
Conation	1. Smooth process - Rough process.
	2. High efficiency - Low efficiency
Affect	3. Enjoyable - Unenjoyable
	4. Satisfied with my performance - Unsatisfied with my performance
	5. Fully focused on my work - Unable to focus on my work
	6. Total control over what I was doing - Always lose control over what I was doing
Cognition	7. Simple - Complicated
	8. Easy to learn - Difficult to learn
	9. Intuitive interaction - Vague interaction
	10. Clear feedback - Ambiguous feedback
	11. Structured - Unstructured
	12. Sufficient information - Insufficient information

Table 3.1: Developer Experience part of Preliminary Questionnaire.

3.2 Delphi Study

The Delphi study aims to improve decision making by seeking the most reliable consensus of opinion from a group of experts (Dalkey and Helmer, 1963). The preliminary questionnaire was based on a literature review and our subjective analysis. We were concerned whether some of the issues were consistent with the nature of a LCDP. In addition, there may also be uncertainty

in the wording. Overly professional expressions may lack practicality while lengthy or overly concise expressions can easily be misunderstood. We expected each item to clearly represent what we mean, as the person completing the questionnaire may be a non-native speaker of English or non-professional in research.

3.2.1 Expert Profiles

The Delphi study was conducted between 6 April and 17 May and held on the zoom, an internet video communication platform. We have invited six experts with technical and research backgrounds. All participants were interested in DX and had related research on it. Three experts work as UX Designers for a LCDP, one is a principal user researcher in global large-scale research programs, and the remaining two have rich experience in industry and practice. They come from different regions, including Finland, America, French and Czech.

3.2.2 Procedures

The Delphi study was divided into two parts. The first part aimed to define the structure and format of the questionnaire (four experts involved), while the second part delved into its contribution to the LCDP (with the participation of two experts). In the beginning of the first part, experts were invited to share their opinions about the open question "You use a development tool to complete a specific task, what factors do you think that influence the episodic experience?" Their opinions came primarily from their user research experience and personal knowledge of LCDP. Some of those ideas really shed new light on our thinking and added new information that we may have missed. For example, a list of factors mentioned are listed below:

- The difficulty of task
- User interface
- Stability of the platform
- Cultural factors
- Developers' prior experience
- Integration of tools with the larger ecosystem
- Reusability of previous work

Then we presented our preliminary questionnaire. Experts were free to make any comments and suggestions on each item. Meanwhile, we explained and discussed controversial items. In addition, a number of follow-up questions were posed to the experts as appropriate which will help us better understand their perspectives. For example, when asked about the top three factors that most affect the experience, experts gave diverse answers. *Enjoyable - Frustrating*, *Fully focused on my work - Unable focused on my work* and *Satisfied with my performance - Unsatisfied with my performance* stood out among items, *Quick - Slow*, *Free - Restrained*, *Smooth - Difficult* were also mentioned.

After each session, we reviewed the opinion collected and refined the questionnaire. The new questionnaire would be presented at the next session. Iterations of collecting comments and then reviewing them allowed us to get constant feedback from experts on what we were revising. After four sessions, the feedback from experts went from a lot to less and less. Our questionnaire was greatly improved.

In the second part of the Delphi study, in addition to make comments on question items, two experts who are working for a LCDP also shared their views on how the questionnaire can contribute to the practical design of LCDPs. Their work experience helped to initially verify the validity of the questionnaire. Both of them gave positive feedback on the value of the questionnaire. They believed that the questionnaire had covered all the factors they know that can affect DX with LCDPs. Furthermore, they made a few suggestions from the usage scenario. Since developers needed to fill the questionnaire out immediately after completing a task, it might be a bit too long. In addition, although the questionnaire reflected different aspects of DX, developers' backgrounds needed to be taken into account. And some open-ended questions were also necessary if specific details are needed.

All experts have taken issue with the wording of some items, either that two adjectives for the same question are not actually antonyms, or that the adjectives may leave the user's understanding ambiguous as to whether the description is of a personal experience or of a tool. Based on this, we decided to divide the questionnaire into three parts to make each part clear and easy to understand.

In the first two Delphi sessions, experts had questioned the appropriateness of most phrases to describe the experience. Some phrases were too specific, for example, *Satisfied with my performance - Unsatisfied with my performance* emphasized developers' performance. On the one hand, developers might recall and evaluate their previous actions during the task. It seems developers have to rate themselves which can make it difficult for them to answer. On the other hand, experience is a subjective experience which is

affected by various dimensional factors. Defining all the factors is complex and can be burdensome for people who fill it out. Moreover, according to Taylor (2009) research of emotional resilience, 90 seconds is all it takes to identify an emotion. Chemicals are built and flushed in your body within 90 seconds which is called physiological experience of a situation. After this duration, you pay attention to your feelings with personal interpretation and you may slant the answer with social emotions and not self emotions. Therefore, the experience of a short time after completing a task is the most realistic feeling the developer has about the task. It is naturally generated without thinking and judging. We would ask developers to fill out the questionnaire as soon as they complete the task. In this case, it would be better for them to spend less time understanding the question. The simpler the question is, the better. As a result, we adjusted the format of the question item. Each one referred to one single emotion.

3.2.3 Results

We eventually got the final questionnaire (Table 3.2). Part 1 focuses on

Think about the session you just completed. Please enter what you consider the most appropriate description for your experience during the session?	
Conation	1. Easy - Difficult.
	2. Smooth - Exhausting
Affect	3. Satisfying - Dissatisfying
	4. In control - Lack of control
	5. Exciting - Boring
	6. Enjoyable - Frustrating
	7. Concentrated - Distracted
Cognition	8. Free to explore - Limited
	9. Productive - Unproductive
	10. Quick - Slow

Table 3.2: Developer Experience of Refined Questionnaire.

developers' personal experience, Part 2 and Part 3 describe the task and the tool separately. Evaluating the episodic developer experience is tied to the difficulty of the task and the use of the platform. Firstly, three aspects are divided into three parts to facilitate the user to understand the focus of each part. In addition, analyzing the correlation between task and developer experience, tool and developer experience is necessary in episodic experiences.

The following sections present the detail and interpretation of each part of the questionnaire.

Part 1 - Developer Experience

To guide developers how to fill this part of questionnaire, it starts with a description: *Think about the session you just completed. Please enter what you consider the most appropriate description for your experience during the session.* Ten groups of adjectives describe developer experience from different aspects. Developers indicate their experience by choosing to move closer to and away from an adjective. They consist of *Smooth - Exhausting*, *Easy - Difficult*, *Enjoyable - Frustrating*, *Satisfying - Dissatisfying*, *Distracted - Concentrated*, *In control - Lack of control*, *Exciting - Boring*, *Free to explore - Limited* and *Productive - Unproductive*.

Smooth - Exhausting refers to whether developers felt that the whole process went smoothly. Even the task is troublesome or they encounter impediments, they can find out the solutions and achieve the goal. If they often stop to think what to do next or they find problems hard to solve, their experience can be exhausting.

Easy - Difficult. Free from any obstacles, developers think completing the task is easy and relaxing. It may be due to the understandability of the task, the smooth of using the tool or the simplicity of finding the document, etc.

Enjoyable - Frustrating. Happiness of developers has been getting more and more attention in studies. Graziotin et al. (2017b), Graziotin et al. (2017c) and Graziotin et al. (2018) have studied the high impact and distinct consequences on development due to happy and unhappy developers. In this study, the well-being of developers will affect the result of the task as well.

Satisfying - Dissatisfying. Satisfaction is generated by whether or not something is provided that is needed or wanted. In other words, developers ability to achieve their goals by using the platform.

Distracted - Concentrated. Developers might be distracted by confusing interfaces or unaccustomed workflows. If they can not fully concentrate on what they are doing, not only does it take more time to complete the task, but their minds also need to constantly switch from the outside world to the task at hand.

In control - Lack of control. Kuusinen (2015) mentioned that the selection of tool affects the developer experience. Unfamiliar or uncomfortable tools can cause developers to feel out of control of the situation. In this study, although developers can't select the platform, their perceptions of low-code platforms and the practice of the platform will also lead to differences in their

experience.

Exciting - Boring is related to any surprising features and developers' interests on the platform. Most developers have a passion for advanced technologies and novel applications. Their desire of explore will affect developer experience to some extent by these factors.

Slow - Quick is one of the most important factors for low-code platforms. Developers expect to build an application well and fast via low-code platforms. It is not only related to the availability of the platform but also to the proficiency of developers. As a conclusion, Developers feel that the overall speed of completing tasks on the platform is fast.

Free to explore - Limited. There is more than one path to complete a task, and a good platform should support users to explore. Developers for low-code platforms often have diverse background and different level of programming knowledge. Their level of acceptance and proficiency in technology is different. For example, novice developers might prefer to use preset styles for components while developers with design backgrounds want to use advanced styles that reflect their personal level and ideas. Supporting the diversity of developer needs is also an invaluable factor in measuring the developer experience.

Productive - Unproductive is also fundamental in low-code development platforms. Whether or not developers can achieve their goals with a low-code platform and how well it is done will also make a difference to the developer experience. For example, if a developer wants to implement map positioning functionality, but there is no way for them to do so. If the application built by low-code platforms can only have limited functions, developers might tend to traditional programming. This question item can also be interpreted into reducing duplication of work and increase the efficiency. Reusability of components is a good example. Allowing developers to create and save their own components or function flows is one possible way to enhance developer experience.

Part 2 - Tool

Developers need to answer the question: *How much do you agree with the following statements regarding the tool?* Statements include *The tool was reliable.*, *The user interface was consistent.* and *The tool was quick to respond.*

The tool was reliable means the tool is consistently good in quality or performance. Comparing to traditional programming, some people remain skeptical of low-code development platforms. They might be afraid of the stability of the application once it is released. Therefore, how to win the trust of developers in the use is an essential goal. Otherwise stated, reliability will

compromise trustworthiness, which in turn affects the developer experience.

The user interface was consistent presents a specific but significant problem of platforms. Different from traditional programming, developers interact with the platform mainly by clicking, dragging and dropping. Visual language should make communications between developers and platform easy to understand. For instance, developers may be confused if they find that the same meaning is represented by different icons. Their experience will be affected too.

The tool was quick to respond. As we mentioned before, quick build is an advantage of low-code development platforms. Quick to respond is also salient in measuring the developer experience. If the developer clicks a button and it takes a long time for the feedback to appear, he/she might lose patience.

Part 3 - Task

Since developers fill up the questionnaire after completing a task. The design of tasks play an essential role of measuring episodic developer experience. In order to analyze the results of the questionnaire, it is necessary to understand the perception of the task from the developer's point of view. Similar question *How much do you agree with the following statements regarding the task?* leads the following statements: *The task was easy to understand.* and *The task was easy to complete.*

The task was easy to understand. Clarity of what to do is a prerequisite for getting things done. In this study, even though researchers decided on the content of the task. Participants did not necessarily agree exactly with the researcher on the task due to cognitive bias. Additionally, participants from different backgrounds have varied perceptions of the task.

The task was easy to complete. Task completion, difficulty of the task and developer experience do not necessarily have cause-and-effect relationship. Understanding how they impact each other requires to know the difficulty of the task from the developer's perspective.

Chapter 4

Validation

We conducted a validation procedure to ensure that the questionnaire we created accurately measures what it aims to do, regardless of the respondents. Valid questionnaire helps to collect better quality data with high comparability which reduces the effort and increase the credibility of data. The goal of the questionnaire is to measure the developer experience when people using a low-code development platform to complete a task. Therefore, we invited participants to do tasks within a specific time and filled out the questionnaire. At the same time, we would observe their behaviour during the process. After collecting the data, we compare the results of the questionnaire with the observations to see if there is consistency or any correlation.

4.1 Task Design

When we created the questionnaire, we assumed that the difficulty of tasks might affect the developer experience. We designed three tasks with different level of difficulty: The first task is the easiest and almost everyone can complete it without hesitation. The second task is harder, but some participants would still be able to complete it independently. The last one increased the difficulty again and fewer participants would feel confident completing it. All tasks would be done on a SAP AppGyver, a Finnish no-code development platform of a kind for LCDP. Participants would build a simple To-do List application after completing all tasks. We hoped participants would find it fun to do the tasks and develop an interest on low-code development. However, each task will be independent so that participants can have the same starting points for each task. Even though they fail to complete the task, it won't affect the following tasks. Moreover, we will take main features of low-code development platforms into consideration in our tasks. The top

three important features of low-code development platforms include: 1) Create user interface by drag-and-drop; 2) Load data from database to pages; 3) Create logic functions visually. If participants hardly use the platform, they can get an overall impression of the concepts during the process.

4.2 Participants

One of the ambitions of LCDPs is to enable people from different backgrounds, including professional developers and citizen developers, to build their own applications without much effort. In this case, LCDPs have to take not only different capabilities but also various needs into account. For example, designers might be more concerned about the appearances of applications and hope the platform to offer more fancy components or styles. The learning costs for professional developers and citizen developers are generally not the same. Their experience in using the platform will also be distinct. Therefore, we created a screening survey to get 9 different groups participants (Table 4.1) to study how work or professional backgrounds affect the developer experience in using the LCDP.

Experience of LCDPs, experience of design tools and programming experience are three main influencing factors that we took into consideration when selecting participants for the test. Participants who regularly use low-code development platforms might be familiar with the concept of the platform during the test and perform better while participants who only used low-code development platforms a few times or never used might be less confident. But if they have experience of design tools, they might find some commonalities between app builders and design editors. Both of them allow users to drag and drop components on the user interface and adjust the appearance by editing the properties. Participants with programming experience might also make use of their prior knowledge to understand the database and the logic functions. Overall, participants with different prior knowledge and experience are likely to get different results and experience during the test.

The screening survey received 83 answers in two weeks. They were divided into 9 groups and 2 - 4 participants were finally selected in each group,. We have arranged two rounds of testing to ensure that even if someone is absent or fails the test, we can re-invite new participants in the next round. 20 invitations were sent out in the first round and 12 tests successfully completed. Then, we got another 7 test sessions in the second round. It took three weeks to finish all test sessions. The participants were all English speakers and mainly from Asia, Europe, Africa and the USA. 10 of them are students while the rest are workers. Their fields of work or study include:

Group	LCDP experience	Design tool experience	Programming experience	Count
A	Regularly use	All level	All level	3
B	Occasionally use	No	All level	1
C	Occasionally use	No	No	2
D	Occasionally use	All level	All level	2
E	No	No	No	2
F	No	All level	No	2
G	No	No	A few years	2
H	No	No	Many years	2
I	No	All level	All level	3

Table 4.1: Participant Group.

research (4), software development (4), engineering (4), design (3), teaching (1), materials (1), and new media (1). Upon completion of the test, each participant was compensated with a gift card worth €20.

4.3 Procedure

Three pilot testings were organized to ensure that the tasks were rational. Participants who were involved in pilot testing were a User Experience designer, a User Researcher and a Data Analyst. After each test, I adjusted the settings (level of difficulty, duration and content) of the tasks based on the feedback. Pilot testing was necessary because it was hard to determine the difficulty of the tasks especially for those who had never used such a platform before, the test was easy to fail at the very beginning. Participants would also feel anxious during the tests. Pilot testing helped us find out potential problems and adjust them until the design of tasks meet the goal of our test. Due to time zone difference, participants chose time slots that suited their schedules when receiving the invitation. They were also asked to sign the consent form if they agree with our requirements. All requirements and procedure comply with the regulations of Aalto University Research Ethics Committee. Some requirements are listed as follows:

- Use computers/laptops
- Use Zoom, a video platform
- Share their screen when they are doing the tasks

- Turn on camera and microphone since it is an interactive test
- Network is stable

The test consists of three parts: introduction (5 minutes), a guided tour(20 minutes) and three independent tasks (35 minutes). The first task is too easy to reflect their experience, so participants only need to fill out the questionnaire after the second and the third task. They were required to answer the questionnaire within 30 seconds after completing the task, so the data gives a more realistic picture of their experience during the process.

In the beginning of the test, participants are informed that the test is not testing their skills, it will be fine if they fail to complete the task. But we encouraged them to try their best. The results can be affected by many factors, for example, the difficulty of the task, the functional design of the platform, the introduction in the guided tour or they need more time to get familiar with the platform. We care mostly about their real experience during the test.

The intention of the guided tour is to introduce the platform to participants. We realized that the low-code development platform required users' efforts to use it. Most of participants have never used such platforms, they might feel confused and nervous using a new tool for the first time. Participants get to know the platform in the guided tour from four aspects: the interface of the app builder, create user interface, connect data to the interface and create logic. We explain each part of the guided tour by building up a simple To-do list application. By interacting with participants, most of them become relaxed and feel confident for following tasks. They are also free to ask questions to better understand.

The independent tasks are to add new features to the applications built in the guide, such as adding new tasks and delete tasks. Participants share their screen to allow us to observe their behaviours. The observation includes four categories:

- ATTEMPT TO DO - Participants are on the right way but failed
- LOOK UP - Participants look up the instructions or documentations
- THINK - Participants stop and get lost in thought. Also includes clicking without purpose
- PROMPT - Participants fall into trouble and ask for help. We prompt the next step by asking them how they think about the task.
- RESULT - 0: fail; 1: success, following prompts; 2: success.

Observation is beneficial to analyze the correlation between developer experience and the actions. By standardizing the observation procedure, it helps minimize observer bias.

4.4 Data Analysis

We first identify differences in how participants felt across tasks and then present the Mann-Whitney U Test to differentiate participants from different groups, the Kendall's Tau correlation to analyze the correlations between various perspectives, and continue with Intraclass Correlation Coefficient (ICC) to test the consistency among participants with similar backgrounds, finally with findings from the observations at the end of this section.

4.4.1 Descriptive Statistics

By comparing the scoring details of the two tasks, we made a simple assessment of the validity of the scoring. We designed three tasks with different levels of difficulty. The first task was just for warming up. Only the second and the third task were used for analysis and 7-scale rated experience, the quality of the tool and the ease of the task from 1 to 7, for example, when rating the smoothness of the process, the larger the score, the smoother it is while the smaller the score, the more exhausting the process is. With regard to task completion, 12/19 people completed task 2, with 5 of them following the prompts. 5/19 succeed in task 3, but only 2 of them completed it independently.

Table 4.2 presents the average scores of task easiness. Responses to the ease of completing the task proved that task 2 is easier to complete than task 3. This finding is consistent with our direction of task design. In contrast, participants considered task 3 to be easier to understand. It can be explained from our observation, when doing the second task, participants needed time to understand the task, but due to the continuity and similarity of the tasks, they could more easily understand the third task.

Participants rated the platform as well. All three evaluation perspectives on the platform showed variability. The user interface consistency and the tool response had larger differences than the tool reliability between two tasks. Although both tasks examined the main functions of the platform, the detailed interactions with the platform were different. And the ease of tasks would also be likely to affect users' attitude towards the platform.

In terms of individual experience factors, Figure 4.1 depicts the average scores in two tasks. *Smooth - Exhausting* and *Easy - Difficult* were low and

had the greatest differences of all factors between two tasks. Most participants complained both tasks were difficult and felt exhausted after tasks. The findings were also in line with their rating of the ease of task completion that task 3 was more difficult. *Enjoyable - Frustrating* received a very varied feedback as well. It was acceptable that doing a difficult task would feel more frustrated. All experience factors got higher scores in task 2 except for *Satisfying - Dissatisfying*,. It seemed that hard task could provide more satisfaction. People might be more satisfied about their performance during the process if they solved a complex problem. *Free to explore - Limited* and *Exciting - Boring* looked nearly the same. On the one hand, the designed tasks involved relatively similar functions and failed to reflect the differences in the exploration of features. On the other hand, participants' experience might also be affected by personal will and reward. They were invited to do the test and excited about the involvement in the topic as well as the compensation provided.

Items	task2	task3
The task was easy to understand	4.842	5
The task was easy to complete	4.158	3.211

Table 4.2: Average score of task easiness in task 2 and 3. N = 19.

Items	task2	task3
The tool was reliable	5.053	5.105
The user interface was consistent	5.158	4.842
The tool was quick to respond	5	5.368

Table 4.3: Average score of tool in task 2 and 3. N = 19.

4.4.2 Mann-Whitney U Test

We run Mann-Whitney U Test to compare whether participants who evaluated the platform or the task high and those who evaluated them low have significant different feelings. We compared responses of those developers who assessed these factors more positively (scores>4) and those who assessed them more negatively (scores<4). Meanwhile, we tested task 2 and task 3 separately to avoid bias since they were different. However, for the platform, there were no statistically significant differences between those evaluation perspectives (tool reliability, user interface consistency and tool response) in both tasks. Although there may be significant differences in one of the tasks,

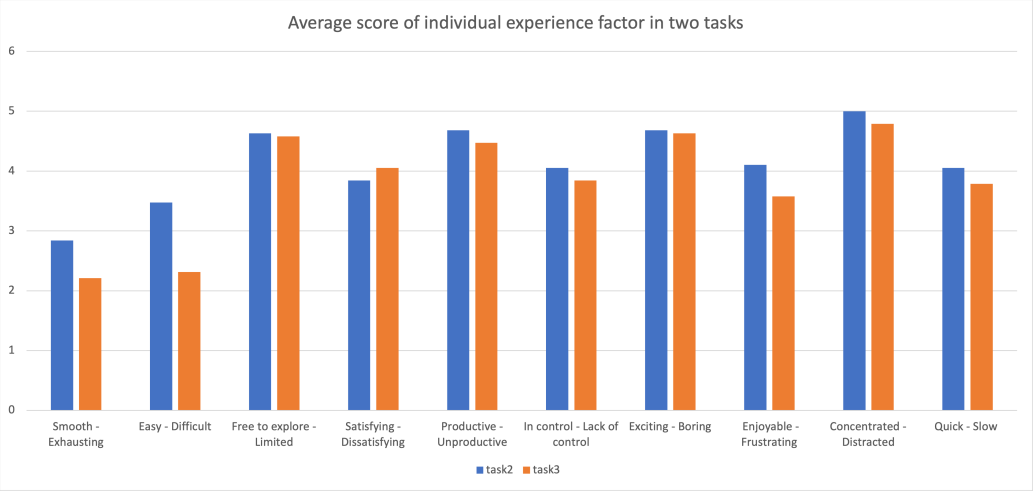

Figure 4.1: Average score of individual experience factors in two tasks.

they do not apply to the other. For example, in task 3, those respondents with high experience satisfaction perceived the platform to be highly reliable more often than those with low experience satisfaction, whereas this was not evident in task 2. This indicated the difficulty of the tasks played an essential role of measuring episodic experience. In the following analysis, we would talk more about experience items that had changeable performance caused by the difficulty of the tasks.

The same applies to the evaluation of the understanding of the task. None of the experience items were able to distinguish between users who found it easy or difficult to understand in both tasks. However, 3 out of 10 word pairs (Table 4.7) supported the separation between participants who found the task easy to complete and those who found it difficult. It means if participants feel easy, in control or excited when doing a task using the LCDP, they are much more likely to find the task easy to complete.

Items	task2_U	task2_p	task3_U	task3_p
Satisfying - Dissatisfying	15	n.s.	5	<0.05
In control - Lack of control	19	n.s.	4.5	<0.05

Table 4.4: Statistically significant Mann-Whitney U test results between respondent groups Tool Reliability_good and Tool Reliability_bad for all items. N = 19.

Items	task2_U	task2_p	task3_U	task3_p
Easy - Difficult	11.5	<0.05	27.5	n.s.
Productive - Unproductive	11.5	<0.05	28	n.s.

Table 4.5: Statistically significant Mann-Whitney U test results between respondent groups User Interface Consistency_good and Tool User Interface Consistency_bad for all items. N = 19.

Items	task2_U	task2_p	task3_U	task3_p
Easy - Difficult	11	<0.05	25	n.s.
Productive - Unproductive	14	<0.05	14.5	n.s.
Quick - Slow	5	<0.01	15	n.s.
In control - Lack of control	15	<0.05	22.5	n.s.

Table 4.6: Statistically significant Mann-Whitney U test results between respondent groups Tool Response_good and Tool Response_bad for all items. N = 19.

4.4.3 Kendall's Tau Correlation Analysis

From the results, for user interface consistency and tool response, few items had strong correlations with them. It was not surprising that these two assessments might be more biased towards objective facts which are not to address personal experience. However, another two assessments (tool reliability and task completion) reflected strong correlations with majority of individual items.

Tool reliability. Correlations between tool reliability and individual experience items are presented in Table 4.8. 7/10 (70%) of word-pairs correlated with the tool reliability in task 2 while 6/10 (60%) had significant correlations with tool reliability in task 3. What is notable is that 4/10 (40%) of items significantly correlated with tool reliability both in task 2 and task 3. Apart from *Smooth - Exhausting*, other experience items could predict tool reliability either in task 2 or task 3. Not just for tool reliability, this item had similar results in other aspects. Only for the ease of task completion, it showed a strong correlation in task 3. In addition to the possibility that it really had little to do with the platform and the task, we also speculated that it was the result of a more diverse user understanding of this experience factor.

User interface consistency. *Easy - Difficult*, *Free to explore - Limited* and *Productive - Unproductive* significantly correlated with user interface consistency in task 2. However, only *Concentrated - Distracted* showed a correlation in task 3 (Table 4.9).

Items	task2_U	task2_p	task3_U	task3_p
Easy - Difficult	8	<0.01	9.5	<0.05
Free to explore - Limited	15.5	<0.05	14	n.s.
In control - Lack of control	15.5	<0.05	8	<0.05
Exciting - Boring	15.5	<0.05	10.5	<0.05
Satisfying - Dissatisfying	32.5	n.s.	5.5	<0.01
Quick - Slow	16.5	<0.05	16	n.s.
Concentrated - Distracted	22.5	n.s.	5.5	<0.01
Enjoyable - Frustrating	22.5	n.s.	0.5	<0.001

Table 4.7: Statistically significant Mann-Whitney U test results between respondent groups Task Completion_good and Task Completion_bad for all items. N = 19.

Tool response. There was no correlation between experience items and tool response in task 2. What is interesting is that in task 3, 6/10 (60%) of items presented significant correlations with tool response (Table 4.10). It seemed that in difficult tasks, the developer experience was more affected by tool response. When participants were doing tough task, they might make a lot attempts to try to solve this problem and instant feedback from the platform could easily lead to good experience. Surprisingly, *Quick - Slow* did not correlate with this aspect. The quick response of tools and perceived quick seemed not completely the same. People feel quick might because they are immersed in the task and feel the time flies. Tough tasks can cause both diverse time perceptions. If they do torment and want to give up, then time will pass slowly, but if even they fail, they enjoy the process of exploration, the time still pass quickly.

Task understanding. In task 2 and task 3, 4/10 and 5/10 of items showed significant correlations with task understanding respectively. Only *Productive - Unproductive* and *Quick - Slow* showed significant correlations in both two tasks (Table 4.11). Productivity can be understood that in a specific time, how much work people has done. Using LCDPs, people can build applications quickly and efficiently. In the designed tasks, they can add features to the to-do list in bulk instead of one by one. If people feel more productive during the process, we believe they understand what they need to do and what features they need to use. And when they understand the task from the beginning, they can get start quickly.

Task completion. 7/10 (70%) and 9/10 (90%) of items significantly correlated with the task completion in task 2 and task 3 respectively. All items had significant correlations with the task completion either in task 2 or task 3 (Table 4.12). It is worth noting that compared to others, *Satisfying*

Items	task2_τ	task2_p	task3_τ	task3_p
Smooth - Exhausting	0.307	n.s.	0.199	n.s.
Easy - Difficult	0.434	<0.05	0.301	n.s.
Free to explore - Limited	0.331	n.s.	0.369	<0.05
Satisfying - Dissatisfying	0.331	n.s.	0.369	<0.05
Productive - Unproductive	0.507	<0.01	0.227	n.s.
In control - Lack of control	0.49	<0.01	0.479	<0.01
Exciting - Boring	0.545	<0.01	0.505	<0.01
Enjoyable - Frustrating	0.514	<0.01	0.597	<0.01
Concentrated - Distracted	0.407	<0.05	0.509	<0.01
Quick - Slow	0.418	<0.05	0.318	n.s.

Table 4.8: Results of Kendall's Tau correlation analysis between The tool was reliable assessment and individual developer experience items. Legend: n.s. = not significant. N = 19.

- *Dissatisfying* and *Enjoyable* - *Frustrating* had stronger relationship (τ = 0.644 and τ = 0.725) with task completion in task 3. The more they felt confident completing the task, the more satisfied and enjoyable they felt especially when the task was hard.

In addition, we used this method to assess whether any two sets of experience factors were relevant. The result for the former revealed that *Smooth - Exhausting*, *Easy - Difficult* and *Exciting - Boring* were strongly correlated with each other and all were positive. We can interpret this as they have positive impacts on each other. Furthermore, since we selected participants by their work or study backgrounds, backgrounds may also be an important factor in the experience. To know if individual experience factor would be related to any background, we analyzed any of the pairs. We put any one kind of background and one experience factor together for analysis. However, no significant correlation appeared in any pairs. In the next section of the consistency analysis, we will discuss further the influence of background.

With Mann-Whitney U test and Kendall's Tau correlation analysis, we found that some experience items had a very different performance in the two tasks. The result of Mann-Whitney U test showed that *Satisfying - Dissatisfying* can be used to differentiate participants those rated differently in tool reliability and task completion in task 3 and also the correlation analysis indicated significant correlations with all aspects of task 3 except for user interface consistency. But in task 2, no strong relationship with any aspects was shown. The similar finding was found for *Concentrated - Distracted*. It performed good correlation with all aspects of task 3 and only tool reliability

Items	task2_τ	task2_p	task3_τ	task3_p
Smooth - Exhausting	0.338	n.s.	0.337	n.s.
Easy - Difficult	0.38	<0.05	0.231	n.s.
Free to explore - Limited	0.376	<0.05	0.336	n.s.
Satisfying - Dissatisfying	0.324	n.s.	0.324	n.s.
Productive - Unproductive	0.378	<0.05	0.12	n.s.
In control - Lack of control	0.268	n.s.	0.365	n.s.
Exciting - Boring	0.359	n.s.	0.362	n.s.
Enjoyable - Frustrating	0.262	n.s.	0.343	n.s.
Concentrated - Distracted	0.29	n.s.	0.464	<0.05
Quick - Slow	0.321	n.s.	0.27	n.s.

Table 4.9: Results of Kendall's Tau correlation analysis between The user interface was consistent and individual developer experience items. Legend: n.s. = not significant. N = 19.

of task 2. And with Mann-Whitney U test, it was only able to distinguish task completion of task 3. On the contrary, *Easy - Difficult* showed a preference for task 2 in terms of the consistency of user interface and the response of the tool in Mann-Whitney U test. It also tended to correlate with all aspects of task 2 except for tool response in correlation analysis. This phenomenon was even more evident if only one analysis method was considered. *Productive - Unproductive* also showed a tendency for task 2. Overall, the difficulty of tasks had an impact on the experience items. Some experience items might be more valid to measure developer's experience for easy task while others are for hard task. There is a shift in the developer's propensity to experience when faced with tasks of varying difficulty. For example, faced with difficult tasks, developers can be more satisfied when they feel more confident to complete the task. But for easier tasks, even developers find the task easy, they may not feel satisfied since they do not think the task is sufficient to reflect their great performance. Another explanation can be akin to Redelmeier and Kahneman (1996)'s medical treatment experiment that the worst part and the final part of the experience was more largely remembered by patients. After completing the task 3 which was harder, participants had deeper impression than after task 2. Thus, some of experience items presented a clearer pattern.

Another interesting finding was that the tool response showed strong correlation with 6/10 experience items in task 3 but none in task 2. We reflected on the design of the tasks, and perhaps the problem was highlighted by the fact that developers spent more time on task 3 and interacted more

Items	task2_τ	task2_p	task3_τ	task3_p
Smooth - Exhausting	0.271	n.s.	0.31	n.s.
Easy - Difficult	0.208	n.s.	0.274	n.s.
Free to explore - Limited	0.202	n.s.	0.395	<0.05
Satisfying - Dissatisfying	0.049	n.s.	0.456	<0.05
Productive - Unproductive	-0.036	n.s.	0.029	n.s.
In control - Lack of control	0.086	n.s.	0.271	<0.05
Exciting - Boring	0.172	n.s.	0.449	<0.05
Enjoyable - Frustrating	0.137	n.s.	0.495	<0.05
Concentrated - Distracted	0.086	n.s.	0.382	<0.05
Quick - Slow	0.138	n.s.	0.351	n.s.

Table 4.10: Results of Kendall's Tau correlation analysis between The tool was quick to respond and individual developer experience items. Legend: n.s. = not significant. N = 19.

with the platform.

4.4.4 Intraclass Correlation Coefficient

ICC was used to find out if there was consistency in the results of participants with similar backgrounds. Table 4.13 presents the results of consistency for each group measuring the experience of the test. Comparison was not made within group B since only one participant belonged to it. As we can see, participants in group E and group F had similar experience in both task 2 and 3. Next we compared whether any two groups have a consistent experience. Table 4.14 only shows groups that have consistency. The complete results are in the appendix. The results indicated that participants in groups B, E, and F had consistent experiences in both tasks. But in task 3, group B and group E participants also have consist experiences with group G, respectively. Based on this finding, we compared group B, E, F participants together and also found consistency among these three groups. In terms of programming experience, participants in groups B, E, and F had none, while those in group G had a little. Other group participants had more. Thus, programming experience does make a difference to the episodic experience of the LCDP, but this effect has variability. The findings is acceptable after talking with participants about their work or study experiences. We found that their programming experiences were diverse, such as programming language and skills. Someone is mainly responsible for data analysis using existing libraries while someone focuses on algorithmic programming.

Items	task2_τ	task2_p	task3_τ	task3_p
Smooth - Exhausting	0.007	n.s.	0.163	n.s.
Easy - Difficult	0.42	<0.05	0.343	n.s.
Free to explore - Limited	0.165	n.s.	0.228	n.s.
Satisfying - Dissatisfying	0.237	n.s.	0.377	<0.05
Productive - Unproductive	0.41	<0.05	0.383	<0.05
In control - Lack of control	0.413	<0.05	0.271	n.s.
Exciting - Boring	0.214	n.s.	0.091	n.s.
Enjoyable - Frustrating	0.215	n.s.	0.465	<0.05
Concentrated - Distracted	0.114	n.s.	0.382	<0.05
Quick - Slow	0.484	<0.05	0.4	<0.05

Table 4.11: Results of Kendall's Tau correlation analysis between The task was easy to understand and individual developer experience items. Legend: n.s. = not significant. N = 19.

Developers writing operational logic and visual interface also have diverse understanding of workflow. Most of them thought their prior programming experience did not contribute to their usage of the LCDP. But one participant who is a front-end developer found it easy to do the task because the logic was almost the same. Furthermore, group G only showed consistency in task 3. We could understand that their programming experience might not be enough for them to solve hard tasks. Task 2 was simpler that they might be more confident than those without programming experience.

In the area of LCDP experience, no consistencies were found either between testers who used the platform regularly, or between those who used it occasionally or had never used it. Group E, F, G, H, I participants had no LCDP experience and only group E and F participants of them had similar experience, so we guessed that programming experience affected experience of other groups. More experiments are needed to support this. Participants introduced the platforms they have used, such as WordPress, TouchDesigner, and even mechanically related. Their working principles and processes are not the same as the platform involved in the test. When they were doing tasks, they had to learn and apply something different from the previous platform in a short time. As a result of this discrepancy, their experience compromised. Only one who had used the given LCDP regularly felt familiar with this app builder and completed tasks smoothly. Strictly controlling the type of LCDP in the further research might get better results.

Finally, design experience did not present any pattern in this study. Among testers with consistent experience, both experienced and inexperi-

Items	task2_τ	task2_p	task3_τ	task3_p
Smooth - Exhausting	0.298	n.s.	0.399	<0.05
Easy - Difficult	0.583	<0.01	0.363	n.s.
Free to explore - Limited	0.389	<0.05	0.37	<0.05
Satisfying - Dissatisfying	0.312	n.s.	0.644	<0.01
Productive - Unproductive	0.465	<0.05	0.372	<0.05
In control - Lack of control	0.495	<0.01	0.372	<0.05
Exciting - Boring	0.576	<0.01	0.519	<0.01
Enjoyable - Frustrating	0.389	<0.05	0.725	<0.01
Concentrated - Distracted	0.265	n.s.	0.412	<0.05
Quick - Slow	0.475	<0.05	0.478	<0.05

Table 4.12: Results of Kendall's Tau correlation analysis between The task was easy to complete and individual developer experience items. Legend: n.s. = not significant. N = 19.

Group	task2_ICC	task2_p	task3_ICC	task3_p
A	-0.123	n.s.	0.104	n.s.
C	0.075	n.s.	-0.051	n.s.
D	0.118	n.s.	0.232	n.s.
E	0.682	<0.05	0.553	<0.05
F	0.792	<0.01	0.457	<0.05
G	0.018	n.s.	0.471	<0.05
H	0.235	n.s.	-0.16	n.s.
I	0.031	n.s.	0.166	n.s.

Table 4.13: Results of Intraclass Correlation Coefficient for each group measuring the experience. Legend: n.s. = not significant.

enced users were included. We tended to conclude that design experience was not as influential as programming or LCDP experience.

4.4.5 Reflections on Observations

We recorded some key behaviours of participants during the test and got interesting findings. 42% and 68% of participants have the largest percentage of "attempt to do" action, they were frequently stuck before the correct step in task 2 and 3, respectively. When faced with difficulties, many participants preferred to keep trying the same action as they thought was right, and rarely changed their minds even when they were confused by the failure. They thought they knew the logic but the way of using the platform prevented them

Group	task2_ICC	task2_p	task3_ICC	task3_p
B & E	0.523	<0.01	0.726	<0.001
B & F	0.554	<0.01	0.634	<0.001
B & G	0.058	n.s.	0.581	<0.001
E & F	0.672	<0.01	0.532	<0.001
E & G	0.106	n.s.	0.570	<0.05

Table 4.14: Results of Intraclass Correlation Coefficient for each two group measuring the experience. Legend: n.s. = not significant.

from implementing it. A few professional developers believed they would complete the task better and more quickly by writing the code themselves. On the other hand, if they had no idea about how to do the task, they looked up the documentation, stopped to think or got anxious and clicked around aimlessly on the interface. These actions happened more frequently in task 3. In order to know their attitude toward the documentation, I observed the frequency of the action of looking up documentations. Unexpectedly, among those failed in task 2, only 1/5 participant looked up documentations frequently while others preferred to think about it on their own. Same in task 3, only 4/13 chose to solve the problem by looking up information and 2 participants didn't check the information at all. It seemed that participants did not like to look up documents although official documents were provided and suggested through the test process. Even some of them opened up the documents, they were confused about which keywords would lead them to the answers. And they closed it after a short hesitation. They responded that they were under much pressure when they needed to complete a task during a limited time, they were not confident and patient in finding answers quickly in documents which was full of information. Thus, they felt it was a waste of time. But they admitted that in everyday use, they would be willing to search for documents or check out how-to guides written by others if there is no time pressure.

Some specific terms were also difficult for non-specialists. For example, a few participants couldn't distinguish *placeholder* and *value* of an Input component. They bound the variable to the placeholder and wondered why it did not work. To sum up, many participants considered the lack of similar experience, such as front-end development, to be the main reason for their failure to complete their tasks. Forced to learn a new platform and do tasks immediately in a limited time constricting their performance.

Chapter 5

Discussion

A LCDP is a platform that provides developers with freedom and flexibility to implement creative ideas by easily and quickly building applications. Since the rapid iteration of the business, the use of LCDPs has become popular. However, as of today, no studies have been conducted on the episodic experience of the developers who use LCDPs, in terms of evaluating how they feel about and perceive their work in a given situation. These concepts are still relatively new. This thesis creates and validates a questionnaire to measure the episodic DX in the context of LCDPs. Meanwhile, the quality of the tool and the difficulty of the task were taken into account to discover the correlation with developer experience. The developer's experience varies with the developer's view of these different perspectives.

5.1 Research Questions Revisited

The research questions addressed in this study are as follows:

RQ1: Which experience factors are associated with the developers' episodic experience of LCDPs?

RQ2: How does background affect developers' episodic experience of LCDPs?

RQ3: What insights can be derived from this study for further research into episodic experience and LCDPs?

For RQ1, we started with three aspects of DX derived from the conceptual framework designed by Fagerholm and Münch (2012) and gave them new definition based on characteristics of episodic experience and LCDPs.

They were **Cognition** — reasonable basis providing competitiveness, efficacy, value or problem solving for the developer; **Affect** — an emotional state for developers in a novel situation or given task; **Conation** — the result/value obtained by a developer using the platform to complete a task. **Social Impact** was the fourth aspect that we considered significant but in this study it had been excluded due to the difficulty of implementation. We picked 6, 4 and 2 experience factors for each aspect from the existing literature. During Delphi sessions, combined with experts' suggestions, we refined the items in terms of comprehensibility, readability and applicability. The items related to efficiency, interaction, feedback, structure, information and learning were excluded and new items about exploration, production, easiness and excitement were added. The final totals of items for each aspect are 5, 3 and 2. All items can be seen in Table 3.2 The task-oriented test that followed demonstrated that all these items had more or less correlations with the tool or the task. They could be predictors of the quality of the tool and the ease the task to some extent although the more precise correlations should be further identified.

For RQ2, we attempted to understand whether programming experience, LCDP experience and design experience have impacts on episodic experience when participants using LCDPs to perform tasks. ICC analysis told us programming experience does make a difference to the episodic experience of the LCDP, participants without programming experience rated the experience items in a consistent manner, those with programming experience rated differently. We realized that programming experience was too vague to determine the influence. According to LCDP experience, no consistencies were found between testers, regardless of which group they were involved. What is known is that people with front-end experience are relatively comfortable using the platform. In the future, more controlled prior experience should be taken into consideration. Even with many uncertain conclusions, we found the role of design experience was not significantly helpful in completing tasks with the new LCDP. Among participants with consistent performance, some of them had design experience while others did not.

For RQ3, this study provides a good starting point for further research. First, let us review our findings. The results of Mann-Whitney U Test indicated that *Easy - Difficult*, *In control - Lack of control* and *Exciting - Boring* are able to differentiate between participant groups with low and high Task Completion Difficulty assessment. Of these three items, the latter two were also proved to be significantly correlated with the ease of task completion by Kendall's Tau Correlation Analysis. In addition, the correlation analysis revealed additional four items that had similar correlation: *Free to explore - Limited*, *Productive - Unproductive*, *Concentrated - Distracted* and *Quick -*

Slow. In terms of Task Understanding, *Productive - Unproductive* and *Quick - Slow* had strong correlation with it.

A number of items were correlated with the reliability of platform although their abilities to differentiate between participant groups were not successfully demonstrated. For example, *In control - Lack of control*, *Exciting - Boring*, *Enjoyable - Frustrating* and *Concentrated - Distracted* are associated with tool reliability. In other words, we might compare the reliability of different LCDPs, and the LCDPs that get higher feedback in these items might be more reliable. User interface consistency and tool response did not show correlations with any experience item. In this study, they could not be predicted by participants' feelings.

Despite the above findings, we still have a lot of questions in mind and need more in-depth research to further test their validity. For instance, why do some items perform differently across tasks, and is it really related to the difficulty of the task? What are the differences in the percentages between the different factors? Why did *Productive - Unproductive* show significant correlation with the ease of understanding the task although they seemed unrelated? We guess that productive platform makes users feel confident in understanding the task. Triangulation is a common approach to analyze data and solve problems (Wilson, 2006). We do not doubt that the current findings can easily be questioned. Multiple measurements should be applied to make the results more convincing and persuasive. Next we can start with these experience factors to validate their correlation with tasks and tools. Or we can also come up with ideas based on the problems identified so that they can be put into test. Most importantly, we can correct the omissions in this experimental design to make subsequent experiments more robust.

5.2 Contributions

Law et al. (2014) provided empirical evidence on the attitudes of the HCI community towards UX measurement that understanding why certain UX measures are taken and how they are used and interpreted to inform design and development decisions is essential. There is still debate on whether quantitative feedback can be more or less important as qualitative feedback. It also applies to DX. It might not be enough to use quantitative results as a basis for design, it is the interpretation behind the numbers that really determines the direction of the design. Thus, when talking about the contribution to the design of LCDPs, we need to be cautious. In this study, we tried to come up with some tips for the design of a specific feature of a LCDP. The validity of these tips may need to be further verified, and our

goal is to provide some insights for subsequent researchers to delve into this field.

In statistical analysis, we discussed those experience factors from two aspects: tool and task. In more detail, the tool consists of tool reliability, user interface consistency and tool response while the task consists of task understanding and task completion. Different experience factors had more or less correlation with each of them. Meanwhile, We think the subjects (tool and task) can be corresponded as platform and specific features in design field. For example, in terms of task completion, it can be treated as the difficulty of using a feature. Under this premise, we can translate our findings into design in two ways. Our first way is inspired by Google's HEART framework. Google proposed a statement when it was establishing a HEART (Happiness, Engagement, Adoption, Retention, and Task success) framework for user-centered metrics for web applications:

> "No matter how user-centered a metric is, it is unlikely to be useful in practice unless it explicitly relates to a goal, and can be used to track progress towards that goal (Rodden et al., 2010)."

It used a simple process to apply the framework to the design: Goal - Signal - Metrics. We set a goal and identify signals for achieving the goal, then select and monitor metrics that are related to these signals. Therefore, we can compare two versions of features, such as A/B test or evaluation after updating the version. For example, if version B gets higher score in *Easy - Difficult*, *In control - Lack of control*, *Exciting - Boring*, *Free to explore - Limited*, *Productive - Unproductive*, *Concentrated - Distracted* and *Quick - Slow* than version A, the feature in version B might be easier to use. Moreover, with the iteration of design, we can monitor the change of these experience factors to estimate the effect of the design. As a result, other similar conclusions can be drawn in the ease of understanding the feature and the reliability of the part of the platform associated with the feature.

The second way of explaining the experience factor scores is in the diverse direction to the first way. The goal of the optimizing the design of a LCDP feature is to improve the developers' experience, but we are easy to get lost in the product design. Generally, many qualitative methods can help us do that, like card sorting, interview, brainstorming, competitor analysis, etc. We can also make use of our quantitative data to decide our design direction. Take *In control - Lack of control* as an example, it has strong correlation with the ease of using the feature. We can think about how to improve the score of this experience factor in order to make users use the feature easily. We might be reminded of the 10 usability heuristics for user interface design (Nielsen, 2005). One of principles is **User Control and Freedom**, users

need a clearly marked exit to escape from current messy situation, so *Undo* and *Redo* actions are implemented. Similarly, there are also various ways to gain users' attention while they are using the platform so that the reliability of the platform can be improved. Watson and Sanderson (2007) suggested using sound to allow users to know what is happening now, for example, the file is downloaded successfully. Vertegaal et al. (2006) came up with magnifying information focused upon by the user, and attenuate peripheral detail to augment users' attention.

In addition to above two ways to involve experience factors into design, another more complicated way can also be taken into account. We made a rough calculation of adding up all experience factors and used the final score to correlate with task and tool. The result showed that it had significant correlation with tool reliability and task completion. Since our sample is too small, it is not suitable to use factor analysis. This result can't tell its validity, but if the sample is large, we can try to make a deeper calculation. Knowing how each experience factor contribute to a goal, an experience score can be calculated as the basis of evaluating the design. The experience score concretizes the abstract experience and can be used as a basis to prove the contribution of design to the product. One number can tell us a lot.

5.3 Reflections on Related Research

5.3.1 Low-code Development Platforms

One of the goal of LCDPs is to lower the barrier to developing software and allow citizen developers build up applications without much effort, the basic concepts and professional terms are still necessary. Thus, onboarding learning is one of the most important part of LCDPs. Making developers from different backgrounds and different level of acceptance to understand the workflow is challenging. For many of the LCDPs now available, users still need to spend much time and energy learning before getting started. During the test, one participant mentioned that sometimes he did not know how to look up documentation because the information did not match his needs. For example, he needed to add a new task to the to-do list, he had no idea where to start. If he is not sure of the logic to implement this function, such as from creating a variable to binding the variable to logic function, he would get lost in the documentation. So we get to thinking about the importance of this programming-like minds in the use of LCDPs. The gap between citizen developers and professional developers is nevertheless wide. Of course, narrowing the gap is something LCDPs are currently trying to

do. For professional developers, if they spend a lot of time on the platform's workflow, they would lose patience and easily get the idea that they would rather write their own code. To attract this group of developers, it may be useful to simplify the use of platforms or standardise workflows across platforms.

5.3.2 Episodic Developer Experience

Current research on episodic experience is scarce either for UX or DX. Instead, the majority of research paid attention to cumulative experience. People have used the product or tool for a long time. Their feelings and opinions about the product will be more comprehensive. The main points of their concern are certainly different from the concerns of the first time they used the product to solve a problem. Bobkowska (2013) built a model to use the term of intuitiveness to describe the UX episodic experience, the perception when people are involved in a new situation or using a new technique. It includes four perspectives: cognitive processes (users' familiarity of the technique), motivations (use the technique to solve a problem), actions (actions lead to good or poor results) and emotions (positive or negative attitudes). In this study, we found the similarities of episodic experience between UX and DX. For example, prior knowledge plays an important role when performing the task on the LCDP. One participant with experience in front-end development and another who had used the same platform performed best, also giving high scores on most of the experience items. The background might not be as important as in the cumulative experience since users are already familiar with the platform.

A lot of overlap between cumulative and episodic experiences. Previous research hardly distinguish them specifically. However, For cumulative experience, the price and community are important factors while in episodic experience, they are not much. Motivation is another rather different aspect between the two experiences. Users seek a solution to a specific problem while they may hope to achieve a bigger goal in a long-term usage of the platform. For example, people's intention to use the LCDP is to build an application which is a long-term goal. The goal will be achieved by several episodic goals which are to implement various features of the application. Motivation also influence the result of the test. Participants did not have expectations to this platform since it was their first time to use it, not to mention their emotional attachment to the platform. They were just required to do the task even they might not be interested to the task. Based on these facts, we ignored factors like *cheap - premium* or *I use the platform because I have no choice.*

Concentrated - Distracted is the only item that was more or less strongly correlated with all aspects of the platform and the task either in the task 2 or 3. This shows that it cannot be overlooked. Users' attention during a event determines the time, efficiency and the completion of the task, therefore, influencing their feelings. However, Nylund did the comprehensive literature review on developer experience but few research mentioned this factor. Jackson et al. (2008) addressed the concentration on task regarding the dimensions of state of flow, although in the study of assessing DX of a Graphical User Interface Designer (Kuusinen, 2016), it was not significantly correlated with developers' overall user experience. Thus, further differences in the role of attention in episodic and cumulative developer experiences should be identified.

5.4 Limitations

Prior knowledge is a significant influencer when studying episodic developer experience of LCDPs. One of the goals of this study is to discover the impact of prior knowledge on using the LCDP to perform tasks. Due to the lack of clear definition and control of participants' background, the correlation between background and experience is not significant. In further research, the selection of participants needs to be more careful. Controlled starting point of participants might contribute to the success of test. One possible way can be selecting participants with or without front-end development experience since it is close to the workflow of LCDPs. Another way can be asking a half of participants to use another very similar LCDP for some time before performing tasks on the tested platform while the other half are not.

The other apparent threat to the validity of the study is undoubtedly the small sample, especially only 2 or 3 participants in each group. More authoritative result will come from larger sample sizes. Similarly, the number of tasks also prevented us from getting more convincing conclusions. Both diversified tasks and repeated tasks are necessary in the study of episodic experience.

The test only covers one LCDP, thus, further studies can compare several LCDPs to evaluate the correlation between experience item and tool reliability. For example, all participants do the same tasks in different platforms. Their scores of experience might be helpful to address the impact of platforms on experience.

Chapter 6

Conclusions

This thesis created a questionnaire to measure episodic developer experience of a low-code development platform. The questionnaire consists of experience items from previous studies and we invited experts from related industries to help shape the final content of the it. Then we used task-based test to predict the correlation between those experience items and the tool reliability, user interface consistency, tool response, task understanding and task completion, respectively. At the same time, the influence of prior background was validated by the test.

The result shows that the questionnaire identified several influencing factors that are helpful to predict episodic DX of LCDPs. All the experience items in the questionnaire were more or less related to the quality of the tool and the difficulty of the task. In terms of impact of developers' background, prior programming and LCDP experience played an important role in measuring episodic experience of LCDPs but more clear definition and control of background are needed to verify this conclusion. Design experience showed little influence in this study. Finally, the findings in this study can be used to provide insights for further research into similar field. Starting with in-depth analysis of specific experience factors is a good direction. And questions exposed in this study can also lead us to think about the real reason and design appropriate study to validate.

In conclusion, future research could focus on episodic experience or low-code development platforms. Many topics are still worth exploring. How to better measure episodic experience? What kind of internal or external threats are there while conducting tests to measure episodic experiences? How to reduce or avoid those threats? How to reduce the learning costs of low-code development platforms so that more people can get started more easily? Is there a need to set a uniform standard for low-code development platforms? We look forward to seeing how episodic experiences and low-code

development platforms will evolve.

Bibliography

Duane F Alwin. Feeling thermometers versus 7-point scales: Which are better? *Sociological Methods & Research*, 25(3):318–340, 1997.

Nathan Baddoo, Tracy Hall, and Dorota Jagielska. Software developer motivation in a high maturity company: a case study. *Software process: improvement and practice*, 11(3):219–228, 2006.

Anna Bobkowska. On explaining intuitiveness of software engineering techniques with user experience concepts. In *Proceedings of the International Conference on Multimedia, Interaction, Design and Innovation*, pages 1–8, 2013.

Andrew M Colman, Claire E Norris, and Carolyn C Preston. Comparing rating scales of different lengths: Equivalence of scores from 5-point and 7-point scales. *Psychological Reports*, 80(2):355–362, 1997.

Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, 2004.

Daniel Dahlberg. Developer experience of a low-code platform: An exploratory study. Master's thesis, UmeÃ¥ University, Faculty of Social Sciences, Department of Informatics, 2020.

Norman Dalkey and Olaf Helmer. An experimental application of the delphi method to the use of experts. *Management science*, 9(3):458–467, 1963.

Tom DeMarco and Tim Lister. *Peopleware: productive projects and teams*. Addison-Wesley, 2013.

Fabian Fagerholm. *Software developer experience: Case studies in lean-agile and open source environments*. PhD thesis, University of Helsinki, Faculty of Science, Department of Computer Science, Helsinki, Finland, 2015.

Fabian Fagerholm and Jürgen Münch. Developer experience: Concept and definition. In *2012 international conference on software and system process (ICSSP)*, pages 73–77. IEEE, 2012.

Bill Gillham. *Developing a questionnaire*. A&C Black, 2008.

Elizabeth Goodman and Mike Kuniavsky. *Observing the user experience: A practitioner's guide to user research*. Elsevier, 2012.

Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. On the unhappiness of software developers. In *Proceedings of the 21st international conference on evaluation and assessment in software engineering*, pages 324–333, 2017b.

Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. Unhappy developers: Bad for themselves, bad for process, and bad for software product. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 362–364. IEEE, 2017c.

Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. What happens when software developers are (un) happy. *Journal of Systems and Software*, 140:32–47, 2018.

Marc Hassenzahl and Noam Tractinsky. User experience-a research agenda. *Behaviour & information technology*, 25(2):91–97, 2006.

Marc Hassenzahl, Sarah Diefenbach, and Anja Göritz. Needs, affect, and interactive products–facets of user experience. *Interacting with computers*, 22(5):353–362, 2010.

Henrique Henriques, Hugo Lourenço, Vasco Amaral, and Miguel Goulão. 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*, pages 200–210, 2018.

Aidah Ichario and Manuel Maarek. Vision: Investigating web api developer experience in relation to terms of service and privacy policies. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 166–171. IEEE, 2020.

ISO. Ergonomics of human-system interaction – part 11: Usability: Definitions and concepts. Standard, International Organization for Standardization, Geneva, CH, March 2018.

Susan A Jackson, Andrew J Martin, and Robert C Eklund. Long and short measures of flow: The construct validity of the fss-2, dfs-2, and new brief counterparts. *Journal of Sport and Exercise Psychology*, 30(5):561–587, 2008.

Timo Kaltio and Atte Kinnula. Deploying the defined sw process. *Software Process: Improvement and Practice*, 5(1):65–83, 2000.

R Kline, Ahmed Seffah, Homa Javahery, M Donayee, and Juergen Rilling. Quantifying developer experiences via heuristic and psychometric evaluation. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, pages 34–36. IEEE, 2002.

Vijay Kumar and Patrick Whitney. Faster, cheaper, deeper user research. *Design Management Journal (Former Series)*, 14(2):50–57, 2003.

Kati Kuusinen. Software developers as users: Developer experience of a cross-platform integrated development environment. In *International Conference on Product-Focused Software Process Improvement*, pages 546–552. Springer, 2015.

Kati Kuusinen. 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*, pages 215–233. Springer, 2016.

Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold POS Vermeeren, and Joke Kort. Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 719–728, 2009.

Effie Lai-Chong Law, Paul Van Schaik, and Virpi Roto. Attitudes towards user experience (ux) measurement. *International Journal of Human-Computer Studies*, 72(6):526–541, 2014.

Mary Lebens, Roger J Finnegan, Steven C Sorsen, and Jinal Shah. Rise of the citizen developer. *Muma Business Review*, 5:101–111, 2022.

Heeyoung Lee and Younghwan Pan. Evaluation of the nomological validity of cognitive, emotional, and behavioral factors for the measurement of developer experience. *Applied Sciences*, 11(17):7805, 2021.

Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-user development: An emerging paradigm. In *End user development*, pages 1–8. Springer, 2006.

Kurt R Linberg. Software developer perceptions about software project failure: a case study. *Journal of Systems and Software*, 49(2-3):177–192, 1999.

Jennifer L Martin, Daniel J Clark, Stephen P Morgan, John A Crowe, and Elizabeth Murphy. A user-centred approach to requirements elicitation in medical device development: A case study from an industry perspective. *Applied ergonomics*, 43(1):184–190, 2012.

Edward McAuley, Terry Duncan, and Vance V Tammen. Psychometric properties of the intrinsic motivation inventory in a competitive sport setting: A confirmatory factor analysis. *Research quarterly for exercise and sport*, 60(1):48–58, 1989.

Tommi Mikkonen. Flow, intrinsic motivation, and developer experience in software engineering. *Agile Processes in Software Engineering and Extreme Programming*, page 104, 2016.

Jakob Nielsen. Ten usability heuristics, 2005. `http://www.nngroup.com/articles/ten-usability-heuristics/`. Accessed 31 August 2022.

Francisco Nunes, Paula Alexandra Silva, and Filipe Abrantes. Human-computer interaction and the older adult: an example using user research and personas. In *Proceedings of the 3rd international conference on PErvasive technologies related to assistive environments*, pages 1–8, 2010.

Jum C Nunnally. Psychometric theory - 25 years ago and now. *Educational Researcher*, 4(10):7–21, 1975.

Anders Nylund. A multivocal literature review on developer experience. Master's thesis, Department of Computer Science, Aalto University school of science, Espoo, Finland, 2020.

Marten Oltrogge, Erik Derr, Christian Stransky, Yasemin Acar, Sascha Fahl, Christian Rossow, Giancarlo Pellegrino, Sven Bugiel, and Michael Backes. The rise of the citizen developer: Assessing the security impact of online app generators. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 634–647. IEEE, 2018.

Jeel Patel. Which method to use – low code vs traditional development?, December 16 2021. `https://www.monocubed.com/blog/low-code-vs-traditional-development/`. Accessed 31 August 2022.

Donald A Redelmeier and Daniel Kahneman. Patients' memories of painful medical treatments: Real-time and retrospective evaluations of two minimally invasive procedures. *pain*, 66(1):3–8, 1996.

Kerry Rodden, Hilary Hutchinson, and Xin Fu. Measuring the user experience on a large scale: user-centered metrics for web applications. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2395–2398, 2010.

Christian Rohrer. When to use which user-experience research methods. *Nielsen Norman Group*, 12, 2014.

Nayan B Ruparelia. Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3):8–13, 2010.

Richard M Ryan. Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of personality and social psychology*, 43(3):450, 1982.

Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178. IEEE, 2020.

Jill Bolte Taylor. *My stroke of insight*. Hachette UK, 2009.

Jan Tuomi et al. Adapting usability evaluation methods for the evaluation of developer experience. Master's thesis, Department of Computer Science, Aalto University school of science, Espoo, Finland, 2021.

Roel Vertegaal, Jeffrey S Shell, Daniel Chen, and Aadil Mamuji. Designing for augmented attention: Towards a framework for attentive user interfaces. *Computers in Human Behavior*, 22(4):771–789, 2006.

Robert Waszkowski. Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 52(10):376–381, 2019.

Marcus O Watson and Penelope M Sanderson. Designing for attention with sound: Challenges and extensions to ecological interface design. *Human factors*, 49(2):331–346, 2007.

Chauncey E Wilson. Triangulation: the explicit use of multiple methods, measures, and approaches for determining core issues in product development. *Interactions*, 13(6):46–ff, 2006.

Appendix A

Survey for Measuring DX of LCDPs

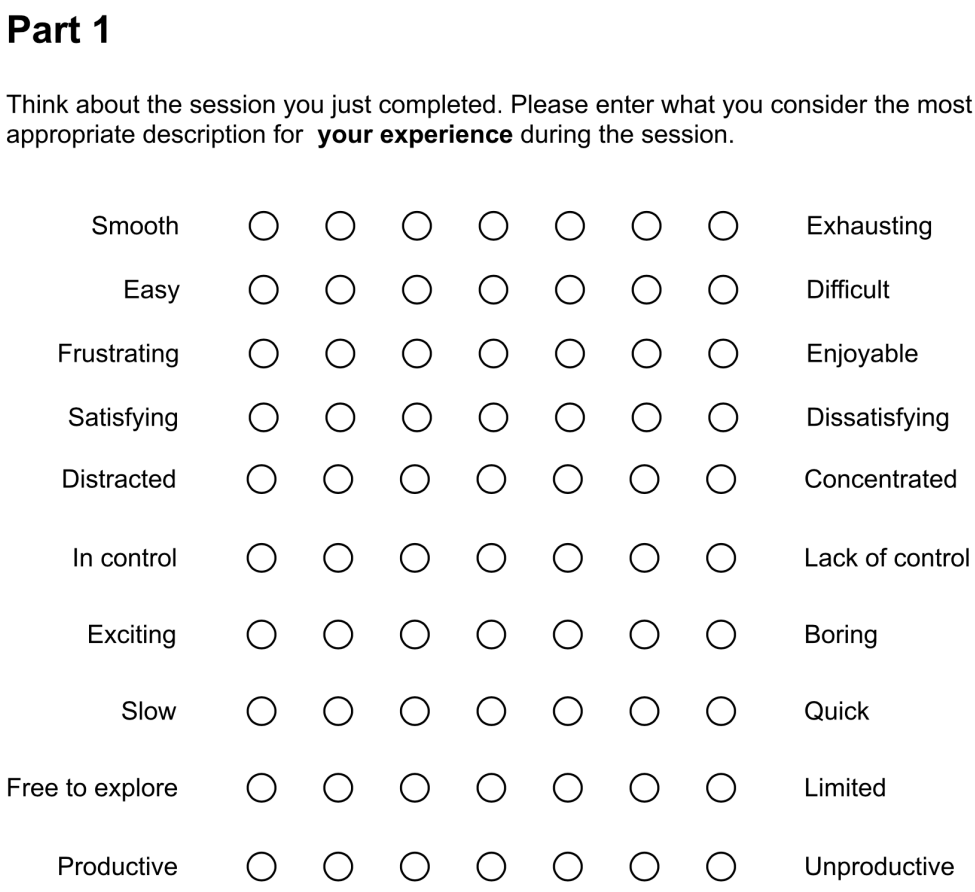

Figure A.1: Questionnaire Part 1.

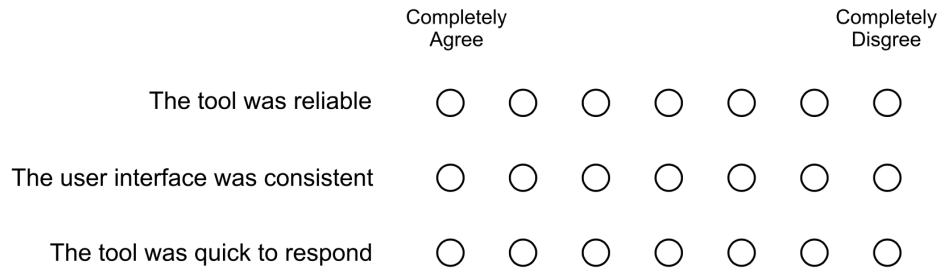

Figure A.2: Questionnaire Part 2.

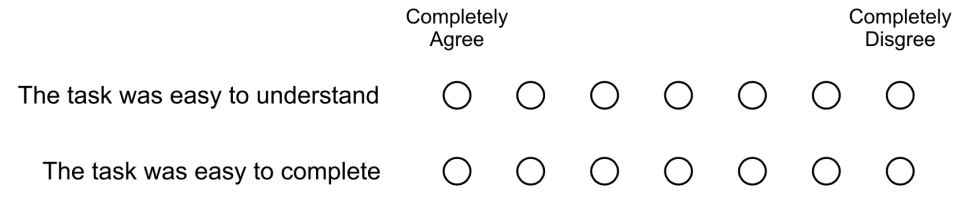

Figure A.3: Questionnaire Part 3.

Appendix B

Observational Task-based Test for Episodic DX

The test consists of two parts:

- **Guided Tour (25 mins)**: The researcher will guide you to get familiar with the AppGyver via implementing a simple Todo List application
- **Independent Tasks (35 mins)**: You will complete three tasks independently. However, you are welcome to give any comments during the test.

B.1 Guided Tour

The researcher will share her screen and show how to implement a simple Todo List application, like the wireframe in Figure B.1. After that, you can explore the AppGyver yourself at the remaining time.

B.2 Independent Tasks

B.2.1 Task 1: User Interface (5 mins)

Task requirements:

Add an **Input Field** and a **Button** component to the interface to allow users to add new asks, like Figure B.2.

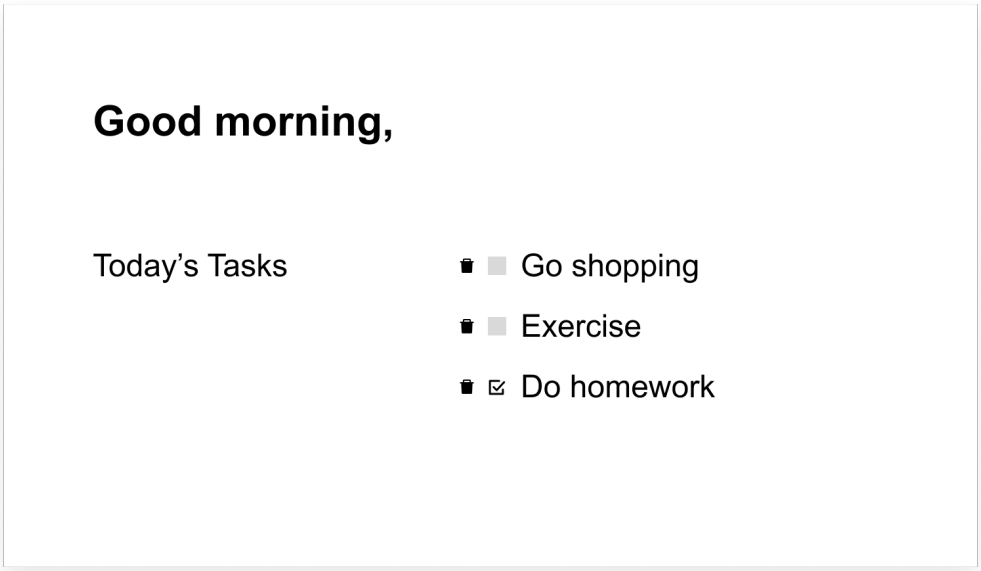

Figure B.1: Wireframe of a simple Todo List application.

B.2.2 Task 2: Add Logic — Delete Tasks (10 mins)

Task requirements:

Add logic on the **rubbish bin** icon. When you click on the icon, the corresponding task will be deleted.

Tips:

You might use the method **Delete Record** on the Logic editor.

After the task, please fill in the questionnaire.

B.2.3 Task 3: Add Logic — Add New Tasks (15 mins)

Task requirements:

Implement the feature: if you type a new task in the input field, such as "running", then click the button, the content "running" in the input will be added to the task list.

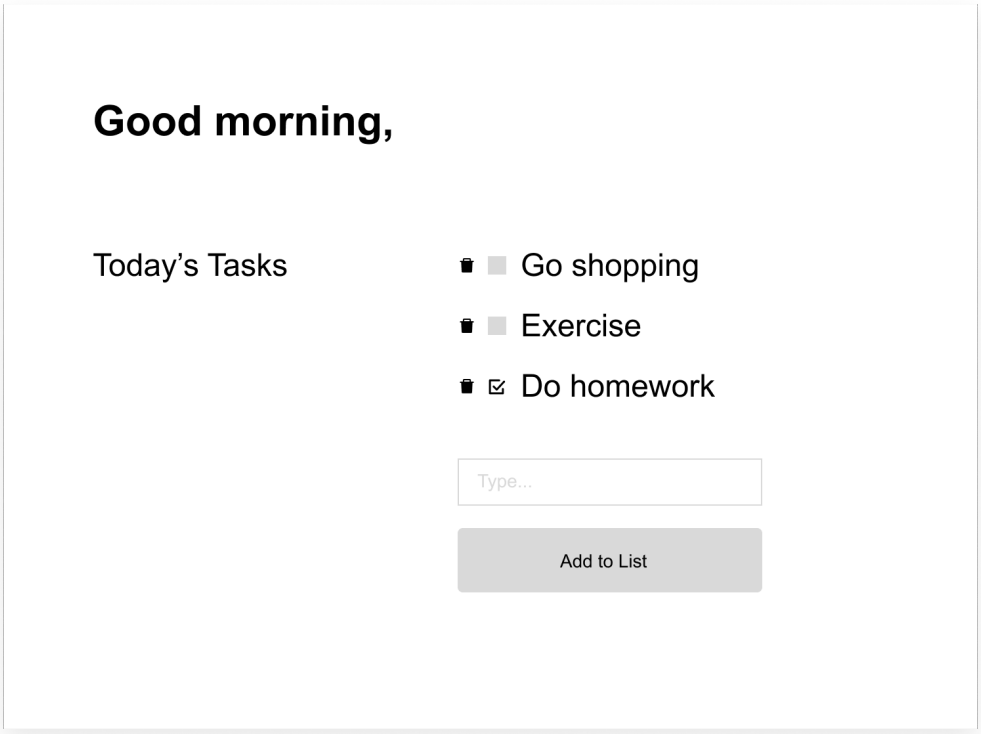

Figure B.2: Wireframe of task 1.

Tips:

You might need to create a **Data Variable** or **Page Variable** to store the new task.

You might use the method **Create Record** on the Logic editor.

After the task, please fill in the questionnaire.