KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering, specialising in ICT Innovation

Second cycle, 30 credits

Improving the user experience of touchscreen text-based code editor in an industrial robot controller

XUANLING XU

Stockholm, Sweden, 2023

Improving the user experience of touchscreen text-based code editor in an industrial robot controller

XUANLING XU

Master's Programme, ICT Innovation, 120 credits
Date: October 4, 2023

Supervisor: Adrian Benigno Latupeirissa
Examiner: Ylva Fernaeus
School of Electrical Engineering and Computer Science
Host company: ABB, Robotics & Discrete Automation

Swedish title: Förbättring av användarupplevelsen för textbaserad kodredigerare med pekskärm i en industriell robotkontroller

Abstract

This project investigated the touchscreen text-based code editor in OmniCore FlexPendant to improve its usability and user experience. This is a powerful but complex application used to program industrial robots. The objective is to redesign the user interface and interactions to make them more user-friendly and intuitive, with the goal of improving efficiency. The principles for designing complex applications and touchscreen products are generated as an outcome. From an academic standpoint, the research aims to fill the gap in text-based code editors for robot controller design and inspire touchscreen code editor design in other fields. Design thinking served as the framework for the design process, which encompassed seven steps that ranged from exploration to conceptualization and user testing. Guidance for improvement is ideated by 'become a user,' competitive analysis, and user studies. In the design phase, a high-fidelity prototype is built upon the original design with completely new interfaces, structures, and interactions. The user experience and usability are evaluated during user testing by counting task completion time, applying two standard user experience measurements, and conducting a brief interview. The results indicate that the new design achieved better completion efficiency in tasks, better user experience and usability scores, and received positive feedback from participants. The new solution meets the objectives and is considered a good reference for the design of industrial robot programming solutions.

Keywords

Industrial Robot Programming, Code Editor, Robot Controller, Teach Pendant, OmniCore FlexPendant, User Experience, Design Thinking

Sammanfattning

Denna studie undersökte den pekskärm- och textbaserade kodeditorn i OmniCore FlexPendant, för att förbättra dess användbarhet och användarupplevelse. Det är en kraftfull men komplex applikation som används för att programmera industrirobotar. Målet är en omarbetning av användargränssnittet och interaktionerna för att göra dem mer användarvänliga och intuitiva, med målet att förbättra effektiviteten. Principerna för att utforma komplexa applikationer och pekskärmsprodukter genereras som ett resultat. Ur ett akademiskt perspektiv syftar forskningen till att fylla luckan gällande design av textbaserade kodeditor för robotkontroller, och inspirera vid designen av pekskärmsbaserade kodeditorer inom andra fält. ”Design thinking” tjänade som ramverk för designprocessen, vilken omfattade sju steg som sträckte sig från utforskning till konceptualisering och användartestning. Vägledning för förbättringar tas fram genom ”att vara en användare”, konkurrensanalys och användarstudier. I designfasen byggs en högupplöst prototyp baserat på den ursprungliga designen med helt nya gränssnitt, struktur och interaktioner. Användarupplevelsen och användbarheten utvärderas under användartestning genom att räkna tid, tillämpa två standardmått för användarupplevelse och genomföra en kort intervju. Resultaten visar att den nya designen uppnådde högre effektivitet i uppgifter, bättre användarupplevelse och högre användbarhetspoäng samt fick positiv feedback från deltagarna. Den nya lösningen uppfyller målen och anses vara en bra referens för design av lösningar för programmering av industrirobotar.

Nyckelord

Industrirobotprogrammering, Kodeditor, Robotkontroller, Robotpendant, OmniCore FlexPendant, Användarupplevelse, Design Thinking

Acknowledgments

First of all, I would like to thank my supervisor at ABB, Björn Löfvendahl, for providing me with such an exciting topic, valuable insights and warm encouragement. I spent a very nice six months at ABB Robotics and I really appreciate the support of the Omega team, which provided me with a friendly environment. I would also like to thank the entire UX team at Robotics, especially Wiktor Bjellebeck, Meng Xu and Lisa Grennberg, who provided me with fresh ideas and insightful feedback in review sessions. To Adrian Benigno Latupeirissa, my supervisor at KTH, I am grateful for your help and encouragement for my project. To all the participants in the user study and testing, I really appreciate your trust and time, as your experience and feedback guide me in the right direction. I am also thankful for Ke, my boyfriend, for his excellent technical assistance in building the interactive prototype. Lastly, I would also like to thank Ylva Fernaeus, my examiner from KTH, Khiet Truong, critical observer from UTwente, and Julia Schwuchow, my opponent and nice friend, for their helpful suggestions.

Stockholm, October 2023
Xuanling Xu

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

KTH	KTH Royal Institute of Technology
SUS	System Usability Scale
UEQ	User Experience Questionnaire
UI	User Interface
UX	User Experience

Chapter 1

Introduction

The first chapter will present the project introduction, research question and limitation to confirm the project subject, objective and scope. The sustainability issues will be discussed to highlight the contribution of the deliverable to the environment.

1.1 Project Introduction

Nowadays, Industrial Robots are widely used in manufacturing industries to automate production and increase productivity. ABB, the world’s leading supplier of industrial robots and machine automation, offers various solutions for different industries. In their product portfolio, ABB offers a wide range of robots that are capable of performing specific tasks under operator control with various control and programming solutions. While ABB provides PC applications for programming and simulation purposes, the primary tool for controlling the robot on-site is the teach pendant. To be detailed, the ’teach pendant’ is the official name for the industrial robot control box, and the ’FlexPendant’ is the ABB robot controller’s brand name.

This project aims to improve the code editor module in ABB’s robot operator, OmniCore FlexPendant (see Figure 1.1). Before the OmniCore FlexPendant, the older model, IRC5 FlexPendant, has been the exemplar in the market for almost 20 years. Today, the latest version of ABB’s teach pendant, OmniCore FlexPendant, was launched in 2019, marking a significant milestone in the evolution of industrial robot controllers. With improved performance, OmniCore introduced an entirely new operating system and a larger multi-touch display, complete with the latest technology [2]. However,

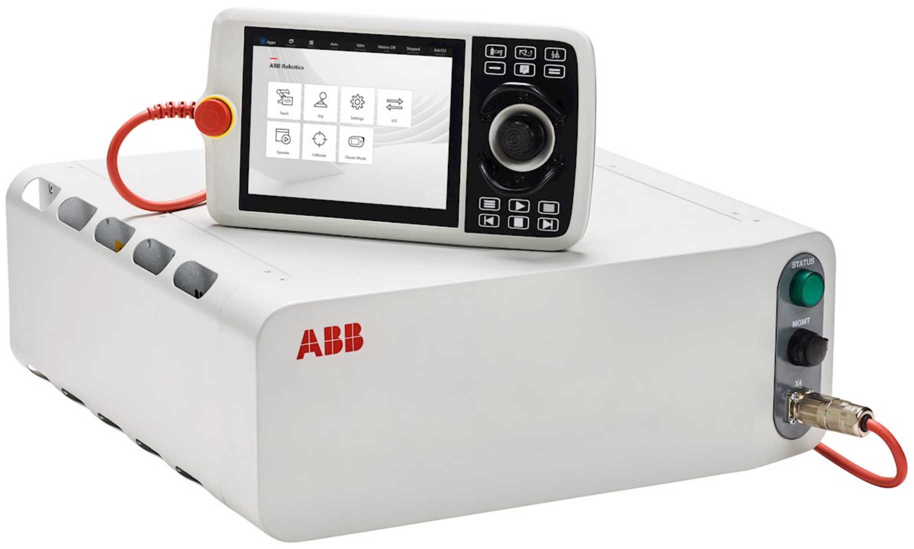


Figure 1.1: ABB OmniCore FlexPendant [1]

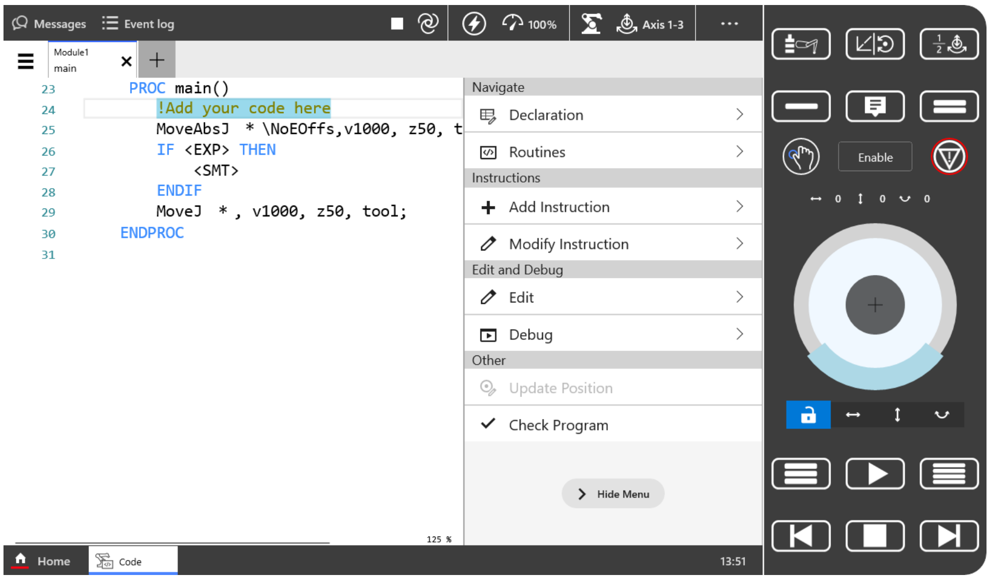
during the early stages of OmniCore's development, the user experience design was not given a prominent emphasis. Consequently, some user experience challenges have arisen while operators interact with OmniCore. Therefore, it is now pertinent to identify and address these user experience issues to enhance work efficiency and overall performance.

1.2 Research Question

The ultimate goal of this project is to improve the user experience of robot programmers using OmniCore FlexPendant for programming robot motion. The main research question of this project is "How can the interface of touchscreen text-based code editor in teach pendant be designed with regard to user experience and usability for robot programmers?" The investigation will focus on improving the user experience and usability of the scenario where robot programmers control the robot by using the text-based code editor in teach pendant. This code editor (see figure 1.2) is designed for touchscreens and is used to generate text-based code lines. The context of users' activities related to manufacturing, production, welding and lab experiments, etc., that use robots as tools to accomplish tasks in the workspace. The aim of this project is to improve the code editor's user experience and usability, so that robot programmers can use it to program the robot in a more efficient, intuitive and user-friendly way. The deliverables for this project include a novel code editor design solution and a set of design principles to navigate touchscreen code editor design. Additionally, conducting research to fill gaps in industrial robot programming solutions is one of the goals.

Objectives and Measurements:

- The new code editor will be more efficient and intuitive than the current one.
  - Robot programmers can spend less time coding tasks
  - Easier to find instruction, modify parameter; reduce redundant taps, steps and navigation
- The new code editor will be more user friendly than the current one
  - Obtain a higher score on User Experience Questionnaire and receive more positive feedback in user interviews
- The new code editor will be more adaptable to the touchscreen
  - Obtain a higher score on System Usability Scale



```
    PROC main()
        !Add your code here
        MoveAbsJ  * \NoEOffs,v1000, z50, t
        IF <EXP> THEN
            <SMT>
        ENDIF
        MoveJ  * , v1000, z50, tool;
    ENDPROC
```

Figure 1.2: A screenshot of the code editor module in virtual OmniCore FlexPendant

1.3 Limitation

There are three main limitations that could affect the improvement of the user experience, but they are not considered in this project. Firstly, some of the experience problems are caused by the hardware, such as the lag of the touchscreen and the size of the touchscreen. For instance, the touchscreen size of OmniCore FlexPendant is smaller than typical tablets (see Figure 1.3) which could be hampering design innovation to some degree. However, the focus of this study is on the improvement of the software experience. Even though the solution could be an attempt to reduce the influence of the hardware, it could not eliminate it. Nonetheless, it provides opportunities to explore small touchscreen interaction design. Secondly, the code editor module is not an independent application in the teach pendant. The programmer has to use other applications at the same time to complete the tasks, such as the I/O setting. Thus, the solution could not eliminate the effect of the design flaw of other applications. In addition, some of the design could not be changed because the entire software system was based on the Windows system. For example, the keyboard could only be the original one, and the problems caused by the design of the keyboard could only be solved by using other walkarounds.

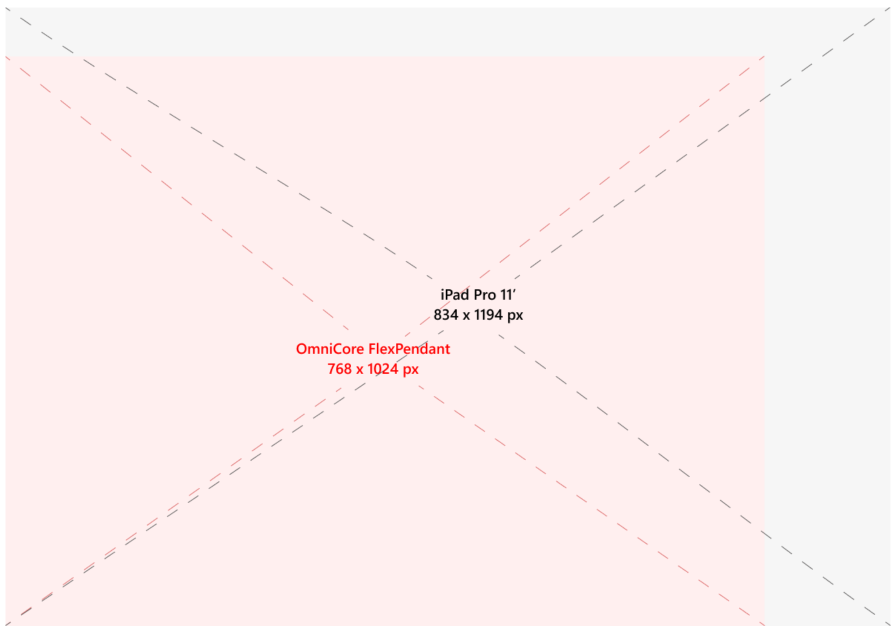


Figure 1.3: OmniCore FlexPendant touchscreen size compares to iPad pro 11's size

1.4 Sustainability

In terms of programmer's well-being, programming is an activity that requires the programmer to invest much cognitive effort and attention to comprehend tasks, program functions and solve syntax errors [3]. For industrial robot programmers, they have to concentrate on operating controller hardware, applications in the controller and robots for safety in their work. This requires multitasking and results in a higher mental workload in processing the information [4]. Regarding factors that hinder workflow, disruptions, time pressure and stress can lead to a high cognitive load that impacts programmers' well-being [5]. The current code editor's unsatisfactory usability and user experience results in a higher cognitive load and increased time pressure. According to previous research, the time pressure was confirmed as the main stressors in workplace [5]. With the increase of time pressure, both the ability to recall and the task accuracy will decrease [6][7]. Programming industrial robots is considered to be high difficulty and multitasking. When time pressure and task difficulty are both high, user performance and mental efficiency decrease, resulting in worse well-being [5]. However, if there is only one factor present, there is no obvious effect [5]. Therefore, by enhancing the code editor's user experience to improve work efficiency and reduce time pressure for complex tasks, individuals can experience lower perceived tension and better well-being. The expected outcomes aim to help programmers maintain good mental health and well-being, aligning with the themes about working conditions promoted by the International Labor Organization [8].

Chapter 2

Background

This chapter provides background information on the project. The contents are divided into two sections. First, the previous research and related work about industrial robot programming is presented to explain the type of programming solution and the current research trend. Then, the information about the programming solutions provided by ABB will be shown to give a better understanding of this project.

2.1 Industrial Robot Programming

Regarding the industrial robot programming method, from different perspectives and with development, the previous research proposed various categorising criteria. The most common way is to classify them into offline programming and online programming [9]. Offline programming allows operators to control and program virtual robots in a PC application, such as ABB's RobotStudio, without having to interact with physical robots in the manufacturing environment [10]. This type of application enables users to build and simulate work cells, test programs, and make adjustments before implementing them on real robots. However, it is necessary to use the teach pendant to fine-tune the robot's programs on site, as the simulation may not achieve the desired accuracy [9], which is referred to as online programming. With the increasing focus on user experience, the concept of end-user programming has been introduced and gained attention. This type of robotics programming allows the user to write a program for personal tasks rather than a generic function [11] and reduces the difficulty for non-professional users to program a robot with user-friendly interfaces [9].

Programming robots via ABB OmniCore FlexPendant is a combination of online programming and end-user programming. The lead-through function in the teach pendant, which is the main method of online programming, is used to control the robot to move to the desired positions and record them. Subsequently, with end-user programming, programmers can flexibly modify the code through application in the teach pendant, eliminating the need for typing code and complex actions.

The study of user-friendly end-user programming interfaces is the main focus of this research. There are two main types of end-user programming interfaces: manual and automatic. Automatic programming refers to the way of programming that has little control over the code, e.g., programming robots by voice or gesture control [12]. On the contrary, manual programming requires the user to directly control the programming instructions in the programming interface, whether generating codes by selecting functional snippets and entering parameters using the code editor on the teach pendant, or typing the code directly using an industrial programming language such as ABB's RAPID or KUKA's KRL [13].

When considering manual programming interfaces, there are two main types: text-based and graphical. In the text-based interface, the code lines are displayed directly to users directly on the code editor. Nowadays, with advancements in technology, programmers no longer need to manually write everything of code, but select instructions and adjust their parameters to generate code lines [12]. An example of such technology is the ABB OmniCore FlexPendant's code editor, which is the subject of this research. The technology leads to a significant reduction in programming time and learning cost. When it comes to the graphical programming interface, it hides the precise code lines and instead offers action blocks that users can drag and drop. These interfaces often present code's execution hierarchy in a flow-chart or diagram format [12]. This type of programming is often referred to as "no code" programming and is particularly user-friendly for novice users. For example, the ABB OmniCore code editor includes an application called Wizard, which facilitates rapid testing of a robot's execution of simple tasks.

To reach efficient information communication between humans and robots, the Graphical User Interface of robot controllers has attracted much attention in recent research [14]. However, for the code editor section, there is a gap in the research of touchscreen and text-based code editors. Relevant studies are more focused on prevalent touchscreens, such as mobile phones and tablets, while other research tends to explore new methods to program robots. For example, Weintrop et al. designed experiments to compare novice robot programming using CoBlox (block-based programming) with two widely-used methods (teach pendant and Universal Robot's tree-based programming) in terms of usability [13], learnability, and satisfaction. Foit also developed a hybrid programming operator to connect the computer to the robot, making it possible to program older machines easily and efficiently with an operator panel [15]. Therefore, the research on touchscreen text-based code editors is essential to fill the gap in some parts of the user experience of industrial robot programming and is also an important section in the field of the development of industrial robots.

2.2 ABB Programming Solution

ABB was one of the best companies in the world providing robotics and automation products and introduced their first robot in the year 1974 when it was known as ASEA [16]. Since then, they have continued to work on the latest and most innovative technology and have introduced a range of robots for manufacturing processes. The ABB's robots are programmed and operated by RAPID, a high-level programming language. It is a Structured Test (ST) programming language, similar to the Python or C language, and it contains the same type of statements, such as FOR, WHILE, IF. RAPID includes embedded functions, also called instructions in robotics programming [17]. For example, the instruction 'MoveL home, v600, fine, Tool0' means that the robot with 'Tool0' will move in a linear way at 600mm/s from the initial position to the 'home' position. The programmer can change the parameters: 'home', 'v600' and 'Tool0' to adjust the position, speed and range of the robot's movement. This type of programming language is easy to understand and easy to learn. In the 1990s, RobotStudio was released with support for offline programming and automation simulation, which was a revolution in the industry [16]. Today, RobotStudio, as the world's most famous offline programming and simulation tool, allows users to accurately build, test and refine the production process in the virtual environment, improving efficiency and productivity [18]. In addition to the desktop application, RobotStudio cloud, to be released in 2022, supports remote collaboration, allowing team members to work together in real-time from anywhere.

In the 1980s, ABB introduced the S series of hand-held robot controllers, including S2, S3, S, S4c and S4C plus, for convenient on-site control of the robots [19]. They are equipped with a display screen, a joystick and physical buttons. The S4C plus was introduced in the early 2000s, which is the final version of the S series and also the first to support ABB's RAPID programming [19]. In 2004, the IRC5 controller was launched, the name of which stands for Industrial Robot Controller. With its touchscreen, modular application and high degree of flexibility, it was the benchmark product of its time and is still widely used today. Its program editor allows the programmer to edit the RAPID program file via the touchscreen, providing satisfactory performance and ease of use. In 2019, the latest series, OmniCore, was launched, offering a larger multi-touch interface, various scalable functions and improved performance [19]. The controller is embedded with RobotWare 7, a powerful and intuitive operating system based on the Windows platform. Designed with the end user in mind, RobotWare 7 provides a simplified user experience that makes programming and operating robots easier, with a wide range of features and functions that enable robot programmers to operate and program robots in complex situations [2]. In addition to the general code editor, OmniCore also supports graphical programming with the wizard application, allowing novice users to quickly learn installation and control. In conclusion, OmniCore FlexPendant is a powerful tool in the real manufacturing workplace.

In summary, ABB offers leading solutions for machine automation with comprehensive robot programming products that meet the requirements in different contexts. RobotStudio allows users to program robots offline and simulate the production line in a virtual environment. IRC5 and OmniCore FlexPendant allow programmers to operate and program robots on-site. The aim of this thesis is to refine the text-based code editor in OmniCore FlexPendant to achieve higher productivity.

Chapter 3

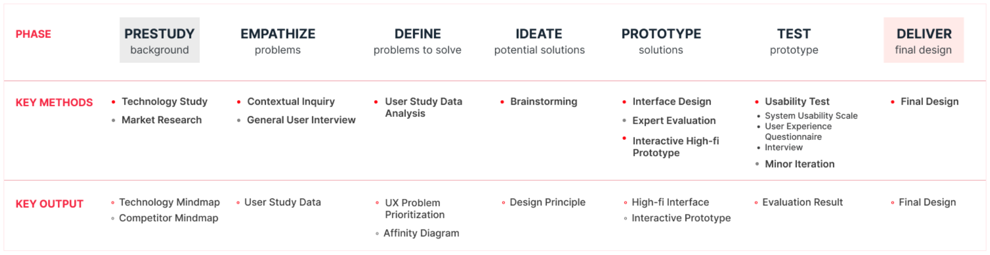
Methodology

This chapter presents the methodology related to the design framework and the methods used during the design execution. The content includes the rationale for the selection of the methods and their characteristics and procedures.

3.1 Design Thinking

Design Thinking is considered a paradigm for solving wicked problems [20], especially in the field of information technology [21]. The wicked problem corresponds to the complex application in this project, which provides broad, unstructured functions or non-linear workflows and is designed for a specialised domain with immense and technological skills [22]. In terms of this code editor, it requires users with industrial robotics education and receiving technical training. At the same time, it needs to collaborate with different applications in the teach pendant or on the desktop to carry out the tasks. The workflows and objectives are also dependent on the type of user and the tasks, which makes it a complex case. Therefore, the design thinking approach is chosen as the project framework to solve the challenge. This framework allows researchers to immerse themselves in the context to experience, create, prototype and test in order to iterate the solutions in a creative and analytical way [23].

According to d.school, the Design Thinking model consists of five phases: Empathize, Define, Ideate, Prototype, Test [24]. However, it is not a concrete and inflexible mode, but a non-linear and iterative process [24]. Therefore, this project framework (see Figure 3.1) was based on this model and added a pre-study to help familiarize with basic industrial programming techniques and market trends and added review sessions in the prototype phase. Following this, the Empathize phase aims to observe people and problems in the context of the challenge. Therefore, interviews and contextual inquiry are chosen as methods to reveal users' requirements and expectations. Then, the aim of the Define phase is to clarify the problems in order to direct the design. Therefore, Thematic Analysis and MoSCoW Analysis are used to manage users' feedback and problems identified in the Empathize phase. Before starting the prototyping work, some design principles are generated to guide the solution design. When moving to the Prototype stage, solutions are generated and then iterated by expert review. When the prototype is ready, in the testing phase, users are invited to interact with the prototypes to evaluate their usability and user experience for further iteration. The final solution is then proposed.



PHASE	PRESTUDY background	EMPATHIZE problems	DEFINE problems to solve	IDEATE potential solutions	PROTOTYPE solutions	TEST prototype	DELIVER final design
KEY METHODS	• Technology Study • Market Research	• Contextual Inquiry • General User Interview	• User Study Data Analysis	• Brainstorming	• Interface Design • Expert Evaluation • Interactive High-fi Prototype	• Usability Test • System Usability Scale • User Experience Questionnaire • Interview • Minor Iteration	• Final Design
KEY OUTPUT	• Technology Mindmap • Competitor Mindmap	• User Study Data	• UX Problem Prioritization • Affinity Diagram	• Design Principle	• High-fi Interface • Interactive Prototype	• Evaluation Result	• Final Design

Figure 3.1: Design Framework based on Design Thinking

3.2 Design Process

The rationale and benefits of the methods used in each process were presented in these sections. The execution process and result will be described in the chapter 4.

3.2.1 Pre-Study

In the pre-study section, in order to put oneself in the users' shoes and understand their experience, the first thing is to 'become a user.' This involves learning how to use RobotStudio and the OmniCore FlexPendant to program the industrial robots. It is an informal research activity and aims to gain background knowledge of this project and get a preliminary understanding of the user workflow. The competitive analysis is then chosen to gain insight into the features, structure and interaction logic of the competitors [25]. By understanding their drawbacks and advantages, designers can build their unique products while avoiding the same design flaws and same scope. It focuses on identifying factors affecting user experience and gaps in the market to refine our design [26]. The analysis starts with the main competitors, but covers more products related to text-based code editors [25]. The research concludes with a map that includes a description of each product, its strengths and weaknesses, a summary and the corresponding interface.

3.2.2 Empathize

Interview and contextual inquiry are the methods used to obtain detailed user experience feedback. Firstly, a semi-structured interview is used to guide the conversation between researchers and programmers, starting from a defined set of questions and continuing with derived questions [27]. The interviews are conducted online to reach users around the world. The aim of the interview is to empathize with users' experiences and to gather their expectations for feature improvements. However, online interviews can only collect the data that users recall, and users cannot use the real code editor to show their issues; thus the memory limitation may conceal some defects [27]. Therefore, contextual inquiry introduced by Beyer and Holtzblatt is the method to complement the interview [28]. It is a field study method that includes four main settings: context, relationship, interpretation and focus [27]. The activity should be carried out in the environment in which they perform their daily work; the relationship between researcher and participant should be a master-apprentice relationship where participants teach the researcher about their activities like teaching an apprentice; the researcher should confirm their observation and understanding of each action with the participant; finally, the research activities should focus on the subject of this project [27]. After this study, the user's real-time reaction, behaviour and feedback will be collected to make the emphasize result more reliable.

3.2.3 Define & Ideate

In the define phase, thematic analysis and MoSCoW analysis are used to extract insights. In order to manage the interview data, the first step is to transcribe the recordings into text. The edited transcript is created by deleting word crutches or misstatements [27]. The researcher goes through the transcript, highlights key passages and gives them a summary of the content [29], and then the content corresponding to the same label is collated as a code card. When all the code cards are prepared, the codes that have a similar meaning or indicate the same feedback are gathered together and a theme is created to summarise the cards in the same group. Once a set of themes has been obtained, the researcher can begin to review the themes and categorise them into new themes with a higher hierarchy. The final result is presented in the form of an affinity diagram and a summary of the findings.

The thematic analysis uncovers users' attitudes towards the overall user experience, while the details of the User Experience (UX) challenges revealed in the interview and contextual inquiry require MoSCoW analysis to organise them. The MoSCoW method is a way of prioritising requirements in application development [30]. As this code editor is a complex application and users have different preferences and suggestions for its improvement, MoSCoW prioritisation can help to identify the importance of requirements to determine the design scope. The metric consists of four prioritization categories: Must Have, Should Have, Could Have and Won't Have [31]. To adapt to this project, the four categories are revised as 'Must Improve,' 'Should Improve,' 'Could Improve' and 'Won't Improve.' To aid the analysis, a UX Issues card is designed to describe each challenge, and then each card is categorised into four areas based on the criteria. Must Improve' indicates the essential issues that need to be resolved in a short time; 'Should Improve' indicates the things that can improve the user experience a lot but aren't necessary right now; 'Can Improve' issues are those that are good to make some changes if the resources are sufficient; 'Won't Improve' issues are those that participants want but don't fit the context. The result is represented by a MoSCoW metric.

3.2.4 Prototype

In the prototyping phase, low-fidelity and high-fidelity prototypes are created using Figma, and expert review helps to evaluate them for initial iteration. The method used in low-fidelity prototyping is to draw wireframes upon the screenshot of the current design to cover the current design elements. The wireframes allow the designers to view the layout of the solutions quickly and focus on the solutions and features that they have come up with [32]. Once the layout, the content and the information architecture have been confirmed, the next step is to create high-fidelity prototypes. The high-fidelity prototypes look like a 'live' editor with well-designed User Interface (UI) components and realistic content [33]. To get reliable user testing results, interactive prototypes are developed with external help.

When prototyping the solution, UX expert review sessions are conducted to help perfect the design. This is a usability inspection method to evaluate the design and identify the issues to be improved [34]. Expert in this section refers to the senior UX designer colleagues at ABB who have solid knowledge and practice in UX design. The aim of UX expert review is to step out of the box to evaluate the solution, identify the things that violate the user experience and gain new perspectives for improvement [34]. The researcher prepared a description of the problem and the solution with interfaces. The participant first identifies the strengths and weaknesses of the solutions, and then rates the severity of the violation, such as high, medium or low [34]. Finally, he gives suggestions or shows examples of good practice to guide how to solve the problem. The result will contribute to the refinement of the high-fidelity prototypes.

3.2.5 Test

In user testing, a within-subjects experiment is designed to evaluate whether the solution meets subjective user experience and usability. This type of experiment requires participants to complete tasks in more than one context to examine how the individual experience changes when the test subject changes to obtain the result [35]. The within-subject method only needs a small sample of participants and does not need to consider the variations between individuals as all participants are in the same conditions [36]. Therefore, in user testing, all participants use two code editors to complete the same coding tasks. There are four evaluation metrics: time taken, User Experience Questionnaire (UEQ), System Usability Scale (SUS) and interview feedback. The time each user takes to complete each task is recorded to analyze efficiency improvement. The Standard User Experience Questionnaire is a survey containing 26 items belonging to six factors to quickly assess the overall user experience of the products [37]. However, a standard UEQ takes a lot of time to complete [38]. Therefore, a short version with 10 items covering four factors is used: Attractiveness, Perspicuity, Efficiency and Dependability, which are aligned with the project objective. Then, the System Usability Scale developed by John Brooke with ten 5-point Likert questions is used to measure usability in industrial system evaluations in a quick and dirty way [39]. The following interview is a short discussion about the user experience and suggestions for improvement to support the iteration design. The iteration ideation helps to polish the final deliverable.

Chapter 4

Execution and Analysis

This chapter presents the execution process and analysis result based on the Design Thinking framework. The content consists of six parts: Pre-Study, Empathize, Define&Ideate, Prototype, Test and Final Deliver. These phases show the procedure from learning the new challenge and identifying the problems to finding the solutions. The solution design was iterated twice through design review and user testing to achieve the objectives. The code editor of the OmniCore FlexPendant was redesigned to improve the user experience and usability for robot programmers.

4.1 Pre-Study

Prior to the collection of user feedback, it was necessary to master the research subject of this project and to analyze the related products in ABB's portfolio and other solutions on the market. Firstly, the 'become a use' activity was not only to learn about the current solutions, but also a way to put the researcher in the user's shoes. Then, through research into the other programming solutions in ABB's product portfolio, it helped to build a comprehensive understanding of each product's target and its design style. The researcher then identified the design focus of this research and ensured that the solution matched its users' and scenarios' requirements. In addition, the market and competitor analysis was to have an understanding of the trends and existing solutions in the market. An in-depth analysis of their interface design was carried out to identify their strengths and weaknesses, which served as lessons learned for future design. This section presents the process and outcome of the preliminary study.

4.1.1 Become a user

’Become a user’ involved learning how to use OmniCore FlexPendant, RobotStudio and IRC5 FlexPendant through official tutorials and YouTube videos. OmniCore FlexPendant is the subject of this study. The main objective was to learn how to use the real OmniCore FlexPendant to configure, operate and program a robot to perform a simple task such as picking and dropping objects. It was vital to go through the whole process to understand how programmers interact with robots and teach pendant. Although RobotStudio and IRC5 FlexPendant were not included in this research, they had a strong connection with OmniCore FlexPendant in the real workplace. In RobotStudio, users can run the virtual FlexPendant to stimulate the process of using a physical controller to control robots in the work cell. Regarding the IRC5 FlexPendant, it was still the prominent controller in the real workplace. Therefore, it was worthwhile to gain insight into its interface and interaction design through study. Through this learning, the author gained enough knowledge to program a simple task for industrial robots.

4.1.2 ABB Portfolio Research

ABB offers a wide range of industrial robot programming solutions, each tailored to different users and scenarios. The purpose of this section is to gain a comprehensive understanding of the product strategies and ecosystem. The analysis of each product helped the researcher to distinguish the target users and scenarios of the research subject. Three industrial robot program editors are presented in the following section.

RobotStudio RobotStudio is the most important programming and stimulation application in ABB Robotics that launched in 1998 (see Figure 4.1). It has been continuously upgraded in terms of features and performance. RobotStudio has powerful functionalities and supports advanced programming. For example, it can stimulate the real working environment with models and test their feasibility before implementation in production. Its target users are professional users, such as technical support and robotics engineers who have knowledge of RAPID programming and robotics. The application can only be run on the PC. To program a robot, users typically enter code directly into the RAPID editor.

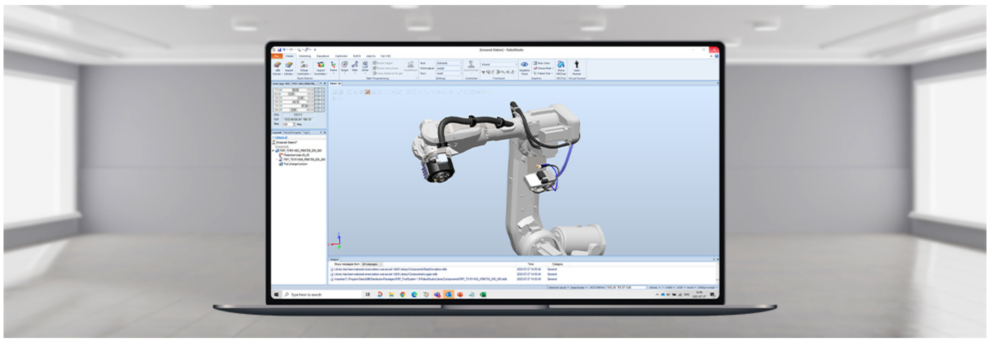


Figure 4.1: RobotStudio [40]

RobotStudio Cloud RobotStudio Cloud is the website version of RobotStudio and was introduced in 2022 (see Figure 4.2). Compared to RobotStudio, it does not support building stimulation cells, but provides strong real-time collaboration and cloud storage functions. RobotStudio Cloud aims to increase digitalization through smooth collaboration across remote teams. Its editor supports users to modify basic data on the property panel and enter code directly. It still requires users to have knowledge of industrial robot programming.

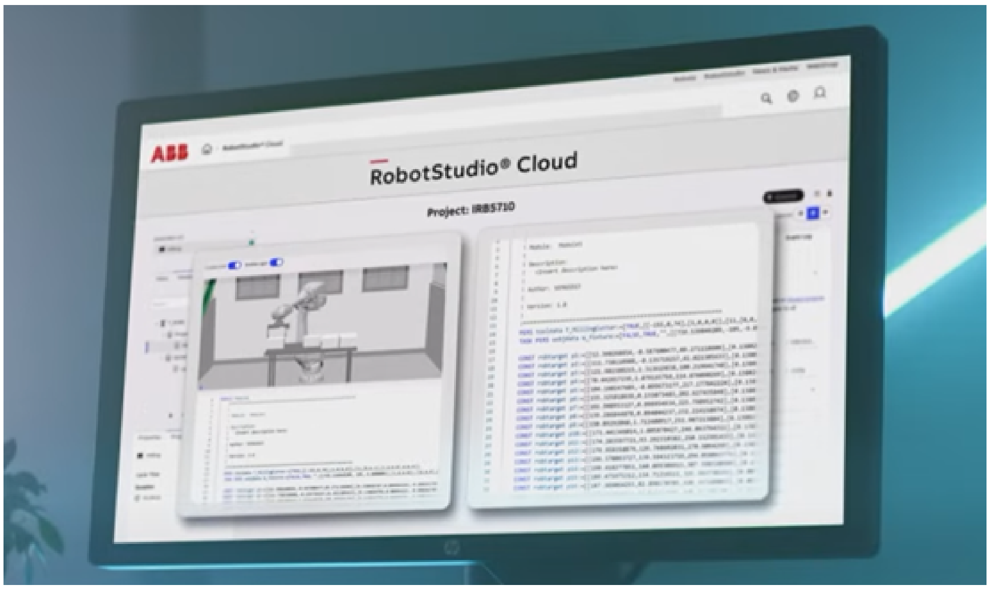


Figure 4.2: A screenshot from ABB tutorial [41] showing the interface of RobotStudio Cloud

Wizard Easy Programming Software Wizard is a graphical programming application in the physical or virtual OmniCore FlexPendant (see Figure 4.3). The target user of this application is the novice user. The user does not need to have any programming knowledge. They can drag and drop the instruction block in the editor and address the parameter to program the robot. The code block expression is like natural language that is easy to understand, for example, 'move tool0 quickly to location'.

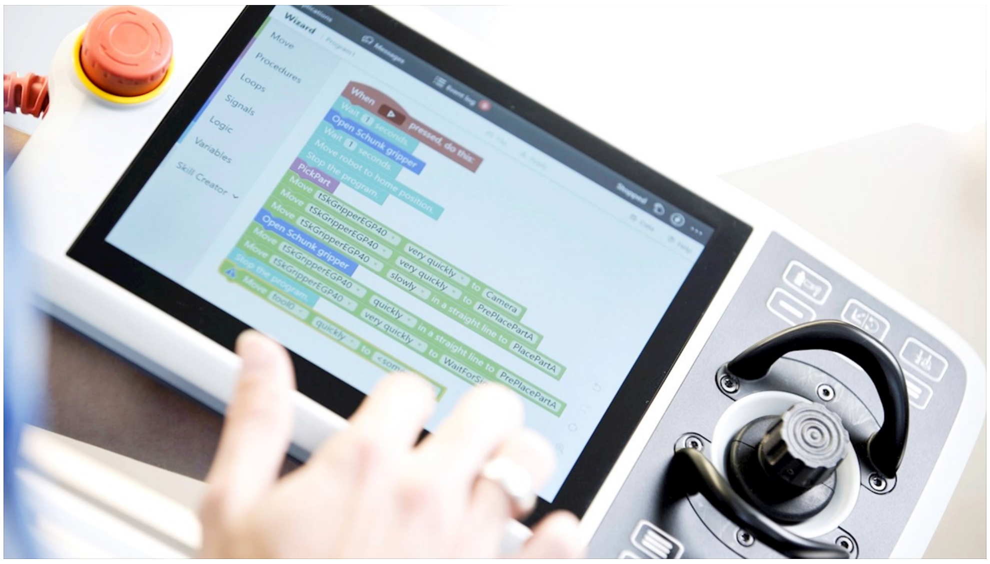


Figure 4.3: Wizard Easy Programming Software [42]

The products mentioned above cover a wide range of users and scenarios, while OmniCore FlexPendant has a different focus. In the real workplace, such as a factory, professional programmers cannot take their laptops with them. It is also impossible to fully simulate the details of the real environment in software. In such cases, OmniCore FlexPendant is a professional tool for programmers to control industrial robots in the field.

4.1.3 Competitive Analysis

Researching existing market offerings is essential to creating a product that meets the needs of the market and users. The process involved first identifying other touchscreen programming solutions on the market. Then systematically categorise and analyse the existing solutions. The main research method is to read the user manual or watch the tutorial video and analyse the interface and interaction design of each solution by identifying its advantages and disadvantages. Findings and insights were then produced to support future designs.

In this research process, nine touchscreen code editors from seven different companies were meticulously researched (see Figure 4.4). Three categories were proposed to classify all the solutions, including flow programming, tree-based programming and text-based programming. For each interface design, an in-depth analysis was carried out, listing the layout, strengths and weaknesses of the interaction and interface design, and a summary of the evaluation of the solution. The result of the analysis is shown below.

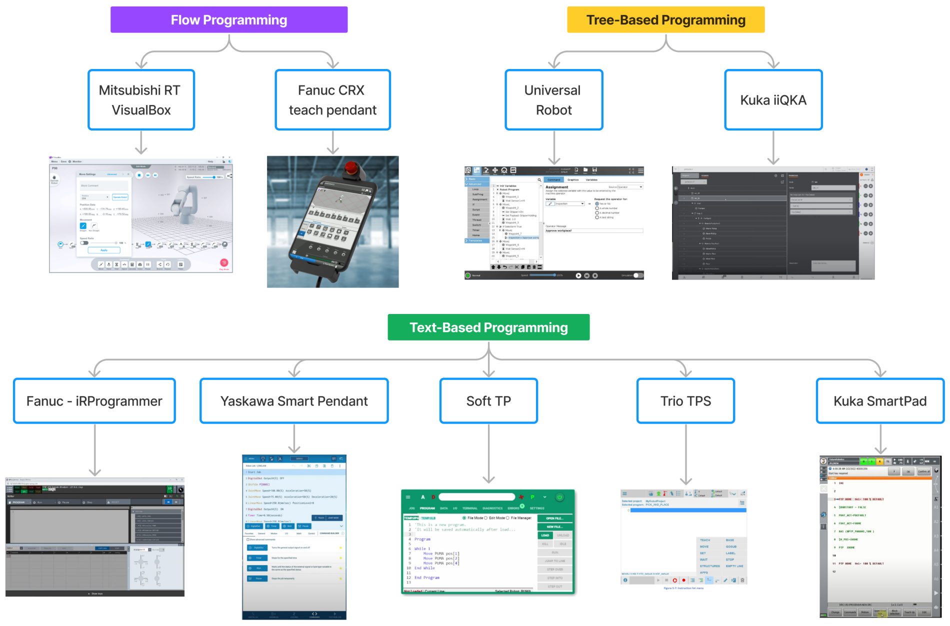


Figure 4.4: Products involved in competitive analysis

Flow Programming Interface This is a graphical programming solution that allows users to drag and drop the instruction block onto a timeline and configure the instruction in the property panel to program the robot. For novice users, this approach is more accessible. Throughout the process, the user does not see the code, but rather an instruction box with an icon and a textual description. This method is similar to ABB's Wizard programming, with the difference that arguments and parameters are not nested within the parent block. This category includes Mitsubishi's RT VisualBox and Fanuc's CRX teach pendant. Both solutions offer clear and attractive interfaces and user-friendly and intuitive interaction. Their property and operation panels are compact, structured and easy to understand. Its programming experience is streamlined, which greatly reduces the learning curve. However, these two editors have certain limitations that make them unsuitable for advanced programming. In terms of RT Visual Box, the layout of the interface limited the number of instruction blocks that make the program unable to craft complex programs. Similarly, in CRX Teach Pendant, when a new motion instruction is added, it automatically generates a new position data, naming it with a number. However, it is a challenge for users to remember which number represents which position data in a complex programme. Although the interface design is not suitable for professional programming, the design of its property panels can provide some inspiration.

Tree-based Programming Interface Tree-based programming goes beyond graphical programming: although it does not display the exact code, it shows the clear hierarchy of the program structure and allows the robot's path to be defined directly. This method is suitable for both novice and professional users, as the program structure is clear and easy to edit, and the properties panel provides both basic and advanced settings. Examples in this category include Universal Robots' 3PE Teach Pendant and Kuka's iiQKA. Both products have a horizontal layout with a tree-based instruction structure on the left and the property panel on the right, with a clean and consistent design style. They have many advantages that are worth learning about. For example, in the 3PE Teach pendant, its keyboard includes several preset keywords that increase the efficiency of finding frequently used symbols. The iiQKA's system displays different statuses or highlights critical information using different colours. Last but not least, its programming structure includes a function for folding and unfolding nodes, which greatly improves the clarity of the structure.

Text-based Programming Interface The text-based programming interface is the research focus, which displays code in a similar way to conventional PC code editors, although the interactions are somewhat different. Programmers are required to select commands from the list and edit configurations in the properties panel to generate code. This approach allows precise scripting and configuration to support advanced programming. As this is the main subject of this research, the author looked at all the touchscreen text-based code editors on the market, including Fanuc's iRProgrammer, Yaskawa's Smart Pendant, Kuka's SmartPad, Trio's TPS and Servotronix's Soft TP. Yaskawa's Smart Pendant achieved the best performance of the five solutions. It has a vertical layout with two responsive windows and thoughtfully places the configuration window in a finger-friendly zone. In addition, the programming panel is responsive and remains visible when the keyboard is engaged, a considerate design choice for uninterrupted workflow. Compared to the other four solutions, it stands out for its user-friendly and elegant interface design. When it comes to the other four solutions, they all have some drawbacks that pose user experience challenges. The lessons learned were summarised as follows:

- Interaction flow: The interaction flow should enable intuitive working processes and streamline the workflow.
- Property panel size: If the argument configuration area is too small, it can be a challenge to display multiple arguments and include shortcuts.
- Icon clarity: Icons should be clear and easy to understand. Whenever possible, include a text description below the icon for added clarity.
- Limit pop-up windows: Avoid excessive use of pop-up windows and reduce the amount of information they contain to avoid clutter. Pop-up windows may not be the best choice as containers for instruction configuration settings.
- Abbreviations: Minimize the use of abbreviations unless they are widely accepted and understood.
- Button icons: Avoid using single letters as button icons as they can be ambiguous.
- Button descriptions: Ensure that the button descriptions are clear and unambiguous to avoid any confusion regarding their functionality.
- Reduce Redundancy: Streamline interactions by eliminating redundant steps to improve efficiency.

4.1.3.1 Competitive Analysis Summary

This section shows that there are several excellent examples of flow and tree-based programming on the market. They provide a commendable user experience. Graphical programming interfaces are the current trend in the market, as these examples are introduced in recent years. However, when it comes to text-based programming solutions, most products on the market also face challenges in user experience and usability. Even Yaskawa's Smart Pendant, which is notable for its user-friendly design, uses a vertical interface layout, while other products in the same category use a horizontal layout, as does the OmniCore FlexPendant. As a result, the market lacks an excellent text-based programming interface for touch screens. This finding indicates an opportunity for ABB to enhance the OmniCore FlexPendant to fill the gap and capture a large market share. The results of this section, especially the lessons learned from the interface analysis, contributed to the redesign of the OmniCore FlexPendant code editor.

4.2 Empathize

In the Empathize phase, a deeper understanding of user requirements and UX challenges is essential, and two research activities are leveraged to collect data. In order to reach robot programmers around the world, online interviews were conducted to gather first-hand feedback on user experience and preferences. Field studies were then carried out to observe the interaction between humans, robots and teach pendants in the workplace. The process of the activities was described as follows.

4.2.1 Interview

The semi-structured interviews were conducted with thirteen participants. All participants are experienced robot programmers and participated in the interview online. The participants described their user experience and the challenges they faced, and shared the screen to show the workflow through the virtual FlexPendant in RobotStudio. The whole interview would take about 60 minutes. The detailed content and process is presented in the following sections.

4.2.1.1 Participant

Thirteen robot programmers with knowledge of ABB robot programming and experience in using the OmniCore FlexPendant participated in the interview. The participant's contact information was obtained from the project provider or from the ABB Robotics Forum. These thirteen participants came from seven countries, including the United States, Sweden, Germany, China, India, France, and Canada; all of them were men. They had professional experience in this field. Among them, five individuals have over 15 years of experience using ABB's products, eight have experience ranging from 5 to 15 years, and the remaining two have less than 5 years of experience. In addition, the majority of the interviewees were ABB employees, while two were external customers, as the OmniCore FlexPendant was not yet widely used in the market.

4.2.1.2 Interview Material

The interview question sheet is in a semi-structurde manner (see Appendix A). It has four sections. First, in the warm-up section, the researcher introduced the process and guided the participants to talk about their backgrounds, roles, responsibilities and experiences in the field. In the main part of the interview, there were two main topics: FlexPendant related and code editor related. In the first topic, the participants were asked about their experience and workflow in using FlexPendant in their daily work. The next topic focused on the code editor application in FlexPendant and the discussion delved into interface details, including menu design, error messages, keyboard and property panel design. It should be noted that although OmniCore FlexPendant is the subject of the research, the older version, IRC5 FlexPendant, was also mentioned as it was well accepted by the market. Finally, in the warm-up section, the research provided more open-ended questions to encourage participants to talk about their expectations and ideas for improvement.

4.2.1.3 Procedure

Prior to the interview, the interview invitation emails were sent to confirm the schedule with the participant. Once the participant agreed to participate in the study, an online meeting was scheduled via Microsoft Teams. At the start of the interview, the researcher began by introducing himself/herself, providing an overview of the project, explaining the interview process, assuring the participant of the confidentiality of the data collected, and asking for permission to record the interview. The interview was then recorded using Microsoft Teams. Throughout the interview, the researcher kindly listens to their answers to the questions and, based on their feedback, pops up follow-up questions for deeper discussion. When discussing the details of the interface design, in order to get a nuanced understanding, the researcher or participants would share the screen to present the interaction details using the virtual FlexPendant in RobotStudio. At the end of the interview, the researcher summarised the key points and thanked the interviewees for their participation.

4.2.2 Contextual Inquiry

The Contextual Inquiry, a method of field study, was carried out at the ABB Robotics Lab on-site with three participants. This activity gave the researcher the opportunity to go through the workflow with the robotic engineers using the physical OmniCore FlexPendant. The aim is to go beyond the recall of the workflow, but to experience it and fill in the gaps in the memory in the actual working environment.

4.2.2.1 Participant

Three robot programmers located in Västerås and working in the laboratory participated. The participant was connected via the project provider. All three participants were male ABB employees and were familiar with the operation of the OmniCore FlexPendant. One of them has 25 year of experience in Robotics, and the other two within 5 years.

4.2.2.2 Procedure

The procedure consists of five sections: preparation, primer, transition, contextual interview, and wrap-up [43]; see the appendix B for more details. In preparation, activity schedules were arranged through Microsoft Teams and a GoPro camera was prepared to record the interaction. The researcher and participants met in a work cell in the ABB Robotics laboratory. Then the study started with an introductory phase, where the researcher introduced the project and the research procedure to establish rapport with the participants, clarified issues related to confidentiality and obtained consent to record behaviour and discussion. A GoPro camera was then installed above the OmniCore FlexPendant to record the participants' interaction with the device (see Figure 4.5). Participants were instructed to treat the researcher as a student and go through the workflow to mimic a teacher teaching a student a skill by doing. The researcher informs the participants that their interaction with the robots and the OmniCore FlexPendant will be observed and that they should expect interruptions for discussion or questions about their behaviour. Participants were encouraged to correct any misinterpretations of the researcher's speculations. Once the participants had gained sufficient understanding of this field study, they began to perform one or two tasks to demonstrate their daily interaction with the robots and the OmniCore FlexPendant. During this process, they were questioned about the purpose of certain behaviours, the reasoning behind certain decisions, the logic behind problems encountered, and expectations for future improvements. The study concluded with a debriefing session in which the researcher provided a summary of the observations and the participants were asked to review the interpretations to ensure the accuracy of the feedback.

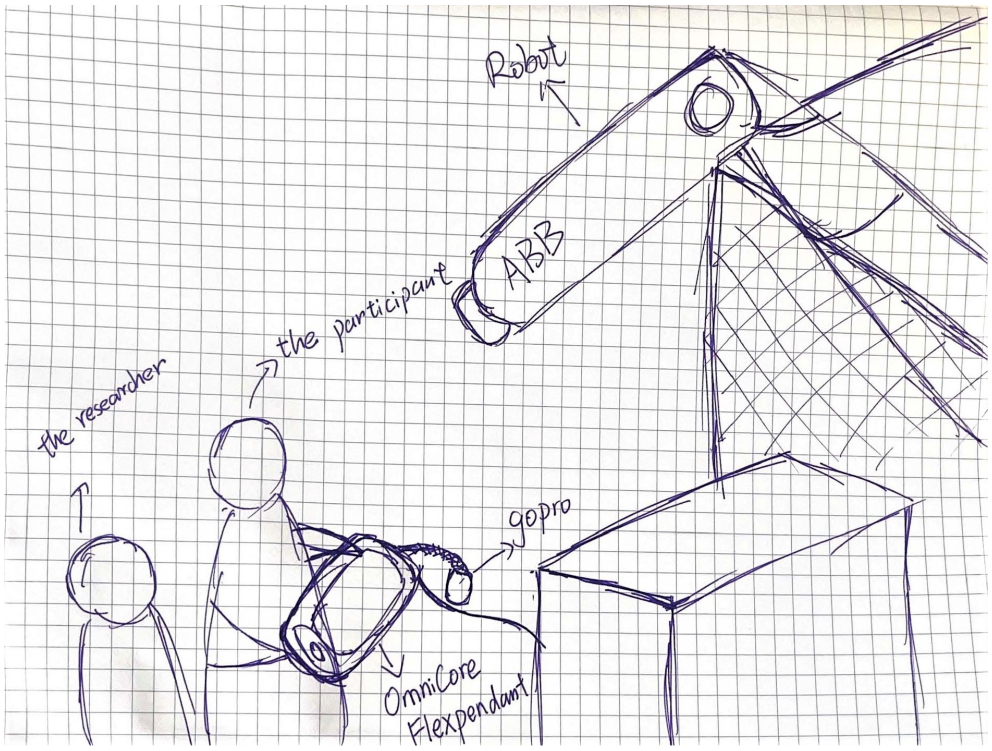


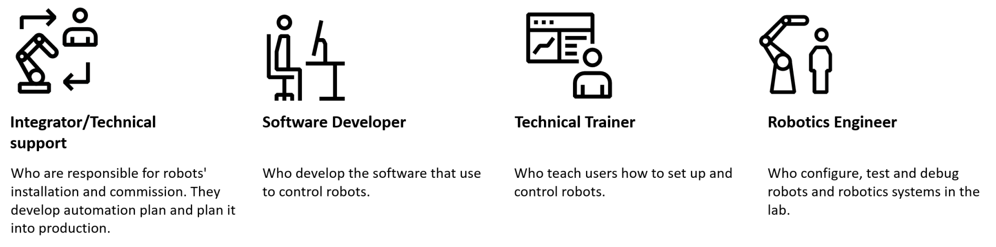
Figure 4.5: Contextual Inquiry environment setting

4.3 Define & Ideate

In the Define phase, four target audiences were identified through user research data, and two methods, namely thematic analysis and MoSCoW analysis, were used for analysis. The transcript of the interview was analyzed using the thematic analysis method, which resulted in the creation of an affinity map to comprehend the users' overall feedback. The MoSCoW analysis facilitated the categorization and prioritization of all user experience issues. The outcome of the Define phase led to the creation of design principles, which would serve as a reference for future designs.

4.3.1 Target Audience

The Empathize phase involved a total of 16 participants, 13 of whom were interviewed online and 3 of whom participated in the field study. The participants came from 7 different countries and represented four key audiences (see Figure 4.6). The first type includes integrators and technical support personnel who deploy and commission robots, ensuring proper configuration and calibration before integrating them into the production line. Their concern is whether the functionality in FlexPendant can support all types of programming tasks, especially the initial setup and debugging process. Second, software developers are responsible for developing software for robot controllers. They are interested in optimizing the software functionalities for better performance and ease of use. Then, technical trainers provide training to students and technical users. They expect the user interfaces to be well organized to locate functions and reduce the learning curve for novice users. Finally, robotic engineers who test robots and robotic systems in the lab and participate in the field study, also provided a unique perspective. Teach pendants are the most commonly used devices in their work, leading them to expect the system to provide better guidance and run without problems. The variety of perspectives provided a comprehensive view of user experiences and the unique challenges faced by different users in using OmniCore FlexPendant. In addition, the composition of the participants ensured data reliability.



Integrator/Technical support

Who are responsible for robots' installation and commission. They develop automation plan and plan it into production.

Software Developer

Who develop the software that use to control robots.

Technical Trainer

Who teach users how to set up and control robots.

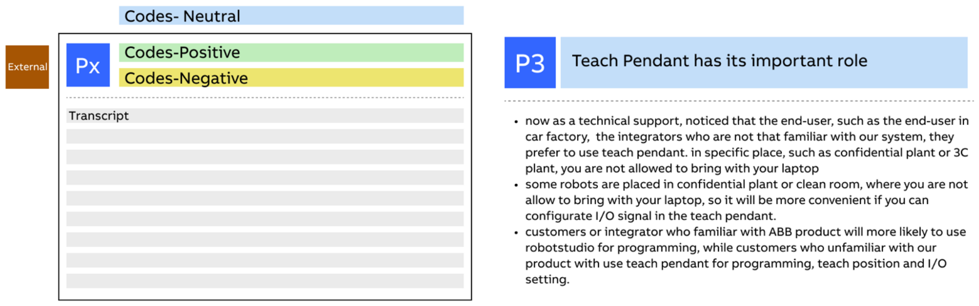
Robotics Engineer

Who configure, test and debug robots and robotics systems in the lab.

Figure 4.6: Four type of target audience

4.3.2 Thematic Analysis

Thematic analysis was the method of choice for the analysis of the interview data. The recording was first transcribed by Microsoft Teams and organized as a Summarized Transcript by picking up the important content. The analysis focused on the assessment of user experience and usability. Due to the large size of the data set, a code card was created to visualize the transcripts. The card listed participant numbers, attitudes, theme, and transcript content related to the theme (see Figure 4.7). User types and attitudes were highlighted using different colors to facilitate identification. Regarding user types, blue represented internal users, while brown denoted external users. As for attitudes, light blue indicated a neutral stance, light green represented positive feedback, and yellow denoted negative sentiments. The transcripts were then examined thoroughly to gather related content and assign corresponding themes to produce code cards. The transcript analysis concluded with 44 code cards that were categorized into different themes to create an affinity map. The affinity map was composed of four main themes and 17 sub-themes (see Figure 4.8). This method clearly illustrates the general attitudes and the number of participants who corresponded with the same topics.



- now as a technical support, noticed that the end-user, such as the end-user in car factory, the integrators who are not that familiar with our system, they prefer to use teach pendant. in specific place, such as confidential plant or 3C plant, you are not allowed to bring with your laptop
- some robots are placed in confidential plant or clean room, where you are not allow to bring with your laptop, so it will be more convenient if you can configurate I/O signal in the teach pendant.
- customers or integrator who familiar with ABB product will more likely to use robotstudio for programming, while customers who unfamiliar with our product with use teach pendant for programming, teach position and I/O setting.

Figure 4.7: Transcript code card and example

Based on the overview, the affinity map reveals that users expect a better experience from OmniCore, as indicated by the predominance of cards with yellow highlights. In terms of the overall user experience, a prominent finding was that participants favoured the user experience of the previous version (IRC5) over that of OmniCore. It must be noted, however, that the reasons behind this preference went beyond the user experience. Users' familiarity with the IRC5 version was the major contributing factor that the theory of baby duck syndrome and familiarity bias could explain [44][45], as further discussed in the section 5.4. OmniCore was launched four years ago with a new interface, which led to users' reservations because they had to spend significant time learning and getting used to it. It is important to acknowledge that the OmniCore FlexPendant is still in its early stages of development, and further improvements are required to enhance its performance and user experience. For example, one participant noted that it took more time to navigate and find features compared to the previous version. Therefore, it is worth considering bringing back some of the good features of the previous version. Apart from ABB's product portfolio, competitors in various markets also offer a range of robotic solutions that seem to better suit local users' requirements in terms of logic and price. In addition to the drawbacks, some participants expressed satisfaction with ABB's product, particularly with the modern interface and state-of-the-art device. It should be noted that while one participant mentioned the interface was user-friendly for novice users, our target audience primarily comprises experienced users.

The second theme discusses the problem of user experience and its impact on productivity. A major issue affecting the user experience was the absence of necessary functionalities. Interviewees highlighted that OmniCore FlexPendant's performance could be significantly enhanced if it included all the functionalities present in the IRC5 version. In addition, participants were concerned about the redundancy present in workflow and navigation design. The primary objective of robot programmers is to quickly access functions and execute tasks efficiently. In addition, the complex data format of industrial programming makes it challenging to present a clear structure of data information. Moreover, the interface design emphasizes mouse-keyboard interaction, rather than touchscreen. For instance, some hotspots were too small to activate using fingers. To summarize, a visually appealing user interface is not a priority, but retaining high productivity is crucial.

When it comes to the role of the teach pendant in work processes, it was certain that the teach pendant played an indispensable role, particularly in factories or labs where using a laptop is restricted, making the teach pendant the only available option. The teach pendant is primarily utilized for fine-tuning and installation purposes. It is notable that customers usually use the OmniCore FlexPendant to control specific models exclusively compatible with the OmniCore version while opting for IRC5 for other models. ABB expects that programmers will soon adopt the OmniCore FlexPendant because of its market strategy and its revolutionary effect on the industry. It is, therefore, crucial for ABB to prioritize the user experience of OmniCore.

Finally, in the expectations section, several participants mentioned their desire for efficiency, such as wanting handy, simple, and easy access. It can be inferred that users expect the focus of improvement to be on reducing cumbersome interactions to make task execution more efficient. Overall, the primary aim to achieve in the design section should be to improve productivity and efficiency.

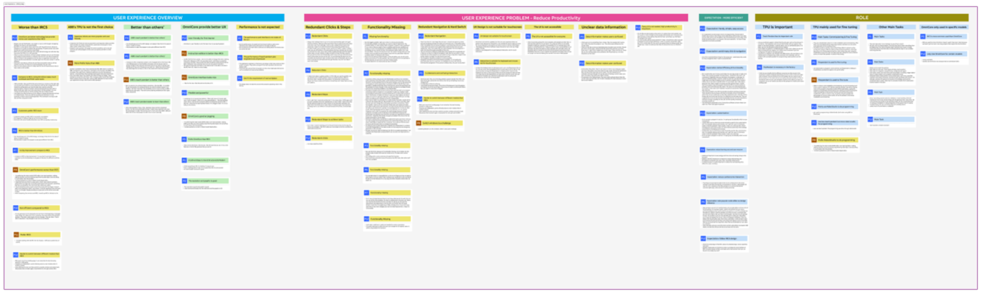
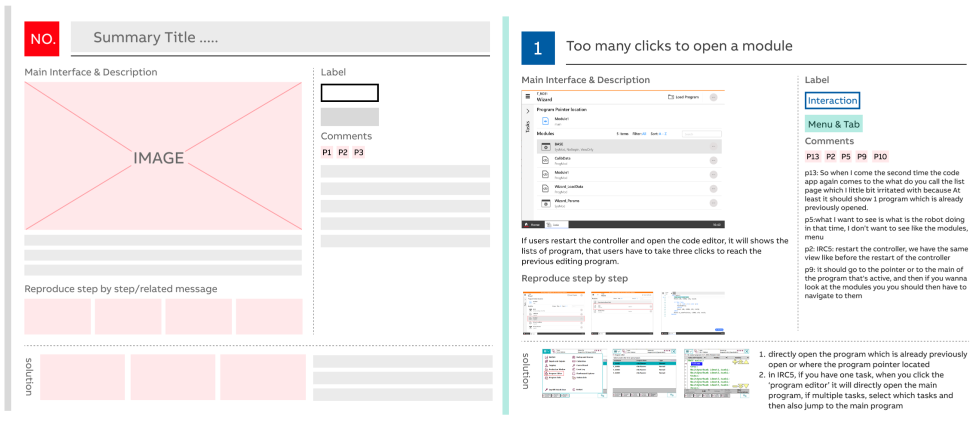


Figure 4.8: Affinity Map

4.3.3 MoSCoW Analysis

In order to dive into the UX challenge, an exhaustive MoSCoW analysis was conducted by thoroughly reviewing all the interview and field study transcripts to identify, list and categorize each issue related to UX. To easily manage the issues, a 'UX problem card' template was created. This card recorded the problem details, the affected interfaces, comments, and the solutions suggested by the participants (see Figure 4.9). In addition, Two distinct labels, represented by different colors, were used to categorize the problems. One of them described the type of UX issue, while the other indicated the function it related to. The first label set includes categories such as interaction, interface & layout, and information display, among others. The 'other' label referred to problems related to robot programming that did not occur in the code editor, and therefore were not included in the MoSCoW analysis. The second label set categorized the related functions into six types, including menu and tab, instruction, debug and edit, code line, data, and others. By analyzing all the recorded user research, 27 issue cards related to UX were generated.



If users restart the controller and open the code editor, it will shows the lists of program, that users have to take three clicks to reach the previous editing program.

p13: So when I come the second time the code app again comes to the what do you call the list page which I little bit irritated with because At least it should show 1 program which is already previously opened.

p5:what I want to see is what is the robot doing in that time, I don't want to see like the modules, menu

p2: IRC5: restart the controller, we have the same view like before the restart of the controller

p9: it should go to the pointer or to the main of the program that's active, and then if you wanna look at the modules you you should then have to navigate to them

1. directly open the program which is already previously open or where the program pointer located
2. in IRC5, if you have one task, when you click the 'program editor' it will directly open the main program, if multiple tasks, select which tasks and then also jump to the main program

Figure 4.9: UX Issue Card and example

Prior to the management of UX problem prioritization, the quantity of function labels on the cards was counted. The outcomes demonstrated that the majority of issues were caused by the menus and tabs' designer, which resulted in complex interactions. Issues in instructions, code lines, and data followed. Therefore, navigation will be the main focus of redesign (see Figure 4.10).

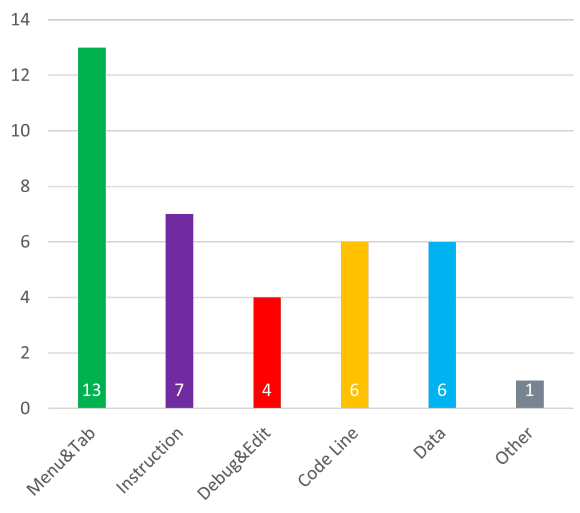


Figure 4.10: Function label calculation

Afterward, all the cards were systematically arranged in the MoSCoW coordinate system, with four areas: Would Improve, Should Improve, Could

Improve and Won’t Improve. The classification is based on the number of participants reporting the problem and the extent to which it disrupts the workflow. For example, a problem that was observed in only one field study, but significantly impeded the updating position data, would be categorized under the ’Must improve’ category. The category ’Should Improve’ included features that would significantly enhance the user experience through redesigning, although they were not immediately necessary. The ’Could Improve’ category includes issues that may or may not directly improve the user experience and may require significant time to redesign. The ’Won’t Improve’ category included features that were incompatible with the context and the OmniCore FlexPendant, such as those unique features found in other teach pendants. For instance, Kuka implemented a feature that enabled split-screen functionality, which was appreciated by a customer. Nonetheless, this feature was incompatible with the interface structure of the OmniCore FlexPendant and, therefore, was not taken into account.

Once the MoSCoW metric was created, it was observed that over half of the cards in the ’Must Improve’ and ’Should Improve’ categories pointed to issues related to menu and tab design, and 9 out of 19 problems were caused by interaction design (see Figure 4.11). The improvement of the design of tabs and the reduction of redundant steps are essential to enhancing user experience and usability.

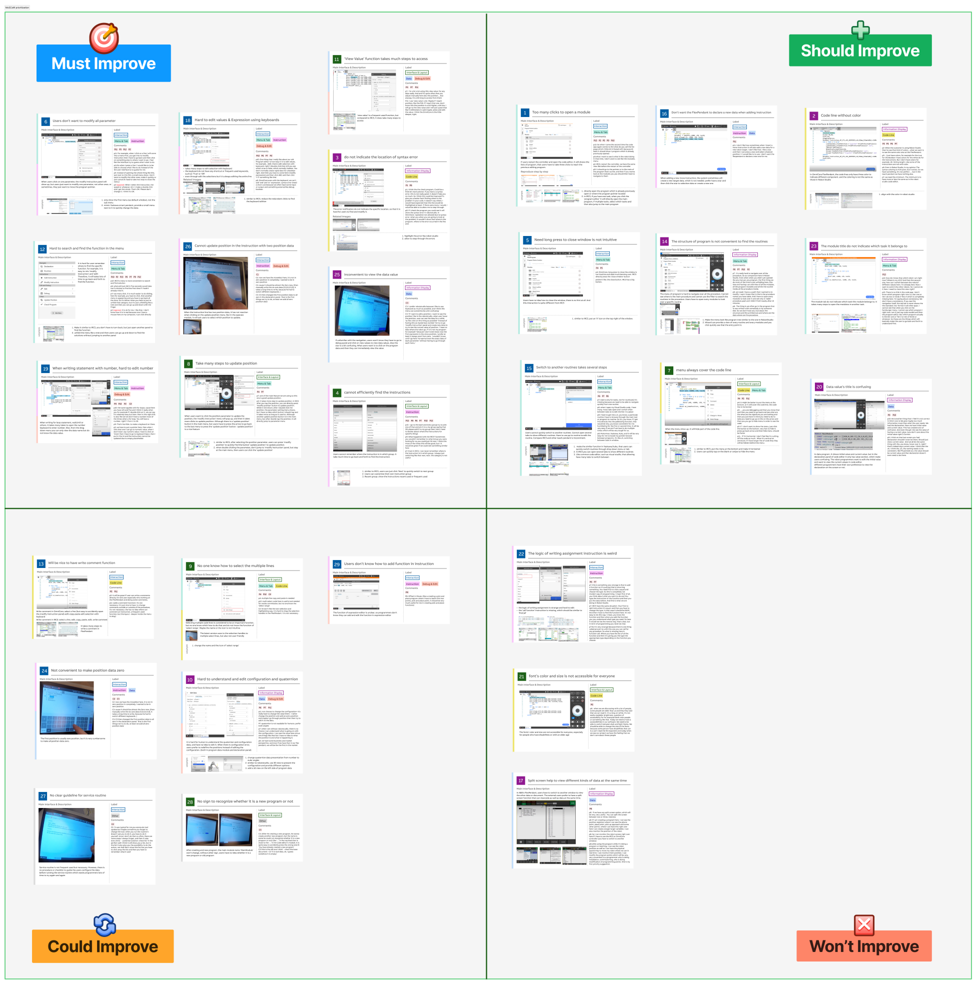


Figure 4.11: MoSCoW Prioritization Metric

4.3.4 Design Principle Ideation

Five key design principles were formulated based on the results of user research to guide design iterations before starting the design process. These principles were formulated to aid in designing an intuitive and user-friendly interface.

’big and bigger’ buttons for finger: The first principle to facilitate touch interaction was to maximize button size. When interacting with a mouse and keyboard, buttons should be larger than 42 pixels [46]. However, some buttons in the current interface were around 30 pixels, making them difficult to access and easy to accidentally touch. Thus, a bigger button size was deemed necessary. Buttons should have a width or height larger than 42 pixels. Due to screen size limitations, a size of at least 50 pixels is recommended.

Replace drop down menu: It is best to avoid using drop-down menus in the tablet application [47]. This decision was driven by issues that arose from interacting with the data list. There were many options for each type of data, but the scrolling space in the drop-down menus was limited, making it inconvenient for users to view and find options. Users would also often accidentally tap the space around the menu area, causing the drop-down menus to close. Using a fixed-size option list could eliminate this inconvenience.

Shortcuts and Customization: Enabling users to personalize the code editor based on their unique preferences and workflows is worth considering. Users have different working styles, and the functions and data they frequently use vary from project to project. Similar to customized workplaces in graphic design software, enabling users to define their own code editor can enhance productivity.

Design for touchscreen with limited size and sensitivity: Optimizing the design for touchscreen interfaces was critical given the constraints of screen size and sensitivity. The current design was deemed reasonable by participants when using the virtual OmniCore FlexPendant in RobotStudio. However, components need to be optimized for the physical device.

4.3.5 Redesign Scope

After thoroughly analyzing the transcript and prioritizing UX challenges, the issues that require improvement were identified. Before starting the design process, the types of issues were defined, whether they were bugs or user experience-related. Next, considering the available time and support, a set of issues to be tackled was selected. Other issues were deferred as they may require more extensive research or have minimal impact on the typical workflow. For instance, issues related to the 'Edit Expression' page encompassed several sections that required additional user research to identify specific details. Here is a list of the problems to be addressed and organized based on the MoSCoW Metric.

Must Improve	**Should Improve**	**Could Improve**
Users do not want to view all the parameters when modifying one parameter	Users do not want to view all the parameters when modifying one parameter	Users do not want to view all the parameters when modifying one parameter
Keyboards need improvement for quickly typing in	Code line without color to identify different elements	Configuration data is not readable for human beings
'View Value' function takes several steps to access	The module title does not indicate which task it belongs to	
Users cannot efficiently find the instructions	Long press to close the window is not intuitive	
Users make some effort in searching and finding the function in the menu	The structure of the program is not convenient for finding routines	
The 'check program' function does not indicate the location of syntax error	Switch to another routine needs a quick way	
The number input function is hidden deep in the menu	The right side panel covers part of the code	
Update position function needs quick access		
It is inconvenient to view the data value		

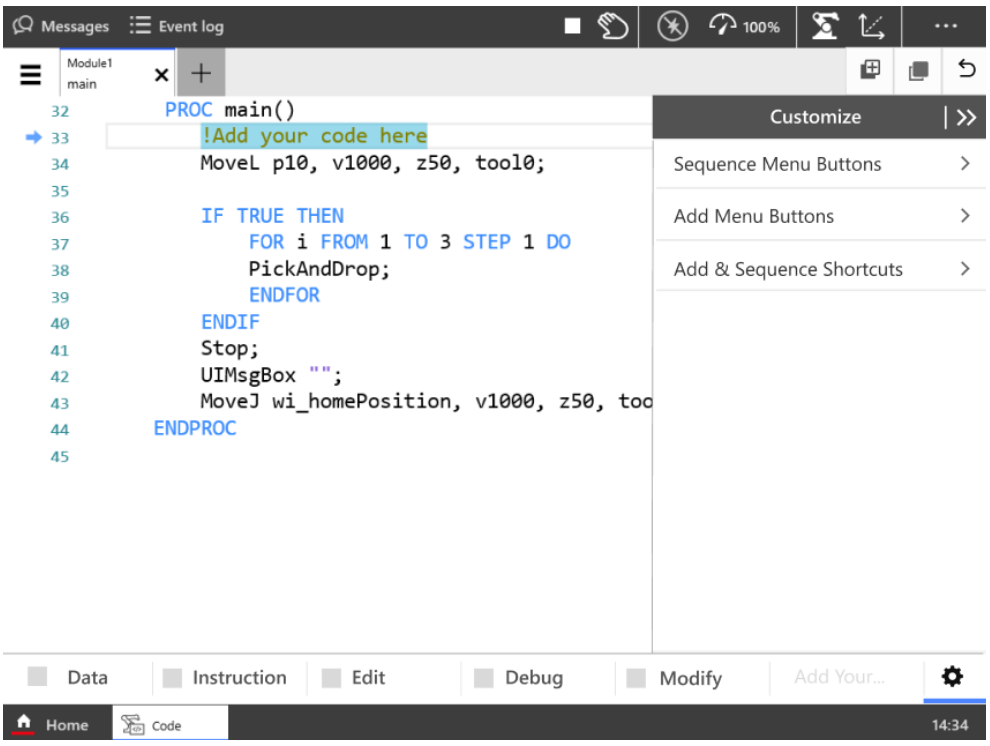
Table 4.1: Problem List

4.4 Prototype

Once the problem has been defined, the following step is to design solutions for addressing them. The design process began with the creation of wireframes and the first version of a high-fidelity interface. Expert review sessions were then conducted to evaluate the design. In response to feedback, I created high-fidelity and interactive prototypes for user testing. Figma was the primary design tool in this project.

4.4.1 First Version Solution

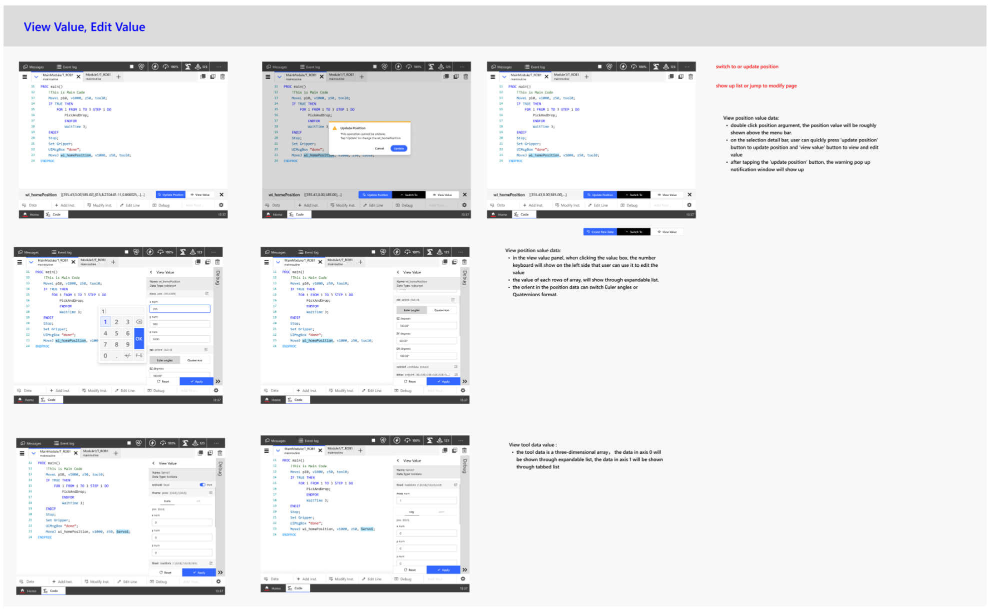
The solution concept was ideated using the screenshot of the current interface as a reference. This method involved building low-fidelity wireframes on top of the screenshots to showcase a rough solution without investing time in perfecting aesthetic details (see Figure 4.12). The preliminary solution was discussed with the supervisor to gather initial feedback. Subsequently, the initial version of a high-fidelity interface design was developed. The Figma page displayed various blocks, including a series of redesigned interfaces (see Figure 4.13). Each title indicated the content being redesigned and attached notes described the changes made. The interfaces utilized assets from the latest ABB Robotics design system to align its design style with other products. The interface design aims to be realistic and precise by standardizing font size, color, margin and component sizes. Because precision played a crucial role in implementing design while creating interactive prototypes. One more reason is the touchscreen's limitations in size and sensitivity. Therefore, the design details became crucial to evaluate in the subsequent process.



```
  PROC main()
    !Add your code here
    MoveL p10, v1000, z50, tool0;

    IF TRUE THEN
        FOR i FROM 1 TO 3 STEP 1 DO
        PickAndDrop;
        ENDFOR
    ENDIF
    Stop;
    UIMsgBox "";
    MoveJ wi_homePosition, v1000, z50, too
ENDPROC
```

Figure 4.12: Low-fidelity design example



View position value data:
- double click position argument, the position value will be roughly shown above the menu bar.
- on the selection detail bar, user can quickly press 'update position' button to update position and 'view value' button to view and edit value
- after tapping the 'update position' button, the warning pop up notification window will show up

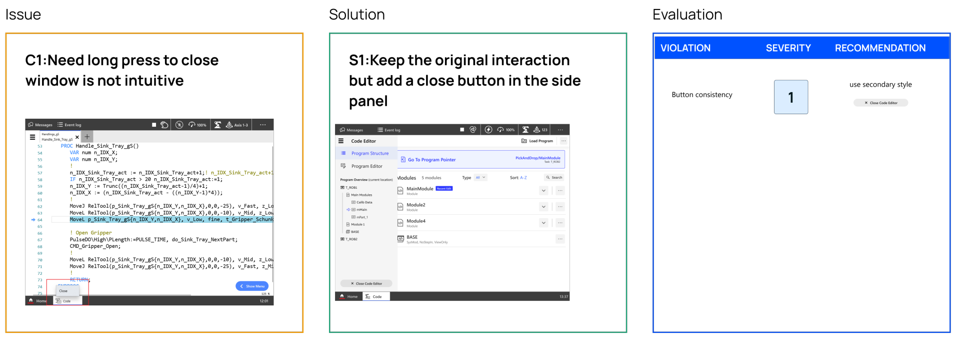
View position value data:
- in the view value panel, when clicking the value box, the number keyboard will show on the left side that user can use it to edit the value
- the value of each rows of array, will show through expandable list.
- the orient in the position data can switch Euler angles or Quaternions format.

View tool data value :
- the tool data is a three-dimensional array， the data in axis 0 will be shown through expandable list, the data in axis 1 will be shown through tabbed list

Figure 4.13: First version design example

4.4.2 UX Expert Review

Following the initial redesign, the first round of evaluation with UX colleagues was conducted to refine the design. Three UX designers responsible for different programming solutions at ABB were invited to this phase and the review meeting was held online. The Figma file for design review was prepared prior to the review meeting. The page is divided into three main sections (see Figure 4.14). The left block displays the description of each problem and the affected interface, whereas the central block shows the solutions and their corresponding interface. The right side is a table that records the evaluation result of each design solution, including whether it has UX violations, the severity of the violation, and the recommendation to modify the design. The review meetings commenced with an introduction to the process, confirmation of data confidentiality and the request to record. The author then presented problems and solutions one by one, soliciting suggestions from the participants. In regard to design violating user experience and usability, participants demonstrated other products' design solutions and proposed their own ideas. Since all participants were experienced designers, they paid close attention to the details of each element, including color, shape, size, component type, and consistent design, which is essential in creating a realistic user interface. Since there were more than 50 interfaces to evaluate,



```
PROC Handle_Sink_Tray_gS()
   VAR num n_IDX_X;
   VAR num n_IDX_Y;
   !
   n_IDX_Sink_Tray_act := n_IDX_Sink_Tray_act+1;! n_IDX_Sink_Tray_act+
   IF n_IDX_Sink_Tray_act > 20 n_IDX_Sink_Tray_act:=1;
   n_IDX_Y := Trunc((n_IDX_Sink_Tray_act-1)/4)+1;
   n_IDX_X := (n_IDX_Sink_Tray_act - ((n_IDX_Y-1)*4));
   !
   MoveJ RelTool(p_Sink_Tray_gS{n_IDX_Y,n_IDX_X},0,0,-25), v_Fast, z_Lo
   MoveL RelTool(p_Sink_Tray_gS{n_IDX_Y,n_IDX_X},0,0,-10), v_Mid, z_Low
   MoveL p_Sink_Tray_gS{n_IDX_Y,n_IDX_X}, v_Low, fine, t_Gripper_Schunk

   ! Open Gripper
   PulseDO\High\PLength:=PULSE_TIME, do_Sink_Tray_NextPart;
   CMD_Gripper_Open;
   !
   MoveL RelTool(p_Sink_Tray_gS{n_IDX_Y,n_IDX_X},0,0,-10), v_Mid, z_Low
   MoveJ RelTool(p_Sink_Tray_gS{n_IDX_Y,n_IDX_X},0,0,-25), v_Fast, z_Mi
   !
   RETURN;
```

VIOLATION	SEVERITY	RECOMMENDATION
Button consistency	1	use secondary style

Figure 4.14: Expert design review template

the three participants didn't assess all the solutions. Two designers reviewed each solution to gain diverse perspectives. Expert feedback instructed to perfect the high-fidelity prototype.

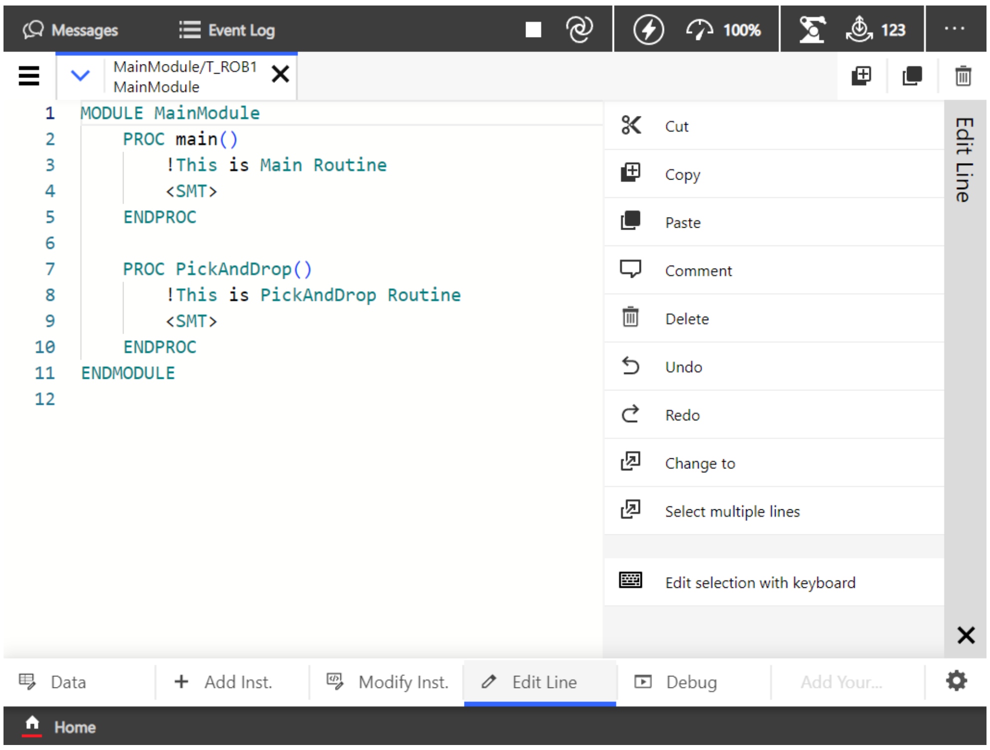
4.4.3 High-Fidelity Prototype

The High-Fidelity Prototypes were created based on feedback from the UX Expert review session. During this phase, the design aimed to achieve pixel-perfection. To maintain design consistency, components with various variants and statues were constructed, and their sizes were adjusted to fit the touchscreen. The main outcome includes two main interfaces, five operation panels, eight instruction configuration panels, and two types of keyboards. There are four significant changes in the solutions. Firstly, the vertical menu was replaced with a horizontal one, which allows quick switching between panels. Next, the accordion list was widely used to replace the drop-down menus or menus with submenus in order to reduce navigating between pages. Predefined and customization functions were introduced to allow users to define their own code editors for improving efficiency. Once the High-Fidelity prototype was completed, the redesigned interface was realistic and accurate enough to be turned into interactive prototypes for future testing in the next phase. The interfaces can be viewed in the Section 4.6.

4.4.4 Interactive Prototype

After confirming the prototype details, an interactive prototype was created to provide an immersive experience during user testing. Due to time and programming constraints, the interactive prototype was developed with the help of a front-end programmer. This prototype was a web-based code editor that used the React library and the Monaco Editor. The functions were hardcoded. Monaco Editor is a browser-based code editor that is also utilized in RobotStudio Cloud. Finally, the prototype can be loaded in the tablet's browser and interacted with using a finger. The prototype's fundamental interface and interaction designs have been implemented, enabling users to perform basic programming tasks. However, some interactions necessitate specific workflows to be triggered. The following list presents all the supported functions:

- Add instruction & Modify instruction: MoveL, MoveJ, WaitTime, ProcCall, Set, Reset
- 'Add instruction' panel: navigate among menus, search for instruction, favorite instruction, set up instruction panel
- 'Data' panel: view data detail, modify data detail
- 'Edit line' panel: cut, copy, paste code
- 'Debug' panel: Check program & modify syntax error
- 'Customization' panel: order menu buttons, add new menu button, add & order shortcuts.
- Switch routine/module through the drop down menu in the title tab.
- View routine/module/task on the program structure page
- Edit IF expression with keyboard;
- Change value with number keyboard
- Add new data with specific name: jposA, jposB, jposC
- Double tab position/speed/zone data to trigger detail bar
- View specific data detail
- Update predefined position data



```
MODULE MainModule
    PROC main()
        !This is Main Routine
        <SMT>
    ENDPROC

    PROC PickAndDrop()
        !This is PickAndDrop Routine
        <SMT>
    ENDPROC
ENDMODULE
```

Figure 4.15: A screenshot of interactive prototype

4.5 Test

Once the prototype was complete, it was time to test whether the design had addressed the challenges identified in the user research. A within-subject experiment was used in which all participants were required to program a robot using both two code editors. The comparison between the two code editors shows the impact of the redesign. The goal of the user testing is to collect feedback and determine which parts need further adjustment. The subsequent sections will detail the test methodology and results.

4.5.1 User Testing

A within-subjects study was carried out with 7 participants, 1 for pilot study and another 6 for contributing to the final result. All participants completed three programming tasks of varying levels using two code editors. Two questionnaires and a short interview were used to gather feedback about their experiences. The study's focus was on identifying design defects causing confusion and interruption in the workflow.

4.5.1.1 Participant

The study recruited robot programmers who could take part in the user test on-site. In the end, seven participants comprising four user profiles were selected. These included two PhD candidates in robotics, one novice with only IRC5 training, two software developers, and two experienced OmniCore FlexPendant users. Regarding gender, two were female, and the remaining five were male. In terms of composition, two were customers who are experienced user of OmniCore FlexPendant. One participant who is a PhD candidate in robotics participated in the pilot study. Apart from these three individuals, the other four participants were employees of ABB. This diverse group was selected to comprehensively evaluate the new design.

4.5.1.2 Material

For user testing, an interactive prototype, three predefined programming tasks, a physical OmniCore FlexPendant, a recording device, and online questionnaires were prepared. The interactive prototype was loaded in the browser of an iPad Pro. The OmniCore FlexPendant was updated to the latest software version. It was then connected to a laptop and RobotStudio to power the device and synchronize documents. The three programming tasks included drawing a triangle, fine-tuning, and picking up and dropping objects. The tasks ranged in difficulty from easy to mid-level. The code for both the initial and expected finish programs was pre-set. Based on the provided code sample, users are expected to modify the initial program accordingly. Recordings of each task's robot movement in RobotStudio were made to aid in explaining the tasks. The interaction process and interview were recorded using the camera and voice recorder on a phone. Two online questionnaires, the short version of the User Experience Questionnaire and the Standard System Usability Scale, were created using Qualtrics. The short version of the User Experience Questionnaire includes 10 questions to evaluate the metrics related to attractiveness, efficiency, clarity, and dependability.

4.5.1.3 Pilot Study

The pilot study took place in a study room at KTH Royal Institute of Technology (KTH) library with a PhD candidate in robotics. The aim was to assess whether the whole process could be well controlled by the researcher and to get a chance to familiarize with the procedure. The data collected during the pilot study would not contribute to the result due to changes in the process and recording mistakes. The pilot study revealed a major problem: the researcher was not able to pay attention to several devices at the same time, such as a mobile phone, a laptop, a second screen and two code editors. Multitasking can interfere with the researcher's observation. Therefore, in the modified user testing procedure, some parts were cut out for better observation. For example, participants did not need to simulate the robot movement process but could only focus on the coding section. As a result, both participants and researchers were not required to look at the second screen every time a movement instruction was added. The pilot study lasted for 60 minutes, which is deemed appropriate for this research activity.

4.5.1.4 Procedure

Seven steps make up the entire process: introduction, exploration, task 1, task 2, task 3, questionnaire, and interview (see Appendix C). The invitation was sent via email and the schedule was arranged using Microsoft Teams. In preparation, the test devices and files were carefully set up in advance in the meeting room. The testing began with an introduction to the testing procedure, the two code editors, and the confirmation of the consent form and recording request. The test sequence was randomly assigned to ensure objectivity. Later, participants were given five minutes to explore two code editors with the 'playground' module. When performing the first two tasks, the participants were allowed to ask the researcher any questions they had if they encountered difficulties to help them become familiar with the code editor. For the third task, the participants were required to complete it individually. Throughout the test, the participant skipped the jogging process and only had to concentrate on the programming part.

The first task was to control a robot to draw a triangle. More specifically, the participant was required to create three new position data and program the robot using 'MoveJ,' 'MoveL' and 'WaitTime' actions. This task was designed because drawing a shape is the first program that learners write during training. The second challenge was to fine-tune a previously created program. Participants must use functions and features, such as the keyboard, to refine a program with syntax errors. This includes checking, modifying instructions, updating positions and changing data options. Fine-tuning was the most common task for a robot programmer when working with a teach pendant. In the last task, participants were required to create a program for

’pick and drop’ objects from scratch. Writing programs for this type of action was one of the main tasks performed by industrial robot programmers.

Upon completing all tasks, participants should complete an online questionnaire containing the UEQ and SUS and subsequently engage in a brief discussion with the researchers. In the short interview, the researcher enquired about their code editor preference, preferred and disliked features, as well as their overall user experience and suggestions for improvement. Next, the interruption during user testing is discussed, including participants’ workflow, behavioral intentions, and expected solutions. The user testing concludes with a section expressing gratitude.

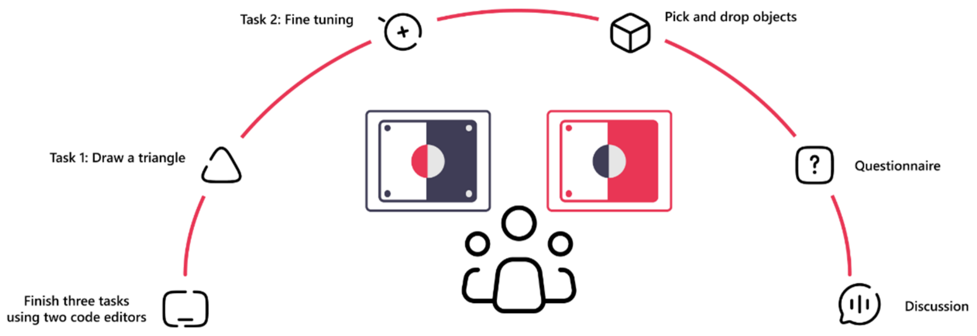


Figure 4.16: User Testing Procedure

4.5.2 Data Analysis & Findings

The data analysis consists of four parts: time spent, user experience questionnaire, system usability scale and interview feedback. This section will detail the analysis of each metric.

4.5.2.1 Time Consumption

The box plot showed that the new design reduces the time required for programming, making the workflow more efficient. Overall, the ’New Design’ required less time compared to the ’Current Design’. Furthermore, there is a visible trend of median time increasing from Task 1 to Task 3, which corresponds to the level of task complexity. Both editors took longer to complete more complex tasks, but the increase was less pronounced for the ’New Design,’ indicating that it handled increasing complexity better. However, participants might have completed Task 2 more quickly than Task 1 because they had become more familiar with the code editor after performing Task 1. To assess the time-saving performance of the new design, the average

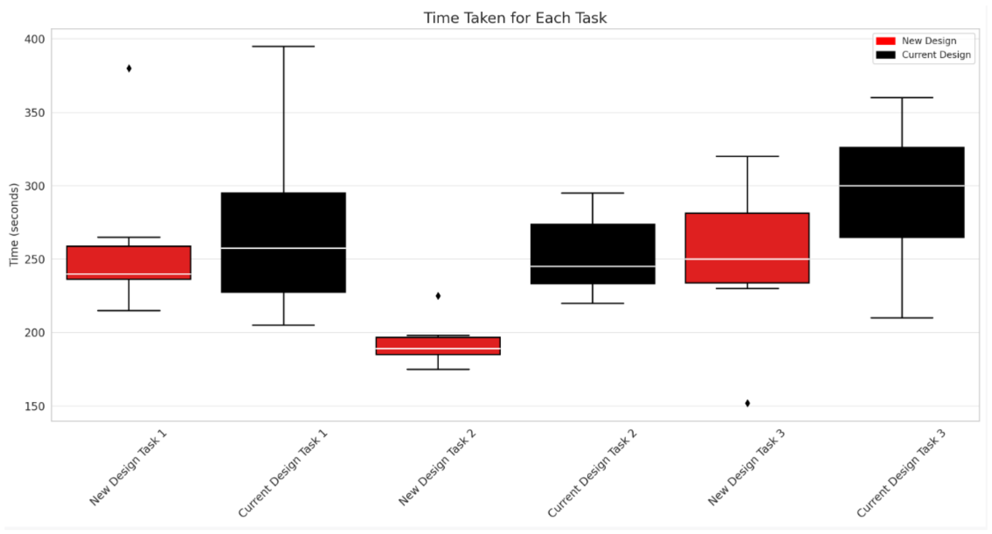


Figure 4.17: Time Consumption Calculation Box Plot

Current Design completion time of each task minus the average New Design completion time, the result showed 12.09%, 15.59%, and 17.13% time savings for Task 1, Task 2, and Task 3, respectively. The size of the boxes suggests that the time consumption variability of the 'Current Design' is greater than that of the 'New Design.' Overall, the results indicate that the 'New Design' outperformed the 'Current Design' for all participants and across all tasks. To summarize, the "New Design" shows lower median times and less variability than the "Current Design," indicating its potential to provide more efficient solutions for robot programmers.

4.5.2.2 User Experience Questionnaire

This Short-Version Experience Questionnaire evaluated ten metrics based on user ratings that range from 1 to 7, with a higher score indicating better performance. The metrics include "Enjoyable," "Friendly," "Efficient," "Practical," "Organized," "Easy to learn," "Easy," "Clear," "Predictable," and "Meets Expectations". The mean was calculated for each metric to analyze the data.

All metrics indicate that the New Design consistently outperforms the Current Design. The 'Efficient' and 'Organized' metrics show the most

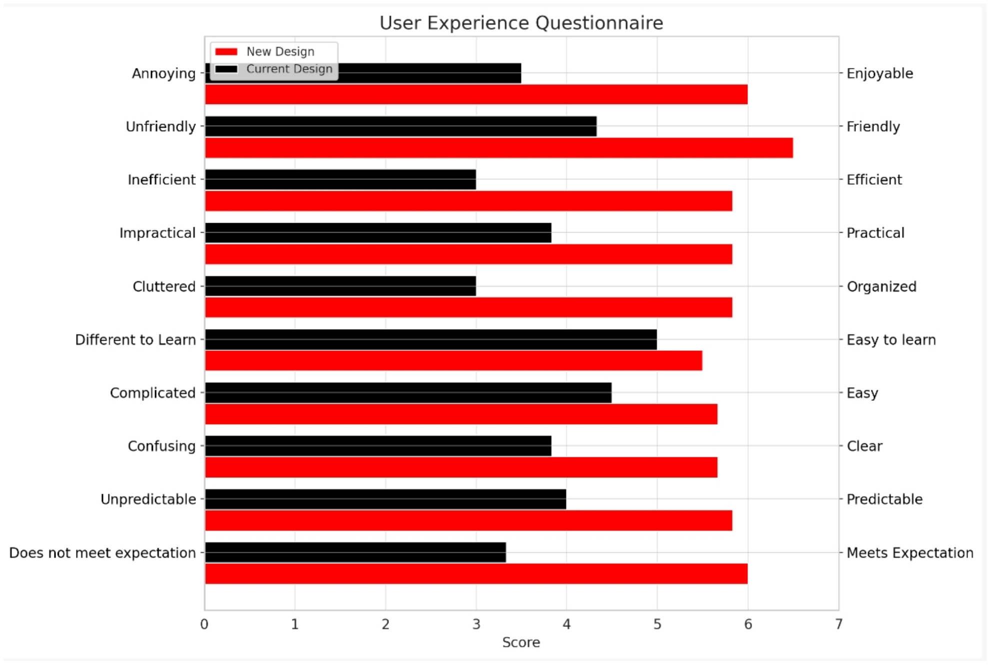


Figure 4.18: User Experience Questionnaire

significant improvements, with a difference of 2.83 points corresponding to a percentage increase of 94.4%. Therefore, users considered the New Design to be significantly more efficient and better organized than the Current Design. The metrics 'Enjoyable' and 'Meets Expectation' showed significant improvement, with an average increase of 2.5 and 2.67 points respectively. These findings indicate that users perceived the New Design to be more enjoyable and satisfactory than the Current Design and better meeting their expectations. Other metrics indicate an improvement of scores ranging from 1.8 to 2.17. However, the 'Easy to learn' metric showed the smallest increase with a mean difference of 0.5 points. This outcome was not unexpected since it aligns with the user study's conclusion that the OmniCore FlexPendant is novice user-friendly. These improvements indicate that the design changes were positively received by users and effectively addressed some UX obstacles.

4.5.2.3 System Usability Scale

The standard SUS was used to measure the usability of two code editors. The SUS comprises ten metrics that are grouped into positive expression (odd-numbered) and negative expression (even-numbered) categories [39]. I separated two types of expressions and calculated the score for each metric as well as the SUS score for the tool to obtain the result.

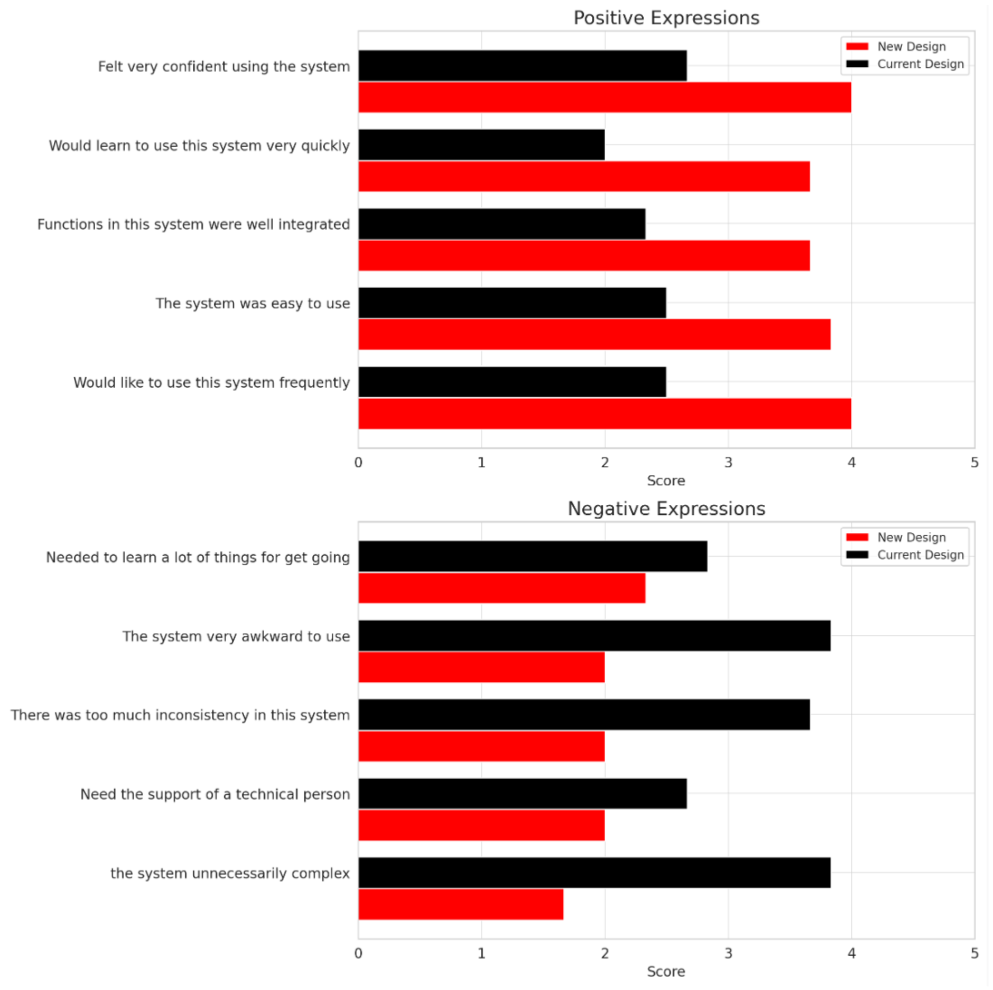


Figure 4.19: System Usability Scale

The analysis of the positive metrics shows that the New Design performs better than the Current Design in all five metrics. A mean difference of 1.33 to 1.67 points was observed in all metrics, indicating a 50% to 83.33% percentage improvement. The results imply that users found the New Design easier to learn, more user-friendly, better integrated and more confident to use compared to the Current Design.

Regarding the negative items, the New Design also performed better. The statement, ”The system was unnecessarily complex,” showed a significant mean difference of 2.17 points, equating to a 56.52% improvement. This emphasised that the users found the New Design less complex and simpler to use compared to the Current Design. Furthermore, the other negative points, such as ”Requiring technical support,” ”Inconsistent interface” and ”Awkward to use system,” had better results also. The item with the least difference of 0.5 points, ’Needed to learn a lot of things before I could get going,’ demonstrates the consistency with the result in the User Experience Questionnaire that the current OmniCore FlexPendant is user-friendly for novice users. These findings demonstrate that the New Design is less inconsistent, less awkward to use, and requires less prior learning.

Finally, according to the SUS score calculation rule, the New Design gained 72.92 points. The SUS score graphs interpreted the score of the new design as a B-, indicating satisfactory usability, although room for improvement still remains [48]. The next sections will detail the defects identified.

4.5.3 Interview Feedback and Discussion

Discussions with participants revealed a consensus that the new design provides a more user-friendly solution. The elimination of redundant steps enabled participants to accomplish tasks efficiently, indicating that the design was moving in the right direction. The positive feedback from interviews corresponded with the questionnaires and time consumption results. Participants expressed appreciation for some features, notably the reintroduction of the horizontal menu, the ability to customize the interface, the quick ways to locate instructions and syntax errors, and the shortcuts. In conclusion, the changes in New Design significantly improved the user experience and usability of the code editor.

It is worth noting that users of OmniCore and IRC5 have developed their own workflows, which has led to the emergence of various ideas in the discussion. A debate arose regarding whether a single or double tap should be used to activate the parameter detail bar. Although some participants were used to double-tapping on the parameter or code line to select it, others preferred to double-tap to trigger the modification panel. Moreover, certain users exhibited distinct preferences in terms of prioritizing frequently-used instructions and data when compared to favouring specific options. While some users would like the system to automatically recognize and prioritize frequently used options, others wish to have the independence to define their option lists. The possibility of providing users with more flexibility to customize their interaction behaviour was considered. However, increasing freedom might lead to reduced efficiency and elevated complexity. Therefore, in this case, it is better to guide users towards a specific workflow in this solution.

In addition to acceptance and debate, participants have identified a few areas that could be improved further. Within the WaitTime instruction configuration panel, the participants agreed that having a number input box streamlined the process of entering numbers. Nevertheless, they found it tedious to repeatedly type in the same number and suggested having a predefined number button. Additionally, the participants appreciated the shortcut to edit the code without opening the 'Edit Line' panel. However, they found that the shortcut buttons' size was small and the meaning of the icons was confusing. A plausible solution would be to enlarge the button and add a text label to clarify its functions. In addition, the 'MoveJ' and 'MoveL' are the two most frequently used instructions that can be input through shortcuts, as per their preference. Regarding the configuration panel for the Movement instruction, the long list of data options always causes parameter titles to be hidden, leading to users missing some of the parameter settings and having to make modifications later. Hence, a sidebar menu could be a better solution than an accordion list in this context. The exact value of each option was not deemed important by the users. Therefore, in the iteration design, the decision was made not to show the exact value of most data types, keeping them only for numbers and signals. This is because it can be challenging for users to remember associations, such as the number '7' being assigned to the letter 'a'.

To summarise, all participants accepted the new design, and the direction for future improvements was confirmed. The outcome contributed to the final concept and the iteration version of the solution will be shown in the final design.

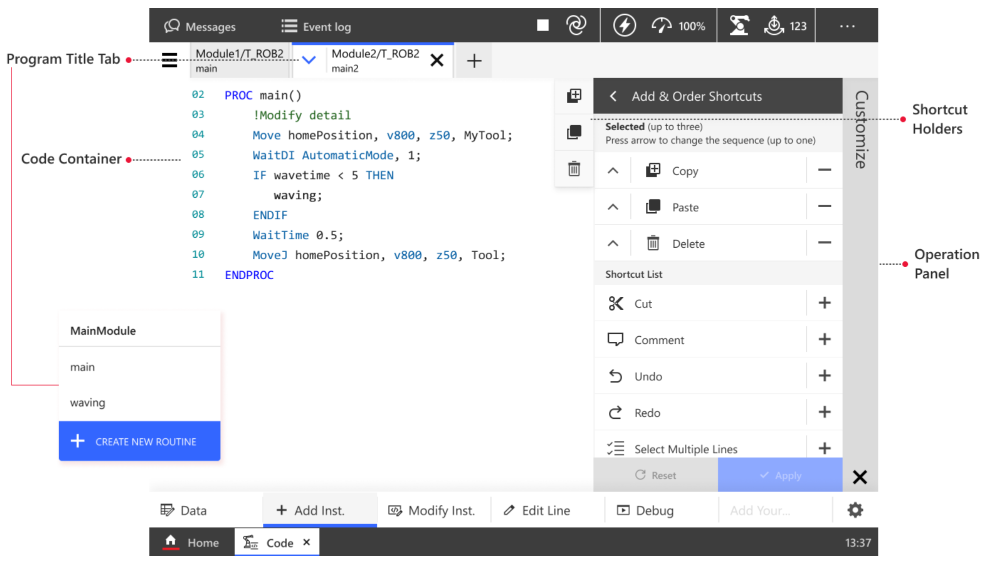
4.6 Final Concept

The final phase is the presentation of new solutions. The two main pages of the new code editor and examples of panels were presented, along with solutions to ten key issues.

4.6.1 Interface Overview

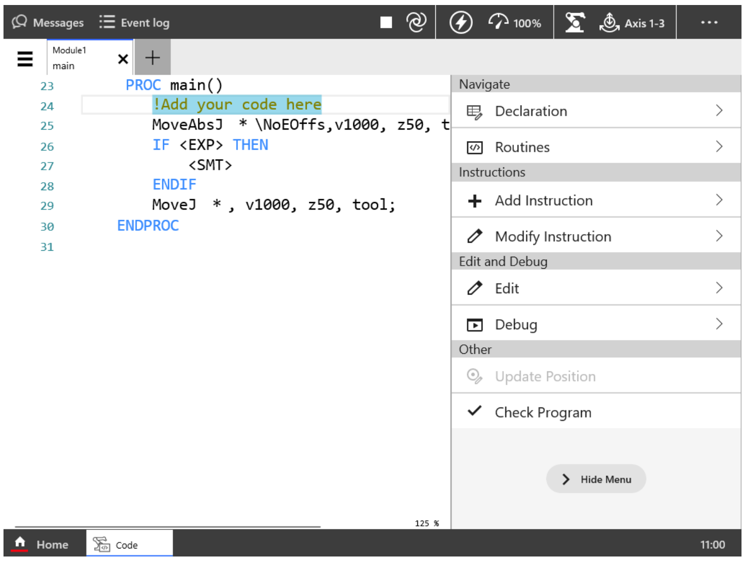
The Program Editor page is composed of four main components (see Figure 4.20). In the middle section, there is an area designated for displaying code.

The code was colour-coded to improve its readability. The tab bar with program details is located above. Each tab clearly displays the name of the module, task, or routine and includes a button to view the list of routines. Below the code holder is a horizontal menu that allows quick panel switching, which is the most significant change from the current version (see Figure 4.21). Once the button on the menu is tapped, the corresponding operation panel will appear on the right side of the code containers.



```
PROC main()
    !Modify detail
    Move homePosition, v800, z50, MyTool;
    WaitDI AutomaticMode, 1;
    IF wavetime < 5 THEN
        waving;
    ENDIF
    WaitTime 0.5;
    MoveJ homePosition, v800, z50, Tool;
ENDPROC
```

Figure 4.20: Program Editor New Solution



```
PROC main()
    !Add your code here
    MoveAbsJ  * \NoEOffs,v1000, z50, t
    IF <EXP> THEN
        <SMT>
    ENDIF
    MoveJ  * , v1000, z50, tool;
ENDPROC
```

Figure 4.21: Program Editor Current Version

The Program Structure page consists of three main parts (see Figure 4.22). On the top bar, there is a hamburger button on the left side, which enables users to open a side menu for quick switching to the Program Editor page. On the left side of the page, there is a fixed menu which presents the task list. Users can tap on this menu to view the files associated with different robots. A list of files is placed on the right side, arranged in an accordion style. The user can expand or collapse the list of procedures to view procedures from different modules. This reduces redundant clicking compared to the current version, where users have to jump back and forth (See Figure 4.23). Users can also open the program with Program Pointer by tapping on the blue quick access button.

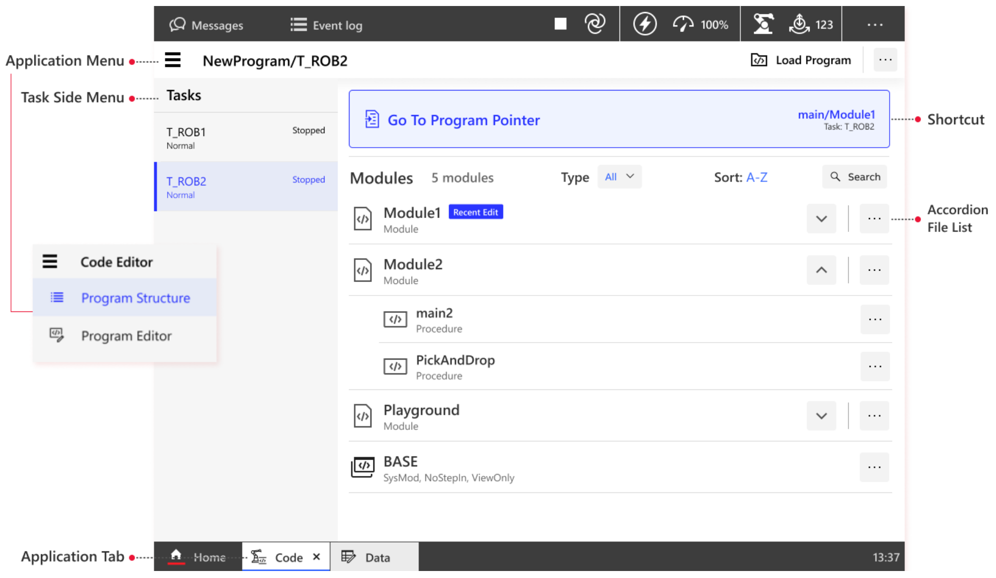


Figure 4.22: Program Structure New Solution

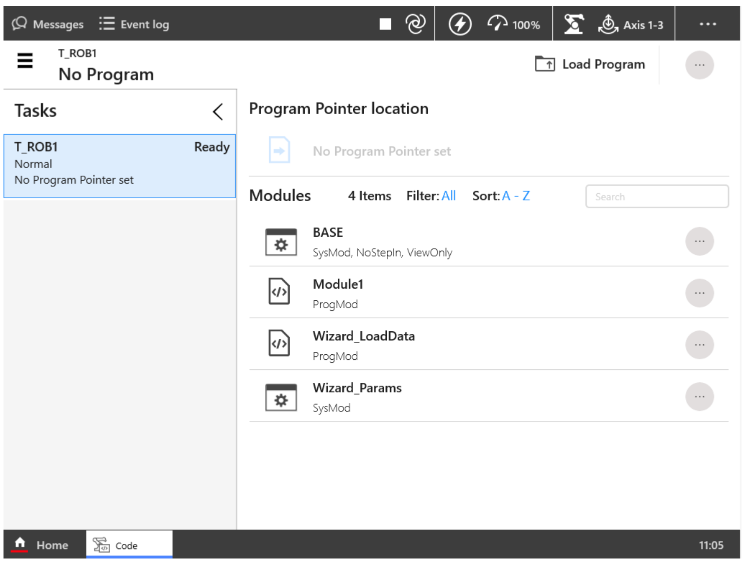
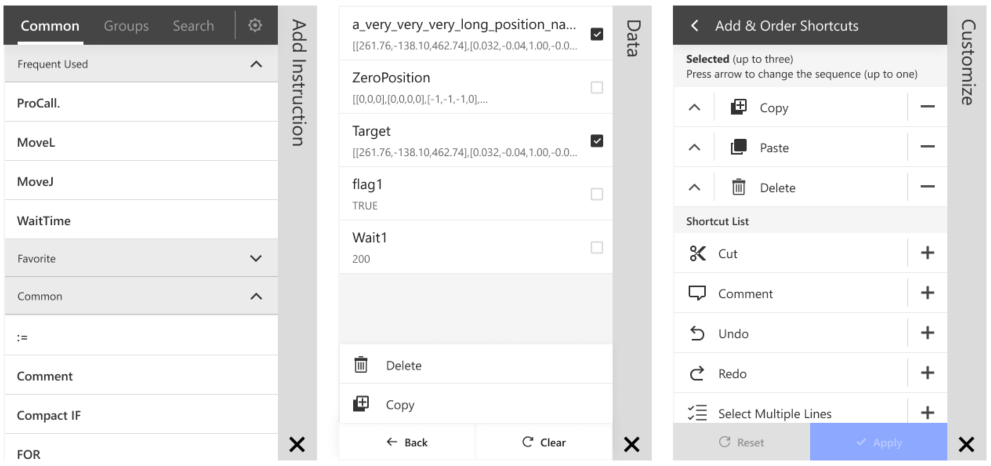


Figure 4.23: Program Structure Current Version

What's more, five operation panels and various property panels have been redesigned to support programming tasks. The design details can be viewed in the following sections.



a_very_very_very_long_position_na... [[261.76,-138.10,462.74],[0.032,-0.04,1.00,-0.0...

ZeroPosition [[0,0,0],[0,0,0,0],[-1,-1,-1,0],...

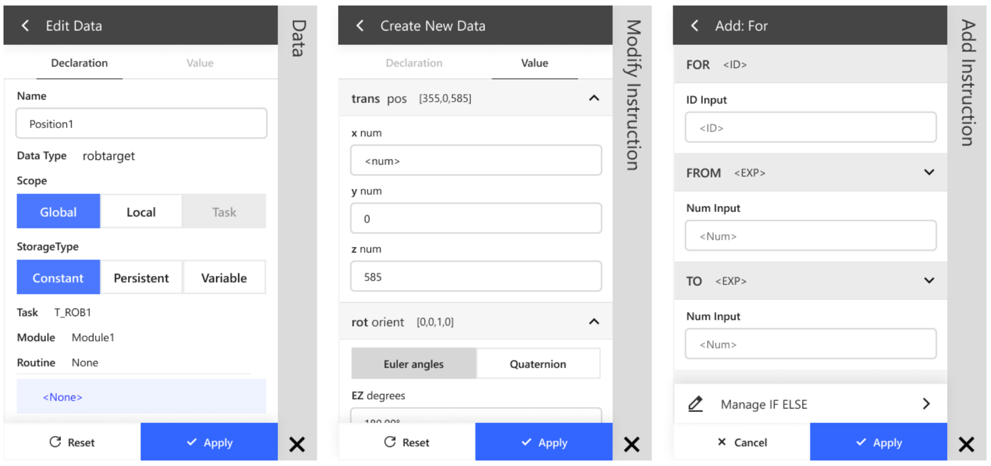
Target [[261.76,-138.10,462.74],[0.032,-0.04,1.00,-0.0...

flag1 TRUE

Wait1 200

Selected (up to three)
Press arrow to change the sequence (up to one)

Figure 4.24: Operation Panel Examples (from left to right: Add Instruction, Data, Customization)



Name: Position1
Data Type: robtarget
Scope: Global | Local | Task
StorageType: Constant | Persistent | Variable
Task: T_ROB1
Module: Module1
Routine: None
<None>

trans pos [355,0,585]
x num: <num>
y num: 0
z num: 585
rot orient [0,0,1,0]
EZ degrees

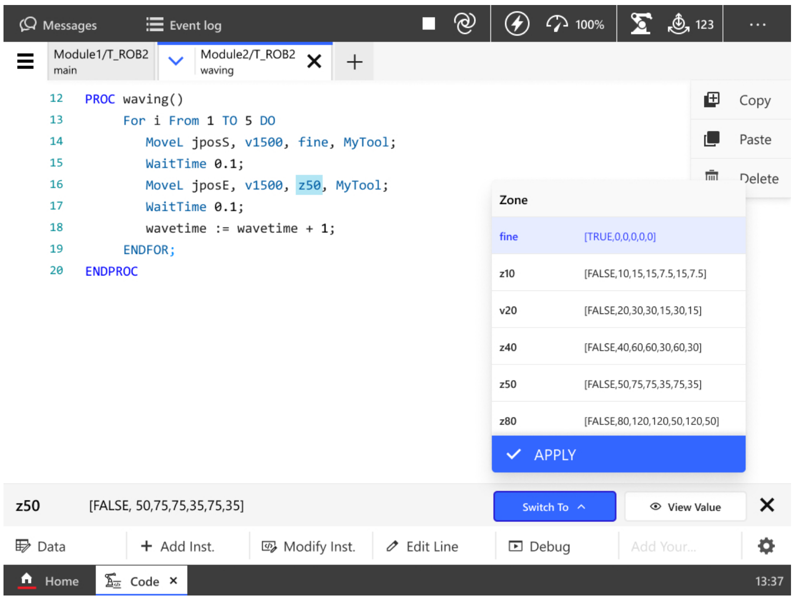
FOR <ID>
ID Input: <ID>
FROM <EXP>
Num Input: <Num>
TO <EXP>
Num Input: <Num>

Figure 4.25: Property Panel Examples (from left to right: Data Declaration, Value, For Instruction)

4.6.2 Problem & Solution

Issue 1: When users double-tap on the parameter to change it, the instruction will show all parameters. However, users want to view only the selected parameter.
Solution 1: The feature has been updated so that users can double-tap on the desired parameter to open a detailed bar. This bar will only show information

about the selected parameter and provide a list of options to modify it.



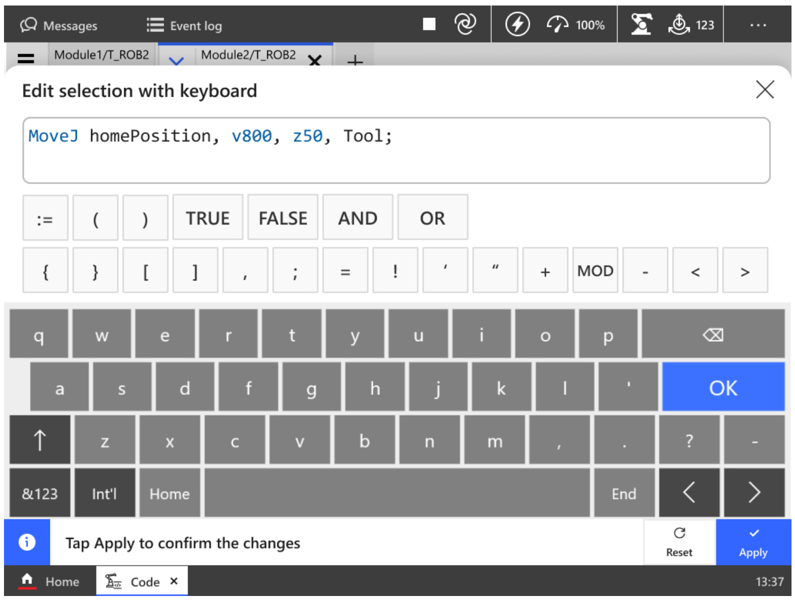
```
PROC waving()
    For i From 1 TO 5 DO
      MoveL jposS, v1500, fine, MyTool;
      WaitTime 0.1;
      MoveL jposE, v1500, z50, MyTool;
      WaitTime 0.1;
      wavetime := wavetime + 1;
    ENDFOR;
ENDPROC
```

Zone	
fine	[TRUE,0,0,0,0,0]
z10	[FALSE,10,15,15,7.5,15,7.5]
v20	[FALSE,20,30,30,15,30,15]
z40	[FALSE,40,60,60,30,60,30]
z50	[FALSE,50,75,75,35,75,35]
z80	[FALSE,80,120,120,50,120,50]

z50 [FALSE, 50,75,75,35,75,35]

Figure 4.26: Solution 1 interface: parameter detail bar

Issue 2: To type symbols or logic words on keyboards, users have to go through three or more steps to find them.
Solution 2: In order to make it easier to access symbols or logic words, the new design has positioned frequently used symbols or logic words above the keyboard.



Edit selection with keyboard

```
MoveJ homePosition, v800, z50, Tool;
```

Tap Apply to confirm the changes

Figure 4.27: Solution 2 interface: edit selection with keyboard

Issue 3: The user found it bothersome to locate instructions efficiently because it was hard to remember which group an instruction belonged to. As a result, users had to flip between pages to search for the desired instruction. The button to clear search words is too small to access.

Solution 3: The new Instructions panel incorporates four tabs. Under the Common tabs, users can view their favorite instructions and the frequently used instructions detected by the system. On both the Common and Groups pages, the general menu has been replaced by an accordion list, reducing the time needed to check each group's subpage. Furthermore, the button to clear input on the search page has been made larger and more easily accessible. Lastly, on the Settings page, users can manage their instruction groups and favorite instructions.

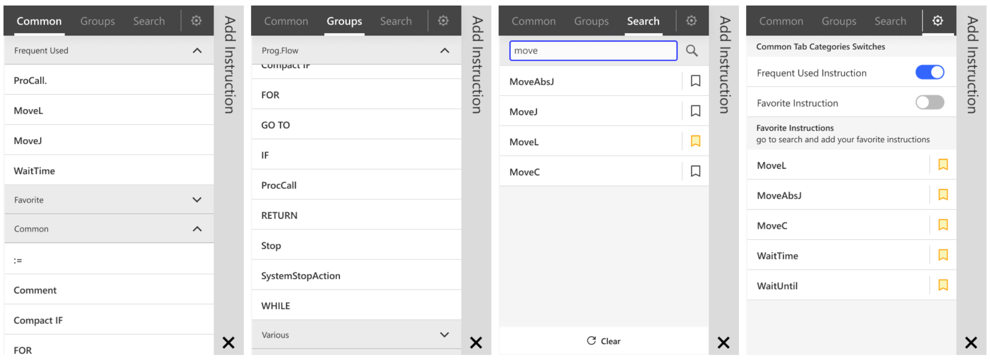
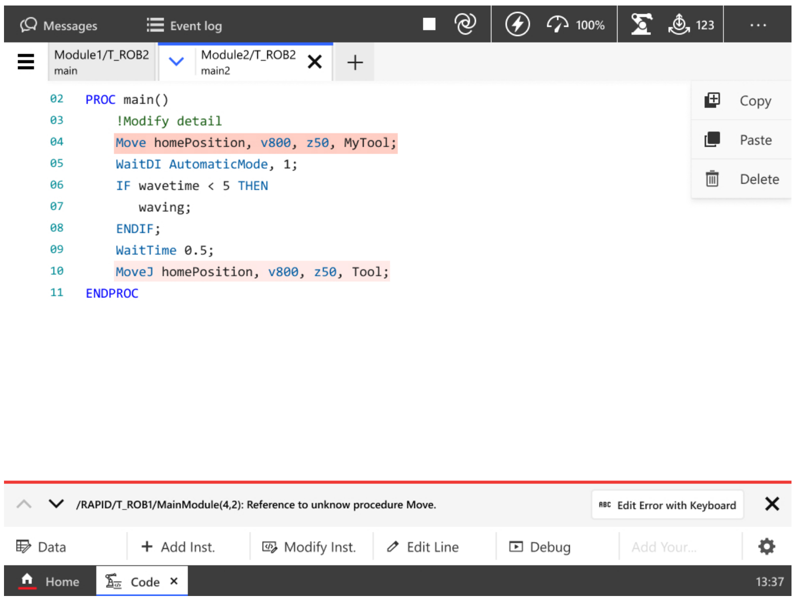


Figure 4.28: Solution 3 interface: add instruction panel

Issue 4: When checking a program to detect errors, only one error can be displayed at a time in the notification bar. As a result, users need to check multiple times to ensure there are no remaining errors. In addition, the error location was presented by a two-dimensional array, making it difficult for users to locate and address the issue.
Solution 4: The new solution highlights all errors in orange, enabling users to identify the location of all errors at once. Furthermore, users can tap on a button in the notification bar to directly open the edit page with a keyboard. After each modification, the program automatically checks for errors again.

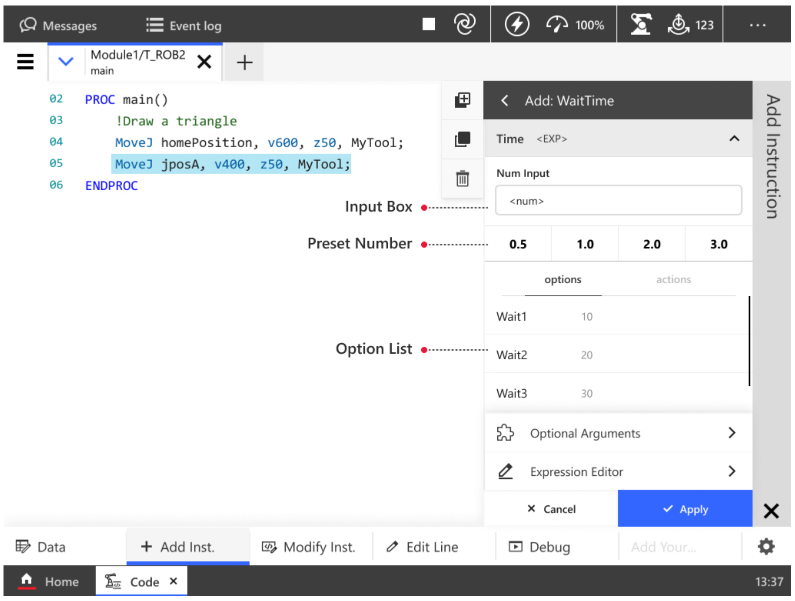


```
PROC main()
    !Modify detail
    Move homePosition, v800, z50, MyTool;
    WaitDI AutomaticMode, 1;
    IF wavetime < 5 THEN
       waving;
    ENDIF;
    WaitTime 0.5;
    MoveJ homePosition, v800, z50, Tool;
ENDPROC
```

/RAPID/T_ROB1/MainModule(4,2): Reference to unknow procedure Move.

Figure 4.29: Solution 4 interface: check program

Issue 5: While adding WaitTime instruction, the data options list only displays the name of the number data, making it inconvenient to identify which number is assigned to it. If users want to input the number, the number keyboard is located deep inside the menu.
Solution 5: In the new WaitTime property panel, the number input box is prominently displayed. To input the number, users can tap on the input box to trigger the number keyboard or tap on the predefined number buttons for the corresponding number to be quickly input instead of typing it manually. Underneath the input box, users can view the exact name and value of the number data, and select them by tapping on them.

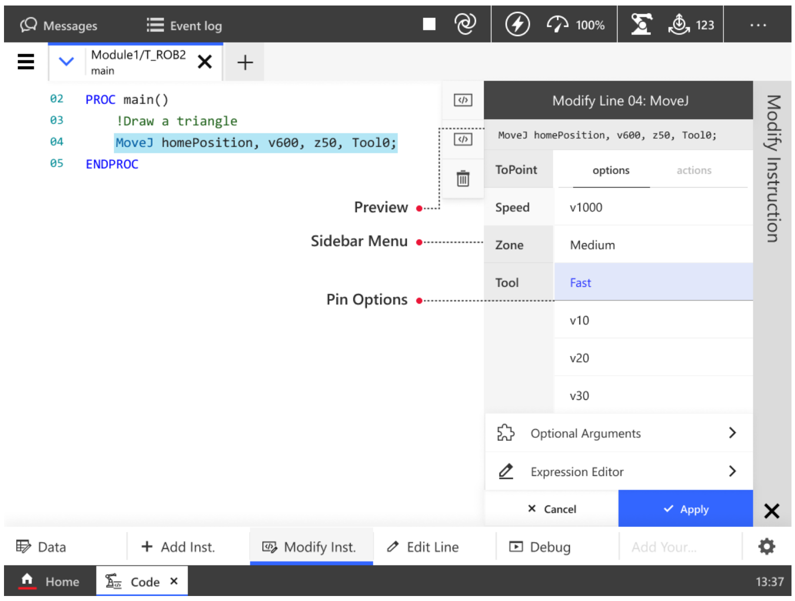


```
PROC main()
    !Draw a triangle
    MoveJ homePosition, v600, z50, MyTool;
    MoveJ jposA, v400, z50, MyTool;
ENDPROC
```

Figure 4.30: Solution 5 interface: add WaitTime instruction

Issue 6: When adding a movement instruction, the property panel displays various drop-down menus for selecting data. However, the drop-down menu is not suitable for touchscreens. The data is presented in alphabetical order, which makes it inconvenient to find the desired option when there are hundreds of them.

Solution 6: In the revised version, the sidebar menu replaces the drop-down menu. In this case, the titles of the parameters are always visible, and the option lists are fixed to prevent accidental tapping. There is a preview of the code at the top of the screen before the configuration is applied, which is handy. In addition, the ability to pin favourite options to the top of the options list makes it quick and easy for users to find and select their preferred options.



```
PROC main()
    !Draw a triangle
    MoveJ homePosition, v600, z50, Tool0;
ENDPROC
```

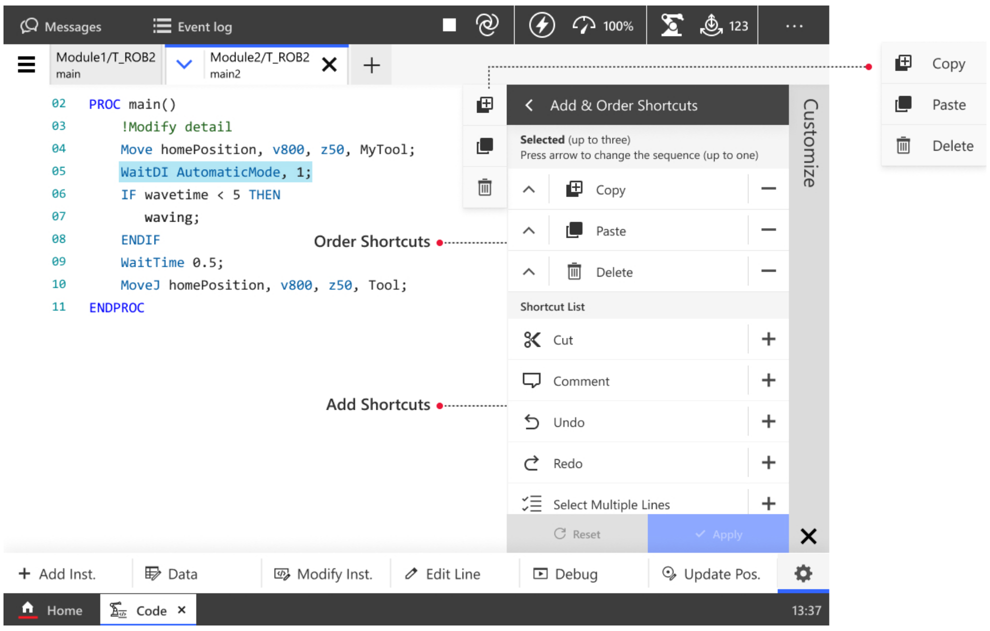
Modify Line 04: MoveJ

```
MoveJ homePosition, v600, z50, Tool0;
```

ToPoint	
Speed	v1000
Zone	Medium
Tool	Fast
	v10
	v20
	v30

Figure 4.31: Solution 6 interface: add Movement instruction

Issue 7: The clipboard functions are located on the 'Edit Line' page. To copy or cut the code, users must first open the 'Edit Line' panel, which also covers a part of the code.
Solution 7: Now, users can customize their shortcuts in the 'Customization' panel. The shortcuts will be prominently displayed at the top right of the code editor, and will feature large buttons and text labels for ease of use. When the right-side panel is displayed, only the shortcuts icon remains attached to its left side.

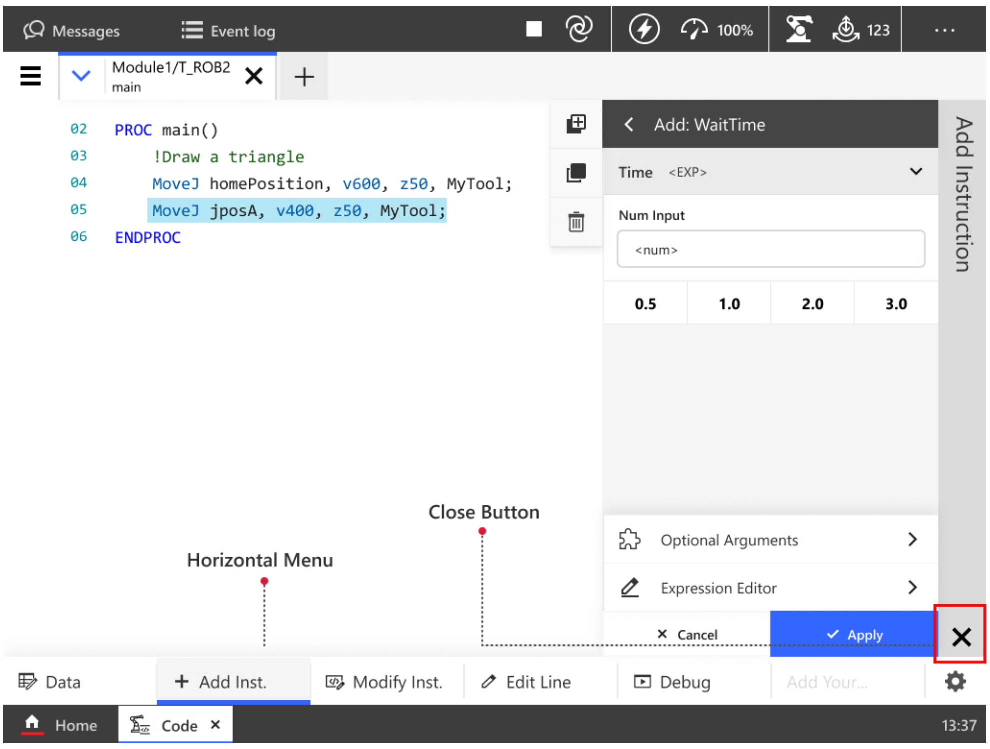


```
PROC main()
    !Modify detail
    Move homePosition, v800, z50, MyTool;
    WaitDI AutomaticMode, 1;
    IF wavetime < 5 THEN
       waving;
    ENDIF
    WaitTime 0.5;
    MoveJ homePosition, v800, z50, Tool;
ENDPROC
```

Selected (up to three)
Press arrow to change the sequence (up to one)

Figure 4.32: Solution 7 interface: customization shortcut

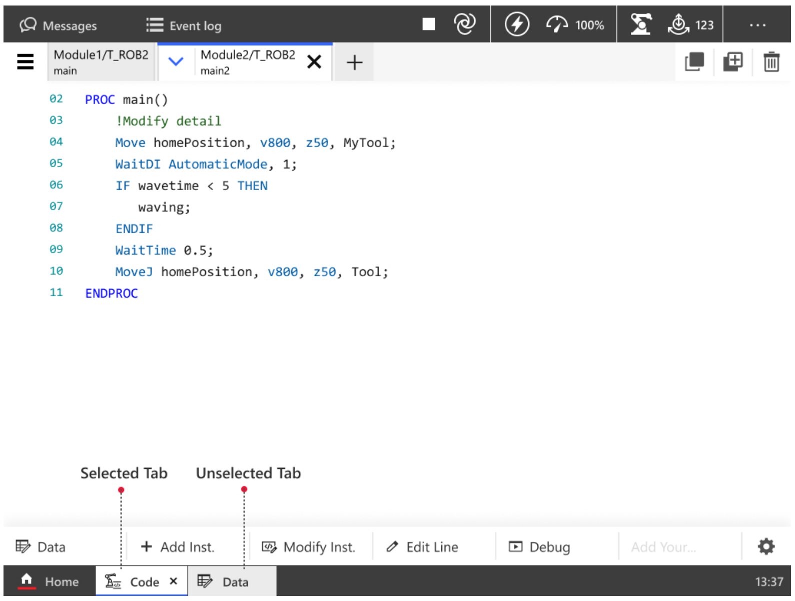
Issue 8: The right side panel unavoidably overlaps the right side of the code, but when users want to close the panel, they can only go back to the main page to find the close window button.
Solution 8: The main menu is modified from a vertical to a horizontal layout to facilitate quick switching of the operation panel by users. Additionally, a close button has been included at the bottom-right corner of each panel to enable users to quickly close the window.



```
PROC main()
    !Draw a triangle
    MoveJ homePosition, v600, z50, MyTool;
    MoveJ jposA, v400, z50, MyTool;
ENDPROC
```

Figure 4.33: Solution 8 interface: horizontal menu

Issue 9: Users have no clue how to close the application tab. Although users are aware of the need to long-press the tab to trigger the close button, the screen is not sensitive enough to promise the response.
Solution 9: The tab bar has been updated to be more like a browser's tab bar. The close button will be displayed only when the relevant application is opened. This enables users to intuitively and easily find the way to close applications.



```
PROC main()
    !Modify detail
    Move homePosition, v800, z50, MyTool;
    WaitDI AutomaticMode, 1;
    IF wavetime < 5 THEN
       waving;
    ENDIF
    WaitTime 0.5;
    MoveJ homePosition, v800, z50, Tool;
ENDPROC
```

Figure 4.34: Solution 9 interface: close tab button

Issue 10: When adding instructions with signal data, it was sometimes challenging to find the desired signal among hundreds of signals.
Solution 10: The new design displays two tabs in the signal settings area: one for favourite signals and another for displaying all signals. Users can now define their own favourite signal and select it under the first tab, or search for all signals by groups under the second type.

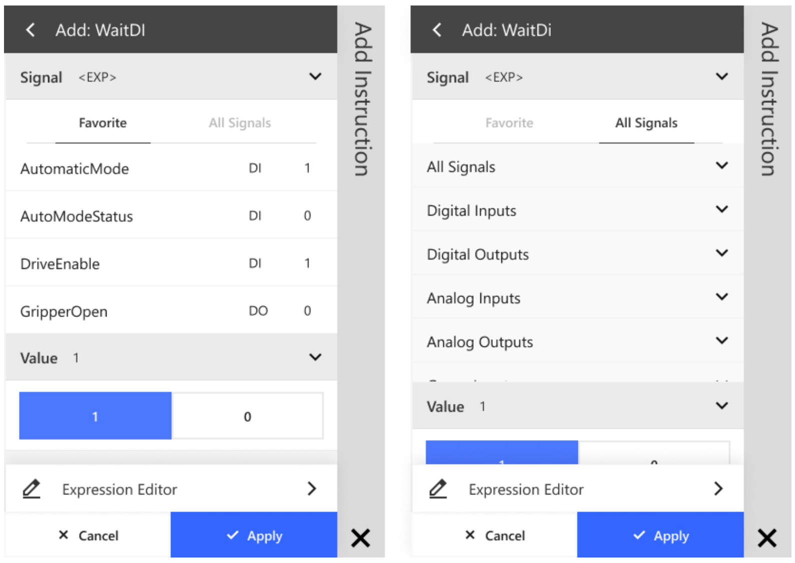


Figure 4.35: Solution 10 interface: signal option list

Chapter 5

Discussion

This chapter presents the reflection on the whole design process and final outcomes. The content includes the discussion about the debate among users and contribution to the industry and limitations within the process.

5.1 Customization VS Standardization

Indeed, accommodating the diverse preferences and habits of users is one of the challenges in user interface design. In the user study, participants expressed their expectations towards getting quick access to various functions and instructions. Therefore, in the solutions, customization is allowed in the menu tab bar, shortcuts, and data lists. However, as mentioned in the user testing feedback, different preferences emerged, such as prioritizing frequently used options versus favorite ones, and double-tapping or single-tapping to select code lines. They presented the expectation towards advanced customization function. The fact is mass customization has perverse effects, leading to chaotic, time-consuming and suboptimal workflow [49]. Thinking about the interface design from a big picture, what designer should allow the customization for what matters most instead of tailoring to specific user preference [50]. To find the right balance between customization and standardization, the designer should understand the context, the users, and the real need of target audience to design hybrid approach. It is also important to remember that it is impossible to cater to everyone's habits. A standardized workflow can lead to higher productivity. The balance in the final solution is to allow users the freedom to customize the things that have changed a lot along with the different projects, such as frequently used data options, instructions, and shortcuts. Things related to user habits, such as double-tapping or single-tapping behavior, are not allowed to be customized to avoid increasing complexity and decreasing productivity.

5.2 User Expectation VS Implementation Result

The user testing result indicates that the findings from the user study sometimes could not lead to the right answer, highlighting the cesssary of user testing to evaluate the hypothesis. In a field study, the participants showed strong expectations to know the exact value of each data option because he/she was a robotics engineer based on the position data to estimate the robot position. In the interview, several users also indicated the function 'view value' is hidden deep in the menu, making it not convenient to reach. Thus, their feedback was interpreted as viewing the value of each data quickly and easily should be put in a high priority. Then, in the solution, various ways of getting access to view the value data were provided, including a parameter detail bar, options list and data list. However, in user testing, participants indicated that they felt the value preview in the data list was confusing and not needed, especially in the position data. They usually remember the name of the data but do not read the value. What led to this wrong decision is that the problem was put in the wrong prioritization and provided too many solutions to solve one problem and more information and choice lead the system more complicated. From this case, there are two lessons learned. Firstly, user testing is quite important to evaluate whether the design meets users' expectations and fulfills the tasks' requirements [51]. In this project, user testing greatly helps find the solution's defect to refine it. The second thing is that whenever a user expresses a strong desire to have a feature, the designer should evaluate its prioritization carefully. Decision making should not be influenced by a specific user or user's emotions to avoid overestimating its urgency. When seeking solutions, the designer can consider the workaround to solve similar issues together and avoid introducing too many new features to make the system more complex. In summary, user expectation is not always the best solution; prudent evaluation and consideration are needed to define the real problem and level of urgency for finding a more suitable solution.

5.3 Reducing Familiarity Bias and User Resistance

Based on interview analysis, a part of users' dissatisfaction stems from the change that the interface of OmniCore FlexPendant differs significantly from IRC5 FlexPendant, which most of the participants have used for over 5 years. This situation can be attributed to both baby duck syndrome and familiarity bias that users tend to use what they are accustomed to and reject new changes [44][45]. Thus, when the interface of OmniCore FlexPendant significantly varies from the old version, the audience may describe the maladjustment as confusion or problem [44]. This situation interferes with the transition to a new system with resistance [44]. The primary causes of resistance reflected in this project are the decreased productivity, reliability issues and transition cost [52] [53] [54]. Therefore, if the learning cost of adopting the new system is high and it does not offer an improved user experience, it becomes a challenge for a company to promote the new system. At the same time, it influences users to make a rational decision about the transition.

In order to reduce familiarity bias and user resistance effect, one solution suggested by Peter Seebach is to offer users more compatibility by supporting old features in the new version [44]. For example, when Windows XP was launched, it allowed the user to continue using the interface structure of Windows 95/98's. Nevertheless, in this project, the OmniCore FlexPendant is not a personal laptop, but rather it is connected to the robot and not owned by a specific user. The hardware of IRC5 version and OmniCore version are different, which makes the interface cannot be compatible. Additionally, introducing compatibility will make the interface excessively personalized. Therefore, it is more reasonable to maintain the good design of the IRC5 version along with eliminating UX violation in the OmniCore version to allow for a smoother transition. As shown in the final solution, the vertical menu and fixed list in the IRC5 version have been reintroduced along with new features aimed at enhancing productivity. Additionally, interface redesign is not sufficient and supplementing with participatory and supportive approaches can be considered in future research [55]. For instance, in the early stage of design, involving users in the redesign process by hosting co-design workshops to encourage communication and information sharing can increase the acceptance of change [56]. Later, providing comprehensive training or onboarding features to help users become familiar with the new system is also an effective approach to relieve the effect of familiarity bias[57].

In conclusion, reducing user resistance cannot be achieved solely through redesign. Offering supportive training, encouraging user involvement and respecting user feedback are also beneficial in building a smooth transition, which points out the direction for future work [55].

5.4 Priciple of Designing Complex Application

Complex applications are defined as software used in a professional domain that differs from generalist applications for non-specialized users to finish casual tasks. These types of applications require complicated technological support, present various types of data and information, offer multiple solutions to accomplish tasks, are used in a specialized environment, and necessitate technical training [22]. In this case, UX designers always need to spend a lot of time conducting user research to have a comprehensive understanding of the context and problems before starting the design process. This project extracts five recommendations regarding interface design to guide the improvement of complex applications' user experience.

5.4.1 Differentiate the level and type of information by color-coding them.

A complex application displays various types of information on the interface. For identifying the type of information, coloring it is a good solution. For instance, in code editors, color-coding is used to enable programmers to quickly recognize different elements in the code, like instruction names in blue and data in black. In the case of messages with different levels of urgency, color indications such as red for a warning, blue for a notification, and green for a task completion are helpful. For example, as shown in Figure 4.29, when checking for syntax errors, the code with such errors is highlighted with a red background.

5.4.2 Clear clarity for each feature

Given the variety of functions in the complex application, it takes time for users to become familiar with them. Thus, each feature's button and label should present its function clearly to enable users to quickly identify its use. One reason for confusion is that the button contains only an icon without a text description. During user testing, users appreciated the clipboard shortcuts. However, they struggled to identify the meaning of the icons, particularly those of the copy and paste buttons. Therefore, the revised solution shown in the Figure 4.32 adds a shortcut button with descriptive text. Another reason is that the function's name is ambiguous. In the current design, users were confused by a feature called 'select range,' which is a feature that allows users to select multiple lines of code. 'Select multiples line' could be a good example to replace it with. Therefore, it is necessary to use a simple, general, and clear description to present this feature.

5.4.3 Flatten menu hierarchy levels

Complex applications often feature menus with multiple layers to accommodate options. However, multiple layers lead to clutter. This makes it harder for users to remember and find options. For instance, in the original design, when searching for instructions by groups, users had to navigate back and forth between several subpages to check every detail. Consequently, the solution (see Figure 4.28) implemented the accordion to reduce the hierarchy of group menus, which enables users to view all instructions under different groups on a single page. Moreover, menu hierarchy reduction led to fewer taps required.

5.4.4 Prioritize the frequently used functions

Although complex applications provide diverse functions to complete tasks, it is necessary to organize them into different hierarchies for high productivity. Identifying the frequently used functions of the target audience is crucial in reducing cumbersome interactions. For example, when adding a new instruction, users usually need to select the data for parameters from an option list. However, for the WaitTime instruction, it is more common for users to manually type the waiting time data. Therefore, the solution (Figure 4.30) shows the number input box in a prominent location instead of hiding it in the 'More Options' menu. In conclusion, based on the typical workflow to optimize the panel and menu structure helps reduce the effort to access the desired functionality.

5.4.5 Providing the ability to customize the system based on the individual and task

As previously stated, complex applications are not designed to solve general tasks; hence, the structure within the application should be flexible. Offering the function to customize the system's shortcut, menu, and options list promotes productivity. If a user must add an instruction more than ten times, it is worth spending a minute setting up a shortcut for that interaction. Customization is a way to enhance the efficiency of repetitive tasks.

5.5 Principle of Designing Touchscreen Application

The code editor for this subject is embedded in a touchscreen with a resolution of 1024 x 768. The interaction with the touchscreen is substantially different from that with a mouse and keyboard, and it comes with limitations in terms of screen size and sensitivity. Three design suggestions for constructing fluid interactions within a tablet application are provided.

5.5.1 Maximize the hotspot size and distinctiveness of the button.

According to research by the MIT Touch Lab, the average width of the index finger can cover 45 to 57 pixels on the touchscreen [58]. As the size is considerably larger than the cursor, if the button size is kept consistent with mouse-keyboard interaction, it may result in frequent accidental taps on other areas or difficulty in detecting the tap. Therefore, each button's size and hotspot should be at least 50px to allow for precise detection. Moreover, buttons with only one icon and without border and background should be avoided. Because touchscreens can not show the hover effect to denote the button's active status and meaning, which can lead to confusion.

5.5.2 Maximize the scrollable area.

In tablet interface design, scrolling interactions should generally be avoided [59]. However, for this project, where users need to quickly view all options within limited space, scrolling remains the best solution. If scrolling is necessary, designers should aim to maximize and fit the scrolling areas to avoid accidental tapping. In the current design, the scrolling dropdown menu always leads to the mistaken selection of options and the accidental closure of the menu. Therefore, the solution (Figure 4.31) includes replacing the drop menu with a fixed sidebar menu to provide a better scrolling experience while searching for data.

5.5.3 Avoid interaction requires higher sensitivity of touchscreen

When interacting through a mouse or keyboard, users can receive tactile and auditory feedback to confirm their actions. However, when users tap on the touchscreen, they can only rely on visual cues to identify their actions, resulting in lower accessibility. As a result, the detection of complex interaction on the touchscreen is difficult due to the need for high resolution and sensitivity. For instance, the current design requires users to long-press the application tab to activate the 'close application' button. However, it is challenging for users to differentiate whether they press or long press the button and whether they are tapping on the hotspot areas. To make the interaction more intuitive, as shown in the Figure 4.34, the long press actions have been removed in the solution. In conclusion, 'single tap' should be the primary method of interacting with elements in tablet interface design.

5.6 Research Contribution

The outcomes of this project contribute to the research on the interface for programming industrial robots. Current research aims to reduce the gap between novice and professional programming by introducing graphical programming solutions. Insufficient research has been undertaken to enhance user experience for professional programmers. The project deals with a complex application intended for use by experts. The results fill the gap in text-based and touch-based research on industrial robot programming solutions, present new solutions for project providers, and summarize the design principles for touchscreen and complex interfaces. The principles can be applied to designing applications for touchscreens with limited resolution and tactile detection. They can also inspire the design of complex applications in an industrial context.

Typically, programmers use a general code editor on their laptops to complete coding tasks and programming on a touchscreen is not an efficient approach worth considering. The touch-based code editor in the field of industrial robotics has the potential to make tablets a viable programming tool with high productivity. The touch-based code editor could become a powerful and convenient tool for providing flexible programming solutions in the future.

Text-based programming, a type of traditional programming method that allows advanced programming [60], still stands out as the most used and widespread coding method. Although it takes more time and effort to master than visual programming, its advantages should not be overlooked. This solution shows the potential to reduce the complexity of text-based programming interfaces, offering a more user-friendly experience for professionals and reducing the learning curve for novices. Generating code without the need for typing enables programmers to inspect the code details and enhance programming efficiency on a tablet. As a result, the research outcomes for industrial robot programming can be leveraged to enhance programming solutions in the future, including general touch-based code editors.

5.7 Research Methods Discussion

The design process presents both limitations and strengths in its methods. As a first step in the prestudy section, learning programming techniques as a user is a good method to become familiar with the challenges quickly. One of the challenges of designing a complex system is caused by its high domain exclusivity [22]. Many users who are experts in this domain have over 10 years of experience. To better understand the context and persuade them to accept a new solution, one must speak the same language as them. This informal research activity can lead to success in user studies. Competitive research also helps in understanding market trends and learning lessons from other products.

When it comes to interviews, it is good to collect a large amount of data, even though some limitations may occur. One of the biggest issues is the time consumption in dealing with the large amount of data. It takes a lot of time to go through all the data and categorize it. One reason is that programming robots in the teach pendant has to collaborate with various applications, leading to complicated issues. The participants' diverse backgrounds and experiences may lead them to view the same problem from different perspectives, increasing the difficulty of interpreting the data objectively. For future studies, a pre-research questionnaire can be considered to pre-screen participants and narrow down their focus on specific problems. During field studies, contextual inquiry can be a useful approach to uncover issues that were not remembered during the interviews. Due to confidentiality issues, all participants are ABB employees, but it may be worthwhile to consider inviting customers to participate in the research to gather more valuable insights. During the analysis phase, both semantic analysis and MoSCoW metric can benefit in providing results to build a comprehensive understanding of user experience and UX challenges.

In terms of the design process, since this is an individual project with a limited time, co-creation sessions were not conducted. As a result, solutions may be limited by individual perspectives. To compensate for this, UX expert review sessions were conducted with UX Designer colleagues to obtain professional assessments of the design. These design reviews were helpful in identifying and rectifying design elements that violated usability requirements. Utilizing realistic prototypes in user testing significantly contributes to the success of this project. Directly comparing the current and new design showcases the differences and provides an opportunity to emulate the actual workflow. The feedback from this user testing is reliable and assists significantly in the iterative design process. One minor limitation is that the prototype cannot control real or virtual robots and is not permitted to collaborate with other applications. As mentioned earlier, collaboration with various applications is inevitable in the real programming context; thus, it should be considered. Moreover, from an academic perspective, the user testing lacks the measurement of cognitive load, user resistance and perspective bias to prove that it can build a smooth transition and help reduce time pressure in the workplace. A deeper evaluation is worth conducting in future work to provide a more comprehensive understanding.

The ultimate solution significantly improved user experience and usability. However, this did not solve all the UX challenges. There are some debates and expectations in user testing regarding its design. Constrained by limited time, knowledge, and techniques, some sections are still waiting to be improved, such as the 'Expression Editor'. As the target audience is not homogeneous, balancing different users' preferences and requirements is a topic worth exploring in the future.

Chapter 6 Conclusion

The objective of this project is to investigate how the touchscreen text-based code editor in the robot controller can be redesigned to improve user experience and usability. The design thinking framework is applied to develop solutions and design principles that answer the research questions: How can the interface of the touchscreen text-based code editor in teach pendant be designed with regard to user experience and usability for robot programmers? During user research, participants expressed a strong desire to efficiently access desired features and options. Therefore, the solution focused on reducing redundant interactions and providing customization capabilities to improve productivity. By prioritizing UX issues, the main solutions are to increase the prominence of frequently used options and provide the ability to define shortcuts. Touchscreen accessibility is also improved by resizing components, using appropriate color contrasts, and providing clear clarity for functions. By incorporating feedback from users and UX colleagues, a final solution was created that results in less time spent and higher UX measurement scores. For the design of complex applications and touchscreen products, the principles can be summarized that user-friendly products should have a clear and simple hierarchy to present information and functions without confusion, and interactions should be easy to learn and easily recognized by the device. In addition, the solution provides a research reference to fill the gap in text-based code editor research in the field of industrial robots. It also provides inspiration for the design of the tablet code editor.

The project had gone through the process of design thinking. However, this process is not linear; designers can repeat the user research, testing, and iteration to achieve a better solution. Regarding limitations, some areas can be improved in future work. The first area is the composition of participants.

Most participants in the user study and testing are ABB employees, which means the data lacks customer feedback. Even though employees are also main users, they are more familiar with the ABB product portfolio and its functions than the customers, who have to use various products from different companies. Thus, it is worth gaining a different perspective from the customers. Secondly, although color coding is used to improve code readability and enable easy identification of different notifications, accessibility for users with visual impairment is not considered. For color-blind individuals, color cannot be identified. Additionally, the font size is not adjusted, making it challenging and time-consuming for those with poor eyesight to read the content. How to improve web content accessibility on tablets with limited resolution is a topic worthy of research. Last but not least, due to limited time, the user testing primarily focused on user experience and usability. Thus, further evaluation is recommended to obtain more reliable evidence to investigate to what extent the new system can reduce familiarity bias, alleviate time pressure, and enhance well-being in the workplace.

Considering implementation, the solutions should be evaluated for feasibility. The current solution proposes an entirely new structure and interaction, which is impossible to implement at once in the real development process. Moving forward, the designer should prioritize the identified problems once again and determine which ones can be immediately addressed within the current context. The solution must be modified to fit within the current design, similar to a workaround. Furthermore, this project is centered on the code editor, which is merely an application within the operating system as a whole. In the real workflow, the code editor must collaborate with other applications, particularly those involved in Data and I/O signal, which can complicate tasks. Thus, improving the collaboration experience could be considered in the future.

In terms of the general lesson learned, complex applications have no one-size-fits-all solution to address UX issues. The design direction is based on the current feedback to design a relatively better solution. Expert users in specialized domains are not eager to a fancy interface, but rather a tool that can help them complete tasks with higher productivity. In conclusion, this project successfully met its objectives through comprehensive user research activities and an iterative design process.

References

[1] "ABB's OmniCore™ robot controller nominated for prestigious IERA innovation award," 5 2019. [Online]. Available: https://new.abb.com/news/detail/23999/abbs-omnicore-robot-controller-nominated-for-prestigious-iera-innovation-award [Pages xi and 2.]

[2] Omnicore controller. [Online]. Available: https://new.abb.com/products/robotics/controllers/omnicore [Pages 1 and 10.]

[3] L. Gonçales, K. Farias, B. da Silva, and J. Fessler, "Measuring the cognitive load of software developers: A systematic mapping study," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 2019. doi: 10.1109/ICPC.2019.00018 pp. 42–52. [Page 5.]

[4] . Örün and Y. Akbulut, "Effect of multitasking, physical environment and electroencephalography use on cognitive load and retention," *Computers in Human Behavior*, vol. 92, pp. 216–229, 3 2019. doi: 10.1016/j.chb.2018.11.027 [Page 5.]

[5] E. Galy, M. Cariou, and C. Mélan, "What is the relationship between mental workload factors and cognitive load types?" *International journal of psychophysiology*, vol. 83, no. 3, pp. 269–275, 2012. doi: https://doi.org/10.1016/j.ijpsycho.2011.09.023 [Page 5.]

[6] R. W. Backs and K. A. Seljos, "Metabolic and cardiorespiratory measures of mental effort: the effects of level of difficulty in a working memory task," *International Journal of Psychophysiology*, vol. 16, no. 1, pp. 57–68, 2 1994. doi: 10.1016/0167-8760(94)90042-6 [Page 5.]

[7] C. M. Inzana, J. E. Driskell, E. Salas, and J. Johnston, "Effects of preparatory information on enhancing performance under stress." *Journal of Applied Psychology*, vol. 81, no. 4, pp. 429–435, 8 1996. doi: 10.1037/0021-9010.81.4.429 [Page 5.]

[8] "Working Conditions (Decent work for sustainable development (DW4SD) Resource Platform)." [Online]. Available: https://www.ilo.org/global/topics/dw4sd/themes/working-conditions/lang--en/index.htm [Page 5.]

[9] D. Fogli, L. Gargioni, G. Guida, and F. Tampalini, "A hybrid approach to user-oriented programming of collaborative robots," *Robotics and Computer-integrated Manufacturing*, vol. 73, p. 102234, 2 2022. doi: 10.1016/j.rcim.2021.102234 [Page 7.]

[10] S. Mitsi, K.-d. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris, "Off-line programming of an industrial robot for manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 9 2004. doi: 10.1007/s00170-003-1728-5 [Page 7.]

[11] D. Blank, D. Kumar, L. Meeden, and H. A. Yanco, "The Pyro toolkit for AI and robotics," *Ai Magazine*, vol. 27, no. 1, pp. 39–50, 3 2006. doi: 10.1609/aimag.v27i1.1862 [Page 7.]

[12] G. Biggs and B. MacDonald, "A Survey of Robot Programming Systems," *In Proceedings of the Australasian conference on robotics and automation*, vol. 1, pp. 1–3, 1 2010. [Page 8.]

[13] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. Shepherd, and D. Franklin, "Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices," *In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 4 2018. doi: 10.1145/3173574.3173940 [Pages 8 and 9.]

[14] B. Argall and A. Billard, "A survey of Tactile Human–Robot Interactions," *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1159–1176, 10 2010. doi: 10.1016/j.robot.2010.07.002 [Page 8.]

[15] K. Foit, "Visual interface for hybrid programming of the industrial robot," *Advanced Materials Research*, 10 2014. doi: 10.4028/www.scientific.net/amr.1036.743 [Page 9.]

[16] History of abb robots. [Online]. Available: https://robotsdoneright.com/Articles/history-of-abb-robots.html [Page 9.]

[17] S. Dietrich, "Introduction to ABB Robot Programming Language," *Technical Articles*, 2 2022. [Online]. Available: https://control.com/

technical-articles/introduction-to-abb-robot-programming-language/ [Page 9.]

[18] Robotstudio suite. [Online]. Available: https://new.abb.com/products/robotics/robotstudio [Page 9.]

[19] Abb teach pendants. [Online]. Available: https://robotsdoneright.com/Robotic-Equipment/ABB-robot-parts/abb-teach-pendants.html [Pages 9 and 10.]

[20] T. Brown and B. M. Katz, *Change by design : how design thinking transforms organizations and inspires innovation*, 1 2009. [Online]. Available: http://biblioteca.iednetwork.com/files/2011/06/change-by-design.pdf [Page 11.]

[21] F. P. Brooks, "The design of design: essays from a computer scientist," *Choice Reviews Online*, vol. 48, no. 04, pp. 48–2104, 12 2010. doi: 10.5860/choice.48-2104 [Page 11.]

[22] K. Kaplan, "Application complexity: a 5-Layer framework," 8 2020. [Online]. Available: https://www.nngroup.com/articles/complex-application-design-framework/ [Pages 11, 68, and 72.]

[23] R. Razzouk and V. J. Shute, "What is design thinking and why is it important?" *Review of Educational Research*, vol. 82, no. 3, pp. 330–348, 9 2012. doi: 10.3102/0034654312457429 [Page 11.]

[24] R. Balcaitis, "Design Thinking models. Stanford d.school," *Empathize IT*, 6 2019. [Online]. Available: https://empathizeit.com/design-thinking-models-stanford-d-school/ [Page 11.]

[25] J. DaSilva, "A Guide to Competitive Analysis for UX Design - Bootcamp," 5 2023. [Online]. Available: https://bootcamp.uxdesign.cc/a-guide-to-competitive-analysis-for-ux-design-1ddafeb9a3e7 [Pages 12 and 13.]

[26] R. Lee, "When, why, and how to conduct competitive analysis for UX research," 11 2022. [Online]. Available: https://www.userinterviews.com/blog/competitive-analysis-ux-research-vs-market-research [Page 13.]

[27] K. Baxter, C. Courage, and K. Caine, *Understanding Your Users: A Practical Guide to User Research Methods*, 5 2015. [Page 13.]

[28] H. Beyer and K. Holtzblatt, *Contextual Design: Defining Customer-Centered systems*, 7 1997. [Online]. Available: http://ci.nii.ac.jp/ncid/BA34227099 [Page 13.]

[29] J. Caulfield, "How to do thematic analysis | Step-by-Step Guide 038; Examples," *Scribbr*, 6 2023. [Online]. Available: https://www.scribbr.com/methodology/thematic-analysis/ [Page 13.]

[30] D. Clegg and R. Barker, *CASE Method fast-track.* Addison Wesley Longman, 1 1994. [Page 14.]

[31] "The moscow prioritization method explained." [Online]. Available: https://monday.com/blog/project-management/moscow-prioritizatio n-method/ [Page 14.]

[32] R. F. Dam and T. Y. Siang, "5 Common Low-Fidelity prototypes and their best practices," 3 2023. [Online]. Available: https://www.interaction-design.org/literature/article/prototyping-learn-eight -common-methods-and-best-practices [Page 14.]

[33] K. Pernice, "UX Prototypes: Low Fidelity vs. High Fidelity," 12 2016. [Online]. Available: https://www.nngroup.com/articles/ux-prototype-h i-lo-fidelity/ [Page 14.]

[34] A. Harley, "UX Expert Reviews," 2 2018. [Online]. Available: https://www.nngroup.com/articles/ux-expert-reviews/ [Page 15.]

[35] G. Charness, U. Gneezy, and M. A. Kuhn, "Experimental methods: Between-subject and within-subject design," *Journal of Economic Behavior and Organization*, vol. 81, no. 1, pp. 1–8, 1 2012. doi: 10.1016/j.jebo.2011.08.009 [Page 15.]

[36] P. Bhandari, "Within-Subjects Design | Explanation, Approaches, Examples," *Scribbr*, 6 2023. [Online]. Available: https://www.scribbr.com/methodology/within-subjects-design/ [Page 15.]

[37] B. Laugwitz, T. Held, and M. Schrepp, *Construction and evaluation of a user experience questionnaire*, 1 2008. [Page 15.]

[38] M. Schrepp, "User experience questionnaire handbook version 10," 05 2023. doi: 10.13140/RG.2.1.2815.0245 [Page 15.]

[39] J. Brooke, "Sus: A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, 11 1995. [Pages 15 and 47.]

[40] "ABB's OmniCore™ robot controller nominated for prestigious IERA innovation award," 5 2019. [Online]. Available: https://new.abb.com/news/detail/23999/abbs-omnicore-robot-controller-nominated-for-prestigious-iera-innovation-award [Pages xi and 19.]

[41] "RobotStudio Desktop." [Online]. Available: https://new.abb.com/products/robotics/robotstudio/robotstudio-desktop [Pages xi and 19.]

[42] "ABB industrial robots get Wizard Easy Programming software," 12 2020. [Online]. Available: https://new.abb.com/news/detail/72178/abb-industrial-robots-get-wizard-easy-programming-software [Pages xi and 20.]

[43] K. Salazar, "Contextual inquiry: Inspire design by observing and interviewing users in their context," 12 2020. [Online]. Available: https://www.nngroup.com/articles/contextual-inquiry/ [Pages ix, 26, and 91.]

[44] P. Seebach, "The cranky user: Baby duck syndrome – Imprinting on your first system makes change a very hard thing," 3 2005. [Online]. Available: https://web.archive.org/web/20120419150252/http://www.ibm.com/developerworks/web/library/wa-cranky50/index.html [Pages 29 and 67.]

[45] A. Tversky and D. Kahneman, "Judgment under Uncertainty: Heuristics and Biases," *Science*, vol. 185, no. 4157, pp. 1124–1131, 9 1974. doi: 10.1126/science.185.4157.1124 [Pages 29 and 67.]

[46] M. Gearon, "Level up — How to design better buttons - Michael Gearon - Medium," 10 2019. [Online]. Available: https://michaelgearon.medium.com/level-up-designing-better-buttons-6c0e2571b1d1 [Page 34.]

[47] I. Apple Computer, *Macintosh Human Interface Guidelines*. USA: Addison-Wesley Publishing Company, 1992. ISBN 0201622165 [Page 35.]

[48] J. Sauro, PhD, "5 Ways to Interpret a SUS Score," 9 2018. [Online]. Available: https://measuringu.com/interpret-sus-score/ [Page 48.]

[49] C. A. Sinsky, H. Bavafa, R. G. Roberts, and J. W. Beasley, "Standardization vs Customization: Finding the Right Balance," *Annals of Family Medicine*, vol. 19, no. 2, pp. 171–177, 3 2021. doi: 10.1370/afm.2654 [Page 65.]

[50] J. H. Gilmore and B. J. Pine, “The four faces of mass customization.” *PubMed*, vol. 75, no. 1, pp. 91–101, 12 1996. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/10174455 [Page 65.]

[51] J. Nielsen and J. Levy, “Measuring usability: Preference vs. performance,” *Commun. ACM*, vol. 37, no. 4, p. 66–75, apr 1994. doi: 10.1145/175276.175282 [Page 66.]

[52] M. N. Ngafeeson and V. Midha, “An exploratory study of user resistance in healthcare IT,” *International Journal of Electronic Finance*, vol. 8, no. 1, p. 74, 1 2014. doi: 10.1504/ijef.2014.064003. [Online]. Available: https://doi.org/10.1504/ijef.2014.064003 [Page 67.]

[53] R. Hirschheim and M. Newman, “Information Systems and user Resistance: Theory and practice,” *The Computer Journal*, vol. 31, no. 5, pp. 398–408, 5 1988. doi: 10.1093/comjnl/31.5.398 [Page 67.]

[54] H.-W. Kim, “The effects of switching costs on user resistance to enterprise systems implementation,” *IEEE Transactions on Engineering Management*, vol. 58, no. 3, pp. 471–482, 8 2011. doi: 10.1109/tem.2010.2089630 [Page 67.]

[55] M. Ali, L. Zhou, L. C. Miller, and P. Ieromonachou, “User resistance in IT: A literature review,” *International Journal of Information Management*, vol. 36, no. 1, pp. 35–43, 2 2016. doi: 10.1016/j.ijinfomgt.2015.09.007 [Pages 67 and 68.]

[56] D. Waddell and A. S. Sohal, “Resistance: a constructive tool for change management,” *Management Decision*, vol. 36, no. 8, pp. 543–548, 10 1998. doi: 10.1108/00251749810232628 [Page 67.]

[57] V. Chang and G. Wills, *A University of Greenwich case study of cloud computing*, 1 2013. [Page 67.]

[58] K. Dandekar, B. Raju, and M. Srinivasan, “3-d finite-element models of human and monkey fingertips to investigate the mechanics of tactile sense,” *Journal of biomechanical engineering*, vol. 125, pp. 682–91, 11 2003. doi: 10.1115/1.1613673 [Page 70.]

[59] C. Tong, “How to design for tablet interfaces - Crystal Tong - medium,” 8 2018. [Online]. Available: https://medium.com/@crystaltong618/tablet-user-interface-guideline-ac8216c7a3f6 [Page 70.]

[60] “Visual Programming vs Traditional Programming: Full Guide,” *nandbox*, 6 2023. [Online]. Available: https://nandbox.com/visual-programming-vs-traditional-programming-full-guide/ [Page 72.]

Appendix A

Interview materials

A.1 Interview Structure

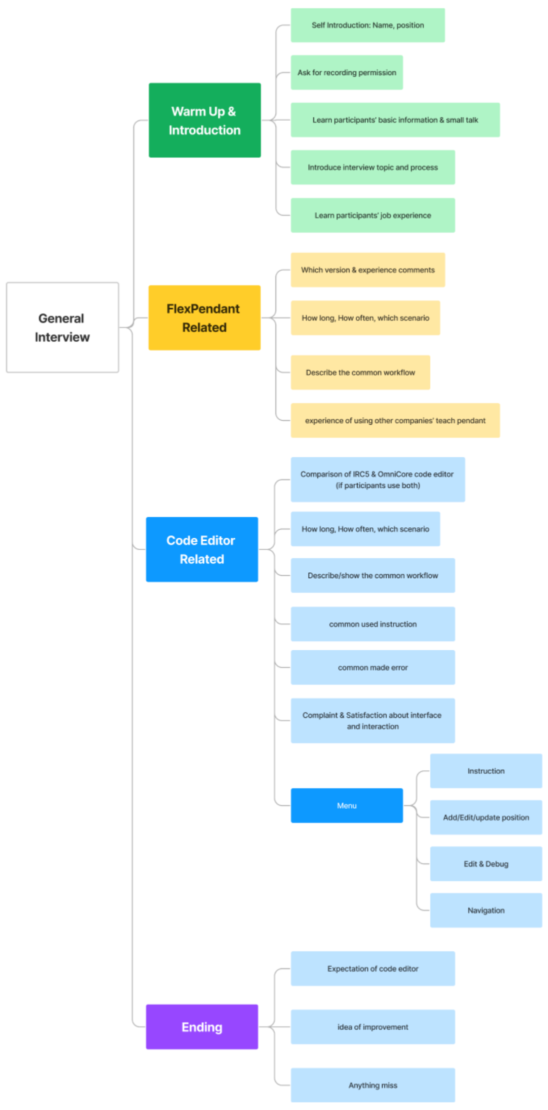

General Interview
Warm Up & Introduction
Self Introduction: Name, position
Ask for recording permission
Learn participants' basic information & small talk
Introduce interview topic and process
Learn participants' job experience
FlexPendant Related
Which version & experience comments
How long, How often, which scenario
Describe the common workflow
experience of using other companies' teach pendant
Code Editor Related
Comparison of IRC5 & OmniCore code editor (if participants use both)
How long, How often, which scenario
Describe/show the common workflow
common used instruction
common made error
Complaint & Satisfaction about interface and interaction
Menu
Instruction
Add/Edit/update position
Edit & Debug
Navigation
Ending
Expectation of code editor
idea of improvement
Anything miss

Figure A.1: Interview Structure

A.2 Interview Question Sheet

Warm up and introduction

- hello, [participant name] I am Xuanling (what can I call you?) , and currently doing my master thesis in ABB Robotics. Thank you very much for taking the time to speak with us today. I am working on the project to redesign the code editor in FlexPendant, so I am appreciated if you could tell me about your user experience with code editor. This interview aims at understanding your workflow, the problems you've encountered, and areas that can be improved. Does that sound good to you?
- What's more. The participation in this interview is entirely voluntary. We'll treat your information, the interview recording, and transcripts with confidentiality, and only the research team will have access to them. I am going to record this interview, is it okay with you?
- Could you tell me about your background and previous experience?
- Could you tell me a bit about the work your do at "company"?
  - What are the main responsibilities as a [work role]?
  - What kind of customers do you work for? (for integrator)

FlexPendant Related

- How long have you been working with FlexPendant? Which version are you using (IRC5 or OmniCore)? [choose the questions list based on which version of FlexPendant they used]
- Could you describe your user experience and how you feel about using FlexPendant?
- How frequently do you use FlexPendant in your work?
- What is the main tasks for you to use the FlexPendant.
  - In which scenario you have to use FlexPendant to operate the robots
  - Could you tell me about the process when you do [tasks mentioned previously]?

- Have you used other companies' teach pendants before?
  - If so, which one did you prefer and why? Can you describe the reasons for your preference?

Code Editor Related

- How long have you been working on Robot Programming?
- How often do you work with code editor in FlexPendant?
- Do you also use RobotStudio for robot programming?
  - How about other ABB internal product? such as Wizard Programming or Robot Control Mate to program robot?
- Do you also use other companies product to program robot now or before? Could you show me some examples?
  - As a robot programmer, which programming method do you prefer more? Graphic or Text-based? Which companies? Could you explain why?
- Could you describe your user experience and your feelings about using the code editor in FlexPendant?
  - Could you compare the difference of two code editor? (if interviewee used two)
  - Whose interface and interaction more intuitive, user friendly, easy to understand and learn? Could you explain why?
- What do you usually use the code editor for? such as change position
  - Does the needed information appropriately presented? such as program name
  - Does the the code editor help to quickly and efficiently finish the work?
- Could you tell us what do you use the menu panel for? (share screen to interviewee)
  - Which functions in the menu you use frequently?
  - Does it work well? What works poorly?

    - Does the menu structure clear? Could you quickly access the functions you need?

- Could you tell us which methods/workflow you usually used to update the position data?

- Could you tell us your user experience about typing number or text using the keyboard in FlexPendant?

- What are the main tasks that you typically perform using code editor?
    - Could you tell me about the process when you do [tasks mentioned previously]?
    - Would you be willing to share your screen with us to show us your workflow?
    - What are the most frequently used instructions that you use in the code editor?
    - What errors do you often make when using the code editor regarding the interaction and RAPID programming?
        - ∗ How do you resolve the error?

- Do you have opportunity to collaborate with others to program robot using code editor? If yes, could you tell me more about the workflow of reusing the code?

- What are you/your customers (for integrator) most want to complain when using the code editor? / What do you dislike the most about using the code editor?
    - Could you tell me more about that? Explain the reason.
    - Do you have any suggestions on how to improve it?

- What do you like the most and least about using the code editor?

Wrap up

- What are your expectations regarding how the code editor should be like within FlexPendant?

- Do you have any suggestions on how we can improve the code editor in FlexPendant?

- Is there anything that we have missed to ask that you think is relevant to code editor user experience?
- Thanks a lot for your time today. If you have any further questions or if you would like to learn more about my project, please do not hesitate to contact me.

Appendix B

Contextual Inquiry materials

B.1 Contextual inquiry procedure inspired by [43]

Preparation

- information sheet and consent form
- Notebook: jot down the discussion as soon as possible
- Phone: recording audio
- Go Pro: recording the video that how they interact with FlexPendant

Primer

- Hello, [Participant Name]. Nice to see you! My name is Xuanling, a thesis worker in Robotics UX team. Thank you very much for taking the time to join my research activity today. The project I am working on is to redesign the code editor in OmniCore FlexPendant. I will mainly focus on the User Experience and Interface of using the code editor in real work place, so your ideas must be important.
- In this field study, I hope you can show me how operate and program the robot using OmniCore FlexPendant during your daily work. It will be great if you can show me two or three tasks like a craftsman teach a student a skill through doing. After this activity, I hope I can understand the role of FlexPendant in your daily work, how you will use it for and the problems you encountered. If I misunderstand your purpose of specific

action, please correct my misinterpretations. It is quite important for me to understand your each decision and action.

- What's more. The participation in this interview is entirely voluntary. We'll treat your information, the recording, and transcripts with confidentiality, and only the research team will have access to them. I am going to audio record our conversation, is it okay with you?
- I am more interested how the code editor is involved in your daily work. So, before showing me your work flow, could you describe how do you use code editor during your work or in which scenario you need to use code editor in the FlexPendant?

The Transition

- Thanks for your explanation. Also, I would like to let you know, when you are performing the tasks, I will observe you behavior. When I find something interesting to discuss or something make me confuse, I will interrupt you. If it is not a good timing for interruption, please let me know. We will continue the discussion until a better stopping point.
- I am interested in the fine details of your work, it will be better you complete your tasks as normal, without I am here

The Contextual Interview
Question:

- Could you elaborate more on what you mean by A?
- I am not quite sure I understood the purpose of that step. Can you explain it to me again in simpler terms
- I noticed you used a particular setting. Can you tell me more about why you chose that specific setting
- Can you show me an example of what you mean when you say ....?
- When you said you needed to X the robot, what exactly did you mean by that
- I am not familiar with that term, can you explain what X means?
- Can you explain the purpose or function of this button/interface for a given tasks?

- Can you walk me through your thought process when you were programming the robot for this task?
- when you made that adjustment to the robot's program, what was your reasoning behind that decision?
- When you encounter X problem while programming the robot, how do you determine the root cause of the issue?
- What is the standard steps to do this? any other option?

Confirm:

- I noticed that you did X, Could you tell me more about why you did that?
- Just to confirm, when you tap on X button, it does Y function, right?
- If I a, understanding correctly, you did X to achieve Y, is that correct?
- Can you show me again how you did X? I want to make sure I understand the steps.
- Did you mean that you need to do X before Y, or was it the other way around?
- To clarify, you are saying that if we do X, it will cause Y to happen. Is that correct?

The Wrap-up

- The teach pendant is mainly used for ....
- Your typical workflow to finish is task is.... begin by... then....
- You must be ensure ... to do ....
- I understand .....
- Is there anything that I misunderstood/missing?
- Thanks for your time!

Appendix C

User testing materials

C.1 User Testing Procedure

Introduction (5 min)

- Introduce Usability Test Procedure (assist first two tasks, and finish the final one by themselves)
- Introduce Two code editors
- Determine the test sequence (current or new)

Exploration (5 min)

- User read the user manuals
- Explore the code editor through ‘playground’ module
- Customize your favorite/frequent used menu button/shortcut

Task 1: Familiarize how to use (5+5 min)

- Add Movement and WaitTime instructions to draw a triangle in ‘main/Module1/T_ROB1’.

Task 2: Fine Tuning - Waving hand (7.5+7.5 min)

- Modify Routine detail in ‘main2/Module2/T_ROB1’ and ‘waving/Module2/T_ROB1’
- Position parameter - Update Position

- Position parameter - View Value: change value
- WaitDI - change signal and value
- Check program - edit selection with keyboard
- IF <EXP> - Edit Expression with Keyboard
- Procall - View Value - Open Routine
- Zone parameter - switch to
- For loop - Edit Value

Task 3: A simple pick up workobject program (10+10 min)

- Finish the 'main/MainModule/T_ROB2' and 'PickAndDrop/MainModule/T_ROB2'
- Move to homeposition and wait one second
- Call routine PickAndDrop
- Move to position 1 (above the workobject) and then open gripper
- Wait 1 second and then move to position 2 (workobject location) and close gripper
- wait 1 second and then Move to position 3 (destination)
- wait 0.5 second and then Open gripper
- wait 0.5 second and then move to position 4 (above destination)
- go back mainroutine and select first two lines and copy and paste

Questionnaire (5 min)

- Finish 2 questionnaire, One for current code editor, another for new design

Short Interview (5 min)

- Short interview for getting user feedback