Thesis no: BCS-2014-01

Evaluation of HMI Development

for Embedded System Control

Linda Andersson

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science. The thesis is equivalent to 10 weeks of full time studies.

Contact Information:
Author(s):
Linda Andersson
E-mail: liaj11@student.bth.se

External advisor:
Anders Leopold
Team Manager R&D,
Yaskawa Nordic AB

University advisor:
Dr. Mikael Svahnberg
Dept. of System & Software Engineering,
Blekinge Institute of Technology

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet	:	www.bth.se
Phone	:	+46 455 38 50 00
Fax	:	+46 455 38 50 57

Abstract

Context:The interface development is increasing in complexity and applications with a lot of functionalities that are reliable, understandable and easy to use have to be developed. To be able to compete, the time-to-market has to be short and cost effective. The development process is important and there are a lot of aspects that can be improved. The needs of the development and the knowledge among the developers are key factors. Here code reuse, standardization and the usability of the development tool plays an important role which could have a lot of positive impact on the development process and the quality of the final product.
Objectives: A framework for describing important properties for HMI development tools is presented. A representative collection of two development tools are selected, described and based on the experiences from the case study its applicability is mapped to the evaluation framework.
Methods: Interviews were made with HMI developers to get information from the field. Following that, a case study of two different development tools were made to highlight the pros and cons of each tool.
Results: The properties presented in the evaluation framework are that the toolkit should be open for multiple platforms, accessible for the developer, it should support custom templates, require non-extensive coding knowledge and be reusable. The evaluated frameworks shows that it is hard to meet all the demands.
Conclusions: To find a well suited development toolkit is not an easy task. The choice should be made depending on the needs of the HMI applications and the available development resources.

Keywords: Software, Reuse, HMI, Standardisation, Interface development.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The development cycle of software applications is getting shorter and the pressure to deliver user friendly interfaces for products are increasing. The user experience is an important factor to compete with other similar products. There are a wide variety of programmable units with interfaces connected to robotics in the industry. These are key components for providing important information to the user. They can display information about the current process, work history, safety, maintenance and error reporting [1]. At Yaskawa Nordic AB a lot of time has to be spent and much code has to be written to make a human-machine-interface (HMI) and this process could be improved.

Figure 1.1: Welding robot from Yaskawa

HMI development should be focused on designing a user friendly product, not on advanced coding. The HMI designer should not have to worry about the target platform or the network connection to the application when he or she makes the interface. Nowadays at Yaskawa, this is not the case and every interface has to be developed specifically for the project. It requires advanced programming knowledge which is a big drawback that narrows down the number of people qualified to develop a HMI. Because every project is a bespoke solution, it is not very easy to reuse solutions from previous projects. It is also hard to keep a unified style, in regards to the interface design and appearance of the application and the structure of the source code, especially considering that the HMI applications are made in multiple different development tools.

The report is primarily aimed at software developers that are going to work in a multi platform environment. It provides information about which tools are best to use for the interface design, what factors are important and how to improve the development process. In this study we investigate the current state of HMI development at Yaskawa. The reason why the study is performed on Yaskawa is because they were the ones who offered the exam job. We did not take contact with any other companies in the same field.

1.1 Aim

By investigating the state of HMI development on a company, factors which are important for effective HMI development can be identified and result in a table with criteria for evaluation of HMI development tools. The criteria presented in the table will be the base of the evaluation when performing a case study on two selected development tools.

The results may provide important information to companies who are in the situation where they have choose a HMI development tool.

1.2 Research Questions

The study investigates the following research questions:

- RQ1: How does the current process of HMI development for industrial robotics look at Yaskawa Nordic AB?
    - RQ1.1: In what way can the HMI development process be improved to shorten development time and required programming knowledge of the HMI developer?
- RQ2: What is the applicability of the selected HMI framework to the current HMI needs?
    - RQ2.1: What are the experiences from applying an HMI framework matching the current HMI needs?

- RQ2.2: What are the pros and cons of the selected HMI framework applied to the current needs?
- RQ2.3: What improvements can be made to the HMI framework ?

The first question is going to answer the first part of the aim, to identify the current state of the development process and identify factors which can be important for a HMI development tool to fulfil.

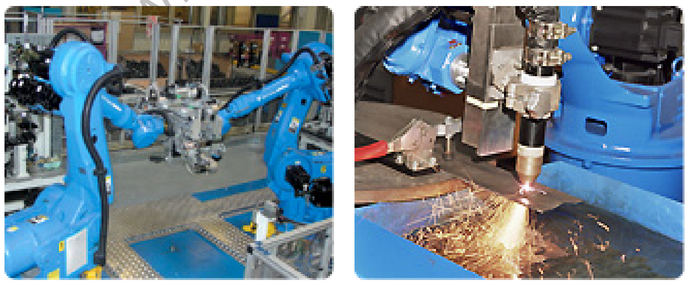
The second question is answered by a case study of the selected HMI development tools. The resulta are analysed and evaluated to see how well they meet the needs.

1.2.1 Limitations

The choice in interviewees will be restricted to involve people with the knowledge and experience of the evaluation area. Only a few of the HMI development toolkits available will be evaluated.The examination of the toolkits will focus on tasks that are important for developing robot interfaces.

1.3 Background

The robots keep the industrialisation moving forward. They automate and speed up the workflow in the industry. Yaskawa is one of the leading companies in industrial robotics. Other big companies in the same field are ABB Robotics, Fanuc Robotics and Kuka Robotics. A robot and the collection of machinery surrounding it is referred to as a "workcell" or "cell". At Yaskawa, most of the projects are custom made robot work cells that are produced for a specific customer and task. The tasks vary a lot and the cells can be composed of a number of different robots. Robots are used for tasks like welding, packaging and assembling.



(a) Assembling (b) Robot used for cutting

Figure 1.2: Robots are used for a lot of different tasks in the industry, here are two examples.

The HMI is an important part of the robot which provides vital information to the user. There are many platforms and combinations of platforms used to display the HMI. Each platform has its own unique possibilities and functionalities. The one closest to the machine is the teach pendant.

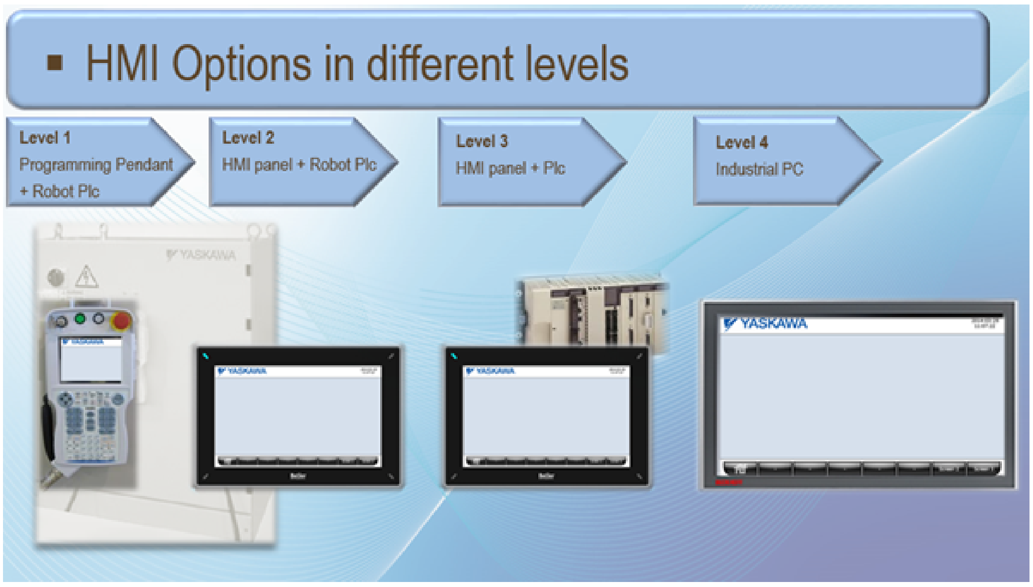


Figure 1.3: Different platforms for HMI presentation, the first alternative is a programming pendant, or teach box. The two following ones are HMI operation panels and the last one is a regular screen connected to a PC

The teach pendant, also called teach box or programming pendant (see 1.3), is a hand held device used by the operator to navigate the robot through a series of points that describe the desired movement path. The controller records these points and saves them as a job[2]. A job is a list of instructions that the robot will perform in a cyclic manner. The main usage of the teach pendant is to provide an interface for programming these jobs, but it is also used to get a detailed status from the robot. Some robot cells only have the teach box and some control buttons as the entire interface.

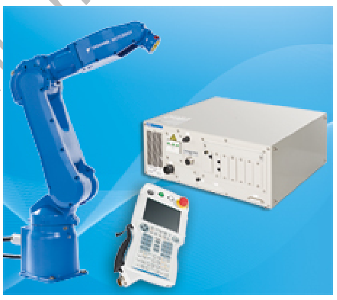
A Programmable Logic Controller (PLC) is one of the main components of automation. They are specialized to run reliably for years, even in rough industrial environments. They should be able to withstand the shock, vibrations, heat and electrical noise that occur on the manufacturing floor. PLCs work in a scan cycle. First, all the input devices are read on the PLC to get their status. With these values the program executes and produces results, which are then sent to the output devices. The programs for PLCs are written with ladder logic and they build on Boolean logic. The ladder logic looks similar to electric ladder diagrams, which engineers and electricians were already familiar with. This is one of the reasons that the PLCs became popular[3].

Today, PLCs are effectively used as cheaper solutions for automation, compared to PCs. The PLC stores instructions and functions such as logic, sequencing, timing, counting and arithmetic in order to control the machine or process[4].

More advanced programs can run on the PLC, compared to the teach box. The PLC is a more convenient target platform when developing larger robot cells with multiple machines surrounding the robot. It can be connected to a HMI panel which can have a bigger screen than the teach pendant, this makes it a more suitable for displaying overviews of report history and job lists. It is usually possible to provide the same functionality on either the teach pendant or the PLC and every robot cell has its own custom-made implementation. But the possibilities to customise the interface of the teach box is rather limited and a custom-made implementation is rarely done, at least not at Yaskawa.

If more computing power or memory is required, a PC can be used. This gives the opportunity to utilize software that is available on a computer. For example it is possible to connect the robot to a database. It is also possible to use a soft PLC which is a PLC running embedded on a PC. On the soft PLC it is easier to troubleshoot and upgrade the program since it can be updated from the PC it is running on, but a PC it is much more expensive than a normal PLC. For a normal PLC the program has to be modified on an external PC and then saved to the memory of the PLC[3].

(a) Yaskawa robots



(b) A robot, its teach box and control system

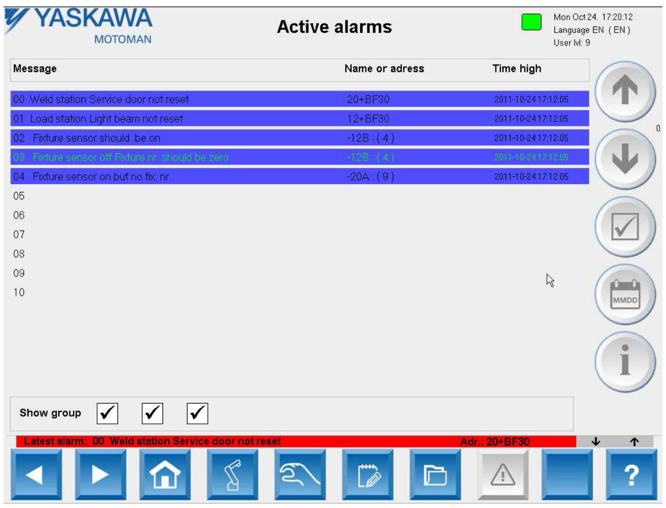
Figure 1.4: The robot is not alone on the manufacturing floor, it is surrounded by various control devices.

The controller is the heart of the system that gives commands to the mechanical parts of the robot. It generally consists of a microprocessor linked to I/O and monitor devices. By running jobs (programs) on the controller, instructions are executed which can activate actuators that move the robot [2].

1.3.1 Robot HMI

The graphical interface runs on an HMI operation panel, which is a rich client. This means that the graphical interface and its rendering, functions and interactions are processed on the HMI panel. The HMI screens are usually developed by PLC manufacturers, and their development tools are made for their screens and software.

Today, a lot of different fieldbus communication protocols are used in automation. For communication between the robot controller and the HMI, High Speed Ethernet (HSE) Server is usually used. The PLC can communicate with the HMI via an Object Linking and Embedding for Process Control (OPC) connection. OPC is a widely used standard in industrial automation built on the Microsoft Windows COM/DCOM. It specifies real-time communication between control devices from different manufacturers.


YASKAWA MOTOMAN

Active alarms

Mon Oct 24. 17:20:12
Language EN (EN)
User lvl: 9

Message	Name or adress	Time high
00 Weld station Service door not reset	20+BF30	2011-10-24 17:12:05
01 Load station Light beam not reset	12+BF30	2011-10-24 17:12:05
02 Fixture sensor should be on	-12B : (4)	2011-10-24 17:12:05
03 Fixture sensor off Fixture nr. should be zero	-12B : (4)	2011-10-24 17:12:05
04 Fixture sensor on but no fix. nr.	-20A : (9)	2011-10-24 17:12:05
05		
06		
07		
08		
09		
10		

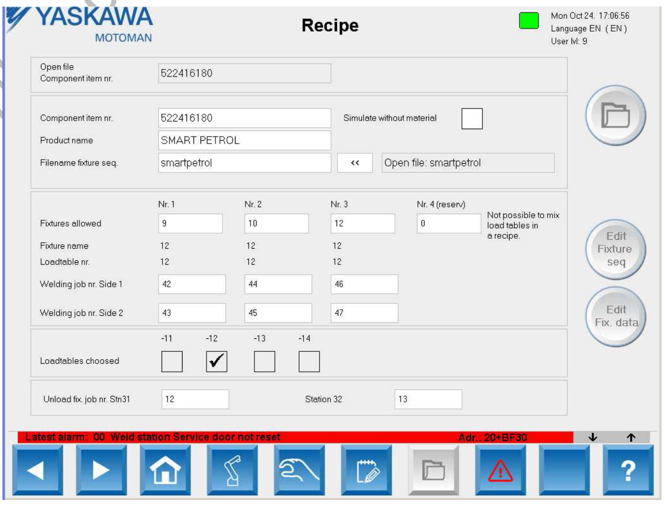
Show group ✓ ✓ ✓

Latest alarm: 00 Weld station Service door not reset Adr.: 20+BF30

Figure 1.5: The alarm page is very important. It provides vital information if and when something went wrong. This page is from a HMI developed with Bechoff's TwinCAT 2.

The HMI application runs on the operator panel or on a PC with a monitor. The task of the HMI applications is to provide an interface to the robot. This should be easy to use and learn for the operators, and it should provide important data about the robot cell.

Some of the common information and functionality provided by the HMI is the layout of the robot cell and its safety, the status of the robot and its input/output devices, current orders and recipes. It should also present alarm messages if something is wrong. Figure 1.5 and 1.6 are examples of a recipe modification screen and an alarm screen from a Yaskawa HMI application.


YASKAWA MOTOMAN

Recipe

Mon Oct 24. 17:06:56
Language EN (EN)
User lvl: 9

Open file Component item nr. 522416180

Component item nr. 522416180 Simulate without material ☐
Product name SMART PETROL
Filename fixture seq. smartpetrol << Open file: smartpetrol

	Nr. 1	Nr. 2	Nr. 3	Nr. 4 (reserv)	
Fixtures allowed	9	10	12	0	Not possible to mix load tables in a recipe.
Fixture name	12	12	12		
Loadtable nr.	12	12	12		
Welding job nr. Side 1	42	44	46		
Welding job nr. Side 2	43	45	47		

	-11	-12	-13	-14
Loadtables choosed	☐	✓	☐	☐

Unload fix. job nr. Stn31 12 Station 32 13

Latest alarm: 00 Weld station Service door not reset Adr.: 20+BF30

Figure 1.6: A recipe page is almost always present in a HMI application, it allows the operator to add and configure recipes. This is page is from the same application as Figure 1.5.

In summary, HMI can be displayed on a number of different platforms. The screen can vary in size and there are many different network protocols available that the HMI should be able to communicate with. This puts a lot of pressure on the HMI development. It is not unusual that PLC manufacturers provide their own tool for HMI development, but this is usually limited to the hardware manufacturer products. Because of these limitations, multiple HMI development tools have been used on Yaskawa. Different toolkits handle in different ways and they are sometimes hard to learn, as having to use multiple IDEs hampers the interface development in many ways.

1.3.2 Interface design

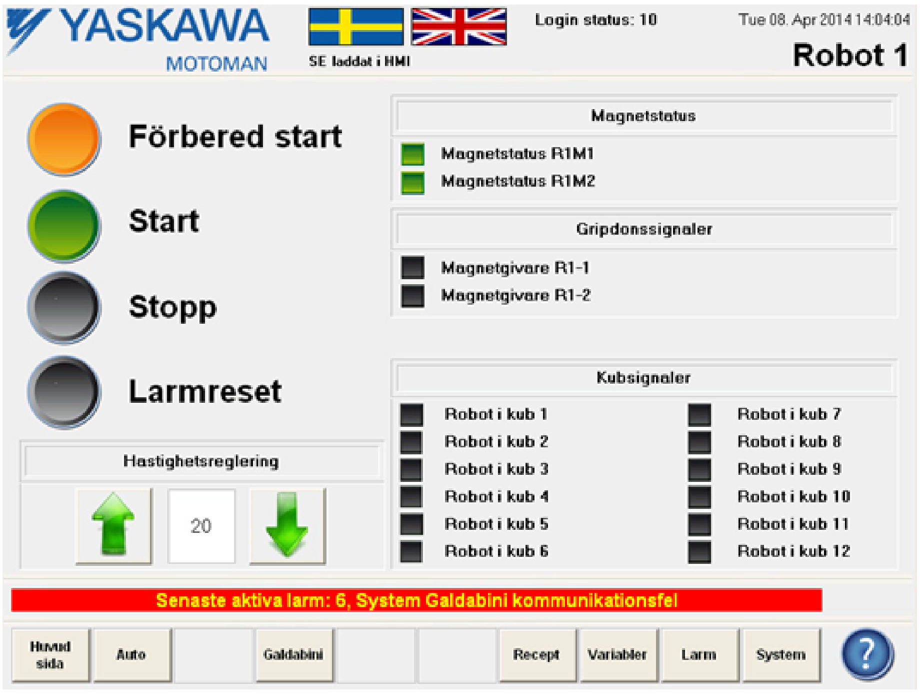
It is hard to design large software systems, but it seems like the user interface is often more difficult to design than the other parts of the system. Brad Myers listed some of the reasons why it is so hard[5]:

- The designer has to know and think like the user.
- Various aspects must be balanced (standards, graphic design, technical writing, internationalization, performance, multiple levels of detail, social factors, legal issues, and implementation time)
- User interface design is a creative process.
- Iterative design is difficult.

Some of the difficulties that Myers brings up have since 1994, when the article was written, become much easier. But as he said in the article, these problems are not likely to go away and interface development remains problematic. To be a good interface designer, the developer has to think himself in the role of the user to predict what he is going to do. It has shown that programmers and designers cannot remember what they used to not know; therefore it is hard for them to think about how the application looks like for a novice user. The interface usually requires deeper understanding of the user than of the design of the functionality, since it must match the target audience's skills, expectations and needs [5].

The look and feel of the application is getting more and more important to compete with similar products. The user do not want to spend time reading manuals, they want to be able to complete their task. Interface design is a from of art and it is a creative process. *Iterative design* is recommended for interface design. Iterative design focuses prototyping and repeatedly redesign and test the application with actual users. It can be hard to test an interface, since it usually gives a lot of options to the user. Making iterations cannot fix a poor design and fixes can also introduce new problems. It can also be hard to find proper test persons which are representative of the target audience.

Many of the PLC manufacturers provide their own HMI development toolkit to make HMIs for their products. For example Siemens provides a toolkit called



Senaste aktiva larm: 6, System Galdabini kommunikationsfel

Figure 1.7: Example of the main status page from a Yaskawa HMI developed in Beckhoff's TwinCAT 2

WinCC Flexible [6], Beijer Electronics offers iX Developer and Beckhoff provides a tool called TwinCAT 2 and they recently released TwinCAT 3. There are also stand-alone products for HMI development for PLCs.

1.3.3 iX Developer

iX Developer is a development tool, created by Beijer Electronics. It is their new UI development tool for their panels which runs iX Software. iX takes advantage of .NET controls, C# scripting, WPF objects and SQL connectivity. It has built in support for features like alarms, recipes and security, data logging, remote access and it can provide the available input and output tags from various PLC brands. A fee has to be paid to Beijer Electronics for every application made with the iX Developer software.

1.3.4 Qt

Qt is a cross-platform user interface framework owned by Digia. It uses C++ or QML, a CSS and JavaScript like language. Qt can be used under open source (GPL and LGPL) or commercial terms. Qt uses Open GL and is therefore able to run on multiple platforms, including Windows CE, which is important in this case. It uses vector graphics which makes resizing the application to different screen sizes easy. In the IDE there are a lot of basic from controls available which can be used in the graphical design view.

1.4 Related work

Fowler, M. writes about the importance of separating the user interface from the application logic. The user interface should, according to Fowler, only be responsible to receive input and to show information. A separate part of the program should handle calculations, validation and communication. Fowler calls these separate parts presentation code and domain code. The domain code should never reference the presentation code, it should be a base on which multiple interfaces can be developed. This makes it possible for teams to work in parallel on the different parts of the program during the development. [7].

Presentation-Abstraction-Control (PAC) pattern and Model-View- Controller (MVC) pattern are two patterns which focus on separating the user interface implementation from the part which processes the program logic. In the MVC architectural pattern, the application is divided into three parts; model which contains core functionality and data, view that displays information to the user and controller that handles user input. The views and controllers together are the user interface. PAC pattern focuses on building a hierarchy of cooperating agents. The agents are responsible for a specific aspect of the application and consist of the three parts: presentation, abstraction and control [8].

A case study in Gothenburg examined the benefits of automatically generated HMI screens and PLC code. They used the SIMATIC Automation Designer from Siemens. The study indicates that the auto generation of code and screens makes it easier to keep the same structure and naming standard in every project. This can help to ensure a corporate standard and quality assurance [9].

Another way to keep the standards and quality is to reuse code from a standardized code base. W. Frakes and K. Kang summarize research made in software reuse. They define software reuse as the use of existing software or knowledge. Reusable asserts are the software or knowledge that can be reused and reusability is the property that indicates the probability of reuse. Software reuse is an important aspect when people want to develop bigger and more complex, more reliable, less expensive and less time consuming software. A key to software reuse is domain engineering (aka product line engineering). By reusing knowledge and software from the same domain (product line), quality and productivity can be improved [10].

Chapter 2

Interview study

To answer research question 1, interviews were made with people who are working with HMI development on Yaskawa. The aim was to gather information about their current work process and hear their thoughts regarding what could be improved.

2.1 Method

The method chosen to answer research question 1 was interviewing. The interviews were carried out with selected people from Yaskawa to gather qualitative data about the HMI development. The questions asked focused on collecting information about how the development process is executed today. Opportunity was given to discuss what is problematic with the current development and what employees think could be good improvements. The room for discussions and deeper insight in the area was the main reason why interviews were the chosen methodology over, for example a survey where it is hard to follow up with supplementary questions [11].

2.1.1 Preparations

The questions asked were open and gave possibility to the interviewees to answer with extensive and describing answers.

Four of the available candidates who have been in contact with HMI development on Yaskawa were interviewed. This may have seemed like a small number of interviews, but each one of the interviewees gave a surprisingly different view on the situation. A more informal interview was also made with a fifth person which provided an additional opinion on the situation.

We did not make any assumptions about the outcome of the interview as the focus was to listen to the interviewees' thoughts and identify the problems that they encounter. The interviews were carried out in Swedish since it felt more natural for both parties.

The questionnaire was written following guidelines given by Eriksson and Wiedersheim-Paul [11]. This was implemented so that the nature of the questions would not conclude in yes or no answers, nor that they would mislead or cultivate subconscious bias. A test interview were performed with one of the interviewees, which resulted in improvements to the questions, some of them were removed and some were modified. Overall this helped to focus the interview questions so that it would provide relevant information about the subject area. The final questionnaire can be found in Appendix A. The modified questions were later asked to the test interviewee in a follow-up informal interview.

2.1.2 Participants

The interviewees were 4 people who have been working with projects involving HMI development on Yaskawa. They have varying experience from different types of HMI development environments and projects of different sizes. The interviewees were selected because of their experience and the insight that they could contribute the research. They were selected by the company sponsor, and he was also one of the interviewees.

2.1.3 Execution

The information that the results from the interview would be presented anonymously were sent out in advance. This was not always explicitly told when the interview started. Each interview was around 1-1.5 hours long. They were arranged on different days and times, when it was found fitting for the interviewee. The interviews took place in the office of each interviewee and the equipment used was a laptop to take notes. The questions consisted of a mix of open questions and some more structured ones, the full questionnaire can be found in Appendix A. The interviewees were very willing to talk and describe their project experiences. Extensive notes were taken during the interview sessions to create a rich and annotated transcription immediately after each interview.

2.1.4 Analysis

The methodology described by J. Craswell[12] was followed to analyse the data. Due to the small number of interviews, some of the steps were found obsolete and were therefore modified or skipped.

A transcript was written after each interview. While doing this, notes were written in the margin and reflections were made about the content. This provided an overall feeling about the message and content in the data. In some cases, extra questions rose while reading the notes and these were asked in hindsight via mail to fill the gaps.

When all the interviews were made, the data was examined again to distinguish the general consensus between the interviewees. Reflecting on the written observations from the interviews were useful when analysing the answers from.

The process of coding data and finding themes and keywords were skipped. Instead all the data was put in a table to get an overview of what the different people had answered on the same questions. This helped to find common answers and different opinions. Because the questions were written to focus on three different areas, who the person is, his experience from HMI projects and what improvements he would seem fitting, these were also found to be good rubrics for presenting the results.

2.1.5 Validity threats

Some of the validity threats that are discussed by Wohlin et. al. [13] are relevant concerns for the reliability and validity of the analysis. Because of the low number of interviews, there is not enough statistical power to draw any conclusions or make a generalization of the problem.

The answers from the test interview were used in the results. The updated questions were asked in hindsight so it provides the same data as the other interviews. The fact that a second chance was given to give input can be a validity threat, because the person knows more about the subject than he did the first time. Due to the nature of the questions asked, we do not feel that this impacted the results. The questions asked were mainly focused on information about the work process; these answers are not likely to change over this short time period.

We feel that the interviewer had little to no effect on the opinions of the interviewee, because they already have the experience and their opinions about the subject. A bigger threat is however that the interviews can have started a discussion among the developers at the company and this can have led to some changes in their opinions. This may have influenced the answers from the interviews which were executed last, but we do not think that was the case. The fact that the investigation have highlighted the problem at the company will make it hard to recreate the study.

One of the interviewees was the company sponsor and another was the interviewer's father. The close relation between the interviewee and the interviewer could be a potential validity threat. But after the analysis, the answers from their interviews did not seem biased.

2.2 Results

The following section presents the results from the interviews. The questionnaire used can be found in Appendix A.

2.2.1 Subjects

The interviewees are employees at Yaskawa and they have all been involved in HMI development several times. Three out of four have been working at the company the last 15 - 25 years. The other one have been at Yaskawa for almost 5 years. They all hold different positions at the company. Two are from the software development department, one is from the sales department, but he is also very involved in the HMI development, and one is an electrical designer and he is mainly working with PLC programming and electric design.

The interviewees who have been at the company for more than 15 years have great experience from HMI development. They have participated in 50 to a couple of hundred projects where a HMI application was developed for the robot cell. The other one have been taking part in 2-3 projects at this company.

2.2.2 Recent project

One of the interviewees is currently part of an ongoing project and another was involved in one over six months ago. The projects had varying length and scope and there were usually one or two people tasked in completing the HMI application. The largest project that one of them had participated in recently was stretching over half a year. Even though it was finished half a year ago, they are still doing some small fixes on it. It was 5 people working on this project, two of them were working full time with the HMI development. The project was very complex because it involved several PLCs and HMI screens, which should provide different data. The more usual length of the projects was around two months. These projects involved development of HMI applications for a single soft PLC or a PC with windows 7.

Different development tools were used for each of the latest four projects:

- Microsoft's Windows Forms [1]
- Beijer Electronics' iX Developer [2]
- Beckhoffs' TwinCAT 2 [3]
- Siemens' WinCC Flexible [4]

Windows Forms a Microsoft development tool which provides basic components for interface applications, it requires good knowledge in C# and it leaves

[1] http://msdn.microsoft.com/en-us/library/dd30h2yb(v=vs.110).aspx

[2] www.beijerelectronics.com/web/beijer_electronics.nsf/docsbycodename/ix_software

[3] http://www.beckhoff.com/english.asp?twincat/tcatdow.htm

[4] http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/wincc-flexible/pages/default.aspx

a lot of freedom to the developer. The other three are toolkits developed by PLC manufacturers for HMI development, usually for their products. These are programs which are custom made for HMI development which means that they come with a lot of built in functionality to ease the common needs of a HMI application.

When asked which other HMI development tools they had used, the person who used Windows Forms said that he had only used this and not tried any of the PLC manufacturer's HMI development tools. The other three told similar stories, that they had worked in different HMI development tools from different PLC manufacturers and they were not familiar with Windows Forms. Among the PLC manufacturer's HMI development tools, iX Developer, TwinCat 2 and WinCC Flexible were the favourites. They preferred these because they think that they are easy to use.

> "Even though WinCC Flexible is very old and hard to use, it is still the program that I prefer. I learnt it the hard way but now I know how to use it." [5]

The person who preferred iX Developer motivated it with the fact that it is easy to use the development tool and the documentation is very good.

> "They provide a big library with a lot of premade components which allows the developer to just drag-and-drop them into his project. It is also a lot of example code and styles available."

The same was said by the person who had only used Windows Forms. He said that it is easy to drag-and-drop different components to the application and you can easily add your own look to it.

One of the interviewees says that in WinCC Felible and TwinCAT 2, it is hard to change the resolution of the application during the project. This is because these development tools are built on pixel graphics. They provide no anchor functionality, which is crucial to assure correct and good-looking scaling to different screen sizes. That means that each project has to be designed for a specific resolution from the beginning, which might lead to problems later in the development if the customer wants to use an other HMI panel. iX Developer and Qt build on vector graphics and it makes it easier to build flexible and re-scalable applications.

2.2.3 Project structure

The following questions were posed about the overall project structure. The answers were almost the same from all the participants. They said that it is very

[5]The quotes are translated into English by the author.

easy to change the design or the layout of the interface if some new features have to be added later in the project or if things are not looking good. Everyone said that they reuse code from earlier projects to speed up the development, but they feel that this could be mate in a better and more structured fashion.

> "We try to reuse as much as possible. You usually take an older HMI application developed in the same environment that you are going to use, open it and copy-paste the parts that you need. Another way is to copy the old project and have is as a starting point. Then you remove and rename the old components to fit the new application."

Logical code tends be reusable and even though large portions usually is case specific to fulfil the needs of each project; graphical asserts are also reused to a lesser extent. The developers say that they think about reusability of code during the project.

There is no official code standard which has to be followed by everyone on Yaskawa. This results in the usage of different programming styles and everyone says that you can recognise who wrote the application by looking at the code. A code standard was drafted for the latest large project since it would be a lot of people involved and the application would be very complex. The standard was however interpreted in different ways and in the end the separate parts looked different despite the efforts to apply a standard.

Two of the developers argue for the use of a code standard and means that it would improve the HMI development. One of them says that it would make the code look more similar and that this would make it easier to fit different parts together. It would also help the readability and reusability of old code, since it would have a know structure. He thinks that this is something that has to be fixed, but due to more urging matters the adaptation of a coding standard has been classed as a low priority.

The same thing applies to the design of the HMI layout; the company do not advocate a standard for the design of HMI applications. Because there is no reference on how the final product should look, it is up to the programmers' feel and taste of what looks good to decide the design. This can make the interfaces look very different from each other and they depend on the developers design skills and the amount of time he put into designing the interface. It is however common that the interface of a new HMI will be produced in a similar fashion if the same customer has purchased a robot on a previous occasion. Older applications developed in the same environment can also be used as reference for the design.

There is no one who intentionally implement any specific design pattern for the HMI and the concept of the design pattern did not seem familiar to, for example the person who is mainly a electrical designer.

2.2.4 Improvements ideas

The thoughts on how to improve development efficiency differed greatly. But the one concurrence between the group was that a single development environment should be chosen and used across the team since it is very inefficient to use multiple development tools.

> "The learning curve is relatively high for many of the development tools and it does not help the developer that the way that they are used differs a lot."

Another common idea was that there should be more ready to use components with a standardised look available. This could be custom buttons or even whole layouts for common pages, with a carefully developed design representative of the company. If these were put in a library which would be shared among the developers, everyone would have the same pieces to build the HMI from and that would unify the look of the applications.

Some of the developers were discussing the benefits of using pictures instead of text in the HMI application. This could reduce the work required for translating the interface into multiple languages. Today it is common to translate the HMI into at least Swedish and English. To take advantage of providing information via pictures instead of text, the usage of them has to be consistent and easy to understand. Another good idea would be to use standard sentences for common lines, these could be available in multiple languages in a dictionary.

> "The standardisation would make our HMI applications look more similar and this would help the customer to recognise functionalities and feel familiar with the interface."

One person said that it may be a good idea to limit the design choices and force a standard look for the pages via the development tool. He had previous experience from working with a HMI development toolkit that was built around this idea and he thought that was good. He thought this made the development process easier and faster, since no time had to be spent on designing the interface, it just required the developer to add the functionality.

2.3 Analysis

Several areas of importance are discussed in the following sections. They result in some key factors to take into consideration when choosing a HMI development tool.

2.3.1 Development environment

Many benefits could be gained if the developers used a single HMI development tool. For example, the code base available on the company would be growing faster and all material produced would be usable for everyone. It would be easier to apply a common style and standard, since everyone has the same tools available. If everyone uses the same environment, team work and experience can be shared and good solutions promoted. If the same development tool is used by everyone, anyone can come in and assist. Since everyone would know the development environment, it would also make it easier to perform maintenance and additional fixes that sometimes has to be made after a project is deployed.

The HMI development tools from PLC manufacturer are great, but some of them are limited to only communicate to the PLCs from the same manufacturer. There are exceptions, such as Beijer Electronics' iX Developer, which supports all the common PLCs but it still requires the developer to use their HMI panels. Since it is possible for the customer to request a PLC from a specific brand, the developers has to adapt the HMI development. This is the reason why so many different development tools are being used at the company. Another reason is that when applications are getting updated or a customer wants to order an additional robot cell, the HMI is usually developed in the same tool as the first one.

The platforms that the HMI application can be deployed at is an important factor, since the company only wants to use one development tool for all applications. In the evaluation of HMI development tools we will call this criteria *Open platform*.

2.3.2 Familiarity of tool

Each person has his own favourite program which usually is the one that he used the most.

It can sometimes go up to half a year before the developer has to use the HMI toolkit again because there is not always need for development of HMI applications. It is therefore important that the toolkit is easy to pick up again after a longer period of time away from it.

It is also important that the development tool is easy to learn and use and this will be called that the development tool is *Accessible* for the developers at Yaskawa.

2.3.3 Standardisation of design

The PLC manufacturers development tools are made for one thing, to make it easier for the developer to make HMI applications for their hardware. Therefore, there are a lot of domain specific functionality available, like schedulers and alarms. This makes it very easy to make standard pages and provide common functionality to applications. In iX Developer, there is even a function to set different languages on separate sections in the interface, which shows that it is made to reduce the effort needed to translate the interface to different languages.

In iX Developer the developer is given a lot of components which can be used in the HMI. There can be limitations to the modifiability and customisation of the components, but he still have control over the design and layout of the application. We could also think about a development tool where the design is set and the only task for the developer is to add the functionality needed in the HMI. This could reduce the time spent on designing components and layouts and result in similar looking applications. But by making a development tool with the purpose to make the HMI development easier, it will at the same time limit the options of the designer and programmer.

If a more general development environment is used, there are more possibilities to create unique designs. By using templates or having a written documentation that the developers has to follow when doing a HMI, the development can be structured but the platform remains open and flexible. The downside is that this can require a lot more work and knowledge from the programmer.

Even applications which were created in the same development tool can right now look very different due to the lack of standardisation (see Figure **??**). It is clear that a standard design is needed to make the final products from the company look similar. How to enforce a standard look could either be made with documentation and guidelines on how certain parts should be designed and implemented. Or another solution is to restrict the design via the development tool.

In most of the interface development toolkits, it is possible for the developer to add custom components. A common customisation is to add a unique look on a button. It may also be possible to create bigger systems of connected controls, with results in complex and feature rich components. This is usually possible in more general development tools like Qt or Windows Forms.

The ability to create templates and custom components are very important as it can save a lot of time for the developer and help to unify the look of the application. It will be addressed as the *Custom components and templates* category.

2.3.4 Code reuse

Currently, the possibilities to reuse code is limited to the number of projects that was previously created in that specific environment. It is very hard to reuse other things than images or a general design from a project developed in an other development tool.

If a standard were adopted and followed, new templates and components could be added to a reusable library. For code to be reusable, it should be documented how and when it should be used. Code reuse lets the development team take advantage of previous successful solutions. It reduces the development and testing time required since a component should be fully functional and tested before it is put in the library.

Since design patterns was not commonly known or used and no code standard is followed among the developers at Yaskawa, it could be hard to structure the code to be reusable. But as seen below, there are a lot of benefits that comes with efficient code reuse so it is probably an adaptation which could be valuable for the company in the long term.

Some benefits from reusing code are:

- Avoidance of errors/bugs, especially the hard-to-find ones.
- Maintainability, by promoting proven design principles.
- Maintainability, by requiring or recommending a certain code standard.
- Performance, by discarding wasteful practices.

The possibility to reuse code from the HMI application will be addressed as the *Code reuse* category.

2.3.5 Experience in coding

The number of people who can work with the HMI development is limited due to high knowledge requirements on many of the different development tools. Adding on top of that the need to use multiple tools limits the personal available even more. Right now people who are not very familiar with PC programming are also developing HMI applications which makes it important to adapt the development tool to their knowledge. This is why the more domain focused development tools for HMI development are very popular, since they are made to make the HMI development more accessible to non-programmers.

The amount of coding that has to be written in the HMI application should be minimal. This would make the developmetn tool available for a non-programmer. To build an user friendly interface is its own field of knowledge. Not all programmers have the feel for what is good and user friendly design, therefore it is important to open up the development to multiple people. Another solution would be to have templates made by designers and a full documentation which the programmer can follow to make a good HMI.

Because the amount of coding required to develop the HMI limits the people who can use the toolkit, it is an important factor when selecting it. This will be one of our comparison criteria called *Non-extensive coding* in the table.

2.3.6 Evaluation categories

From the previous sections, various different criteria were outlined and identified to be of importance when selecting a HMI development tool. These properties were selected based on the results from the interviews, it is therefore possible that they are somewhat affected by the current situation on Yaskawa.

The table 2.1 below contains the categories in the right column. Listed in the header row are some of the HMI development toolkits which would be interesting evaluation subjects.

	Qt	iX Developer	Windows Forms	TwinCAT 2	WinCC Flexble
Open platform					
Accessible					
Custom components and templates					
Code reuse					
Non-extensive coding					

Table 2.1: The table shows the categories which may be important in HMI development. In the header row are some of the development tools which would be of value to evaluate.

These where the main aspects which were found important for the improvement of the HMI development on Yaskawa. Since Yaskawa is one of the biggest in the domain of industrial robotics and they are a well established company, they are a good representative of the domain. It is therefore very possible that other similar companies have the same problems.

Other aspects of the HMI development tools could have been important to evaluate. This could be the time to perform a task, error cost, performance, documentation and support. We selected our evaluation categories based on the results form the interviews, since these would be the most important aspects for this company.

Chapter 3

Experiences from prototype implementation

After the analysis of the data from the interviews, several important criteria for a HMI development tool was shown. A case study were made to evaluate two of the available HMI development toolkits to see how well they match the needs posed on HMI development.

3.1 Method

The method used to answer the second research question was a case study with action research. This method was chosen because we wanted to evaluate how well the development tools matched the criteria in the table 2.1. This information was not available and it was in general very hard to find any research on the usability of the development tools. To gather the data needed for the evaluation of the toolkits, we had to conduct the study of them ourself.

Action research is a flexible method to solve a presented real world problem. This method is used when collaborating with both theoretical research and practical execution. During the case study, the impressions, problems and how long it took to solve them, were written down in a diary.

3.1.1 Choice of development tools

The development tools chosen for the case study were Qt and iX Developer.

iX Developer is one of the bigger HMI development tools provided by a PLC manufacturer. It is developed by Beijer Electronics and has support for communication with multiple PLC and PC brands. The development platform is custom made for HMI development, naturally this means that functionalities that are usually used in HMI applications are already there. iX Developer was chosen for the evaluation since the software is modern, looks promising and has good support for communication. It is a good representation of a tool that is developed for HMI development and it is aimed to be easy to use by people in that domain.

Qt is a multiplatform UI development tool with basic components available. It has similarities with Microsoft's Windows Forms in usage, but Qt is a open source project which means that it is continuously updated and patched. There is no previous experience of using Qt for HMI development on Yaskawa. To gain new information, it was decided by the company that they would prefer a evaluation of Qt over Windows Forms. Both of these represent the more general purpose development tools in which it is up to the developer what he wants to use it for. It was important to evaluate the pros and cons of this type of development tool in contrast to the more domain specialised iX Developer.

3.1.2 Preparations

The two frameworks were downloaded from their respective homepage and installed. For the evaluation of iX Developer, the 30 days trial version was used[1]. It is not limited in functionality only in the amount of time that you can use it. The Qt application was made using the Qt Creator design tool. The open source and licence free version was used[2].

To get a basic and equal understanding of the frameworks, a simple test application was made in both of them. This consisted of basic form elements, events and functions connected to them. There are a lot of documentation material available for both programs. The documentation about iX Developer was easy to use and it provided simple and direct answers to all the questions that arose. This was however not needed to a greater extent, since the development tool was surprisingly easy to learn. The documentation of Qt is also very good, but because of it being a more general tool, it had to explain more detailed and advanced things. This made the development tool a bit harder to start using right away.

It also took a much longer time to start up with Qt because there are a lot of different versions available. There is Qt Widget, which looks a lot like Windows Forms and then there is a newer tool called Qt Quick which looks like a mobile application and it was hard to know which version to use. In iX Developer, the freedom is limited and the developer can only choose which screen the interface should be developed for and which PLC or PC it is going to communicate with. This automatically sets the right resolution and provides the in and output tags if a PLC was selected.

3.1.3 Test application

A lamp that shows the status of the robot is a usual asset in a HMI application. There is usually a Start, Stop and Emergency stop button available on the main screen of the HMI. The corresponding lamp will glow when the state is active. The task evaluated in the different development tools was how to make a lamp like this for a HMI application. The lamp should be listening to the status of the

[1] `http://www.beijerelectronics.com/web/beijer_electronics.nsf/docsbycodename/ix_software`

[2] `http://qt-project.org/downloads`

robot, but since we did not have time to implement the interface to the robot, we made it listen to the click event of a button.

3.1.4 Validity threats

There is no guarantee that the implementations were made in the most efficient way. When using a new toolkit it is possible to miss features that could have been useful or made the task easier.

Since the test applications were not tested on accrual robots, we cannot evaluate the whole process of deploying the HMI to a real case scenario and see what problems might arise when maintenance is required.

Another validity threat is how well the opinions of the evaluator represents the actual opinions from the view of a typical developer. What is seen as non-extensive coding is a matter of perspective and it is therefore important to know what it is compared to and what the scale is.

3.2 Results

Following are the results from the case study of the two HMI development toolkits.

3.2.1 Qt

By default there is no basic component for displaying multiple textures so it was necessary to create a custom widget for this.

These were the required steps to make the custom widget:

- Make custom class which inherits from a Qt base component, we used QWidget.
- Add member variables for state and two images.
- Load images
- Overload the onPaint function so that the image drawn depends on the state.
- Add a Slot for the custom event where the state will be toggled.

These steps only have to be done when a new component has to be made or modified. There are no limitations on what can be made, but it requires good knowledge in C++ programming.

When the desired component is ready and available in the toolbox, the following steps are required to make the test application:

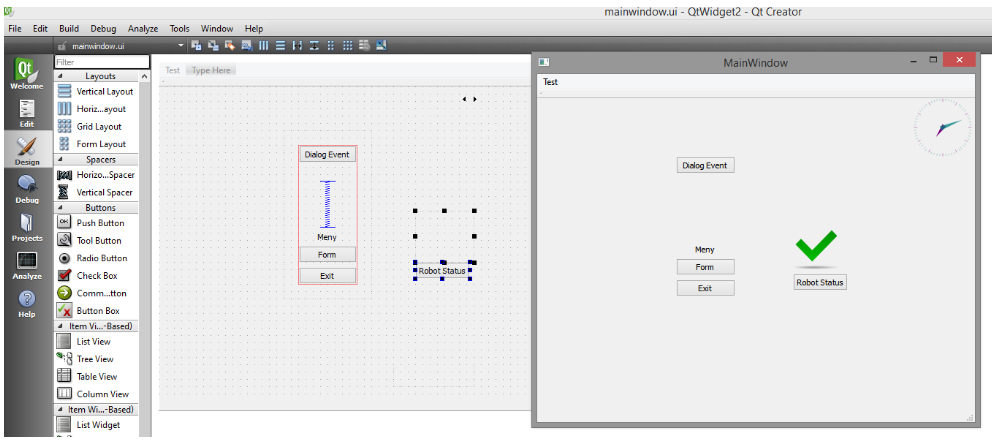
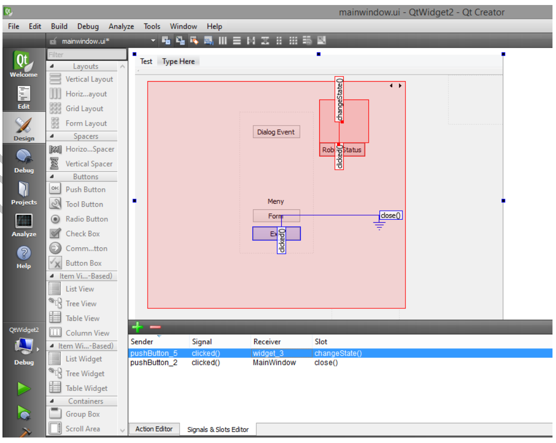


Figure 3.1: The test application developed in Qt. To the left is the graphical interface designer and to the right is the running application.

- Drag-and-drop components from toolbox to the window. We used the custom control and a button. [3]
- Connect the button with the custom control using the graphical interface and select the proper Signal and Slot from the list.

Signals and Slots are used in Qt to connect a trigger with a follow-up action. It is necessary to add the custom Slot to the list manually the first time the custom control is used. After that, the custom Slot can be selected from the drop down menu, same goes for custom Signals. This means that when a custom control is done, there is little to no requirements on programming. One negative thing is that the custom components wont be visually represented, instead it will be displayed as one of the base components in the designer.



Sender	Signal	Receiver	Slot
pushButton_5	clicked()	widget_3	changeState()
pushButton_2	clicked()	MainWindow	close()

Figure 3.2: In Qt, it is possible to connect signals and slots via a graphical interface by connecting two components and selecting the desired trigger and action.

3.2.2 iX Developer

Tags are an important concept in iX Developer. The value on the tag can be

[3]The button should represent the status of the robot. In a real case scenario, it would have been necessary to make a class which could communicate with the robot. From there events could be trigged.

toggled, increased, decreased and set
by a action and the controls that are connected to that tag will be updated accordingly. In iX Developer there is is a component called multiPicture which fits for the task. To make the test application, the following steps were performed:

- Drag-and-drop components to the interface window. Here, a multiPicture and a button were used.
- Add pictures to the multiPicture. There is a dialog where the pictures can be added.
- Assign what value each picture will represent. [4]
- Make a tag.
- Add a function to the button which toggle the tag on the click event.
- Assign the tag to the multiPicture.

No code had to be written and it was very intuitive and easy to use the development tool to perform the task.

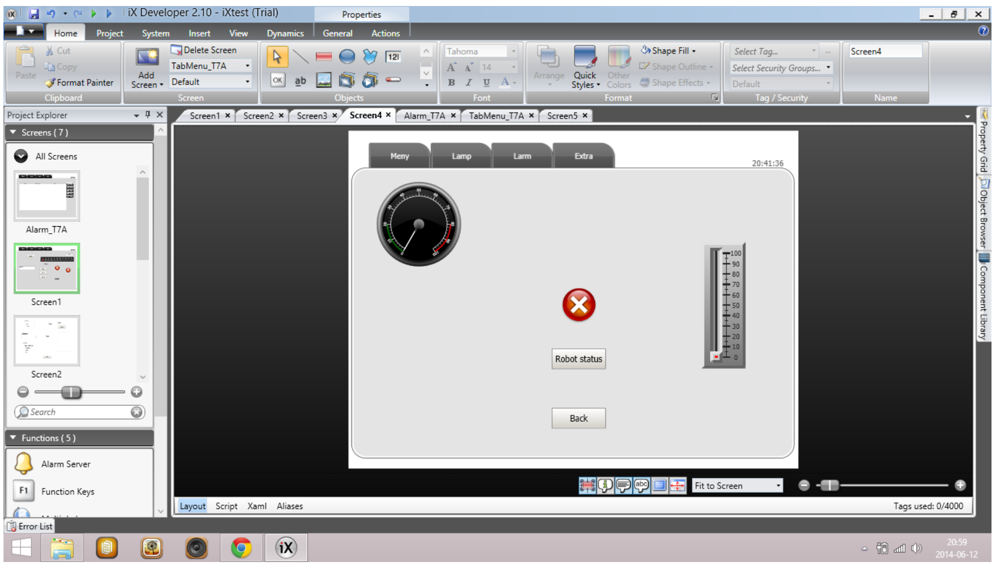


Figure 3.3: The test application developed in iX Developer. The possibility to add backgrounds to the pages is useful to keep the same style on each page. This can for example be a menu with buttons to the different pages.

[4] The possibility to set a range for when a specific picture should be displayed can be very useful in, for example alarm systems, where a specific picture might be shown if a value extends the normal.

3.3 Analysis

With the results from the case study, the table that were presented in section 2.1 of the last chapter could be filled in for Qt and iX Developer.

	Qt	iX Developer	Windows Forms	TwinCAT 2	WinCC Flexble
Open platform	Yes	Partially			
Accessible	Partially	Yes			
Custom components and templates	Yes	Partially			
Code reuse	Yes	No			
Non-extensive coding	Partially	Yes			

Table 3.1: The table shows the results from the case study of Qt and iX Developer

3.3.1 Open platform

Qt is available on multiple platforms, including Windows CE which is important in this case. Since it is open source it is free to use and it comes without any for the restrictions which are common among the HMI development tools provided by PLC manufacturers.

The iX Developer can be used on HMI panels from Beijer Electronics. For every application, the developer has to pay for the usage of the software. iX Developer is however open in the sense that the HMI panel has support to communicate with all the common PLCs and PCs. This is not always the case among this kind of development tools, since the manufacturer wants the developer to use their hardware.

In this case, Qt is the favourable choice, since it limits the dependencies towards a specific company and can be used for free.

3.3.2 Accessible

The HMI development in iX Developer is adapted to fit non-programmers. It has a friendly interface and it provides all the functionality that is needed in a HMI application. Because all these components and events are available from the start, it is very easy to pick the parts needed and quickly make a good looking HMI.

Because Qt is a much more general development tool, it is also much more complex since it give so much freedom to the developer. This can make it hard to use in the beginning because it is hard to know where to start. Even though it has some similarities with Windows Forms, which we had some experience from, it still took some time to understand the concept of Signals and Slots and how to use them. The documentation can be a bit overwhelming due to the complexity. When we say that Qt is partially accessible, it is in comparison to iX Developer which is very easy to use and a code based interface design tool where all the components has to be positioned without a graphical design tool.

iX Developer seems to be the better choice when taking into consideration the knowledge of the HMI developers at Yaskawa. They are the target audience towards which iX Developer was developed. Because the tool is easy to use, it is also easy to pick it up after longer periods of time away from HMI development. This can be a problem in more complex development tools, where several steps may have to be remembered and performed in a specific order to fulfil a task.

3.3.3 Custom components and templates

In Qt it is fully possible, and usually required, to make custom components. This gives a lot of freedom to the developer to make unique components and designs. A negative point is that the custom made components are not visually displayed in the interface designer, this requires more of the developer since he has to remember how each components look. The process of developing custom components for Qt can be quite advanced, but when they are made they can used as any other basic component. The responsibility to develop custom components could therefore be assigned to one person, everyone else would just need to use the produced components which is not that advanced.

There is usually no need to develop new components for iX Developer, but it has support to load custom made objects and .NET controls. When a component is designed or a whole page, they can easily be saved in the component library or as a page template. This makes it simple and easy reuse components which is important when a standard is applied.

The development of custom controls can be made by anyone in iX Developer but in Qt it requires advanced programming knowledge. When the components are made and available to the designers, it is still easier to use and bind events to them in iX Developer and therefore this tool fits the target audience better.

3.3.4 Code reuse

The programmer has full control over the code in Qt. This allow for optimizations and application of design patterns to structure the code in a reusable way. However, this requires good programming knowledge. All code is generated automatically in iX Developer and it is therefore irrelevant to think about code reuse.

It requires a good structure and well written code to take advantage of code reuse but if it is made in a successful way, it can speed up the development process as well as it reduces the amount of code which has to be written. This could make Qt more accessible for the less experienced programmers. If no good code base can be developed, we feel that iX Developer is a easier tool to reuse components in.

3.3.5 Non-extensive coding

The programmer has a lot of freedom and control over the code in Qt but also a lot of responsibility. Qt requires C++ programming to make custom components, which can be very hard for people who are not used to the language. On the other hand, when the components are made they can be easily added to the application and events can be connected via a graphical interface.

Because of the predefined functions and the code generation there is usually no need for coding in a normal HMI application when using iX Developer. There is however still possible to make custom functions with scripts. The absence or coding requirements makes the development tool more accessible and this is a big advantage over Qt.

3.3.6 Summary

To make Qt a valid option for HMI development on Yaskawa, it would require a library of custom made components and a standard for the design. This would allow the developers to take advantage of code reuse. An extensive documentation on how to develop custom controls would be required to guarantee quality and standard on new components. This would mean that less amount of new code is required for every HMI application, and that would lower the requirements of programming knowledge. The benefit is that Qt is open, free and gives the programmer full control over the program.

iX Developer is a very good development tool as it is. Everyone can participate and contribute in the development of a standard design and a custom look for the components. The tool is accessible and easy to use, even for non-programmers and this is a important factor when considering the knowledge among the HMI developers. The only crucial drawback of using iX Developer is that it is limited to Beijers HMI panels and it also cost some money for each application produced.

Chapter 4

Discussion

The current HMI development work process has been examined from an outside perspective and several important criteria for a development tool have been highlighted. The leading issue was that it is a necessity to use a single development tool among all the developers because using multiple development tools has several negative consequences. Good ideas and parts of code can be easily forgotten or made redundant, because they are made in different development tools. Not to mention the time has to be spent to solve the same problems in multiple development environments.

There are many factors that affect which development platform is the best choice. This includes the reusability of code, design assets, the knowledge and coding requirements and the possibility to enforce a standardized HMI look. The most important factor being that the development tool should meet the needs of the developer.

4.1 Study evaluation

The interviews provide valuable information about the situation at a specific company. The selected methodology was an important tool to achieve these results. It would not have been possible to get the same insight to the situation by for example, doing a survey.

The results however are produced from a small number of subjects and it is hard to make any general assumptions based on them alone. It would have been a good idea to contact multiple companies to get a bigger picture of the HMI development. This could have made the results more general for the HMI development domain, but with the current information it is very hard to know if other companies have similar problems or if this is an isolated problem at Yaskawa. Yaskawa was the basis of the investigation and many of the important aspects which are discussed, come from the interview results because they are problems which occur in real case scenarios. Yaskawa is one of the bigger companies in the domain of industrial robotics and can therefore be seen as a representative company for the research.

The research was carried out on a request from Yaskawa and that is reason why we investigated this company. We believe that this did not add bias to the results produced, it only helped to give an insight of the current situation.

The case study of the two selected HMI development tools could have been better structured and made on a larger scale. The results can be somewhat biased considering the programming background of the evaluator. What is considered "accessible" or "non-extensive coding" depends on what it is compared to and this may be affected by the knowledge and expectations of the evaluator. The results would have proven more reliable if multiple people, who could better represent the knowledge of HMI developers, would have participated to produce the evaluation results.

4.2 Open platform

Because of the wide range of technology involved in the industrial robotics, it is important not to tie the development of HMI to one specific hardware manufacturer. It is very valuable if the HMI application can run on any operating panel and be able to communicate with any PLC or PC. The hardware limitation is usually the problem with the development tools provided by PLC manufacturers, because they want you to use their products.

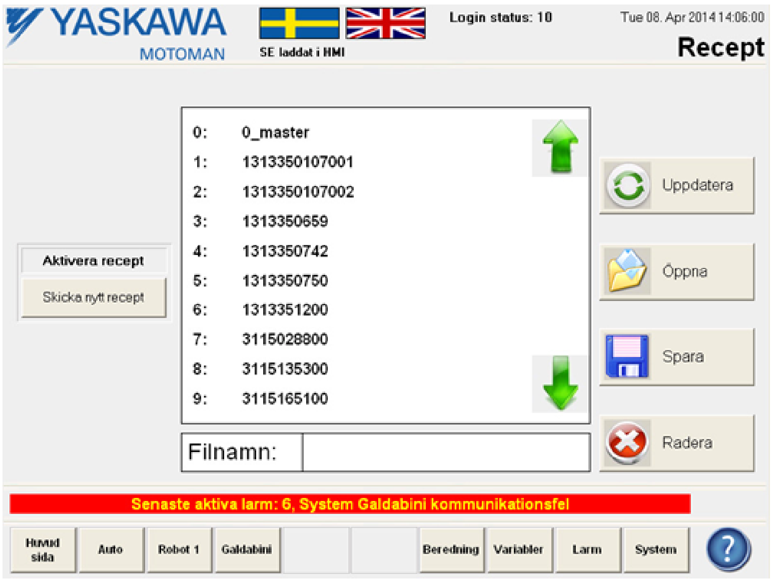
4.3 Accessible

The benefits from using a more accessible development tool are many, especially if there are longer periods when a developer is not working with HMI development. Because there was not enough time to learn everything about the development tools, some functionalities that would have been useful may not have been used to their fullest potential. The view can however bee seen as a first impression and it can also be a benchmark for how easy the development tool is to learn and intuitive to use, which is an important property. It is also very important that it is easy to come back and starting to use the development tool again, without having to read a lot of documentation to remember how to use the program.

4.4 Custom components and templates

As seen in figure 4.1, it is very hard to keep a unified look and feel of the applications without a general standard and it is not certain that the applications will look the same just because they are developed in the same tool.

Using a common standard for the HMI design benefits both the customer and the developer. If the HMI applications has a similar look, the company products can be recognised. It also helps the customer to feel familiar with the products



Login status: 10

Tue 08. Apr 2014 14:06:00

Recept

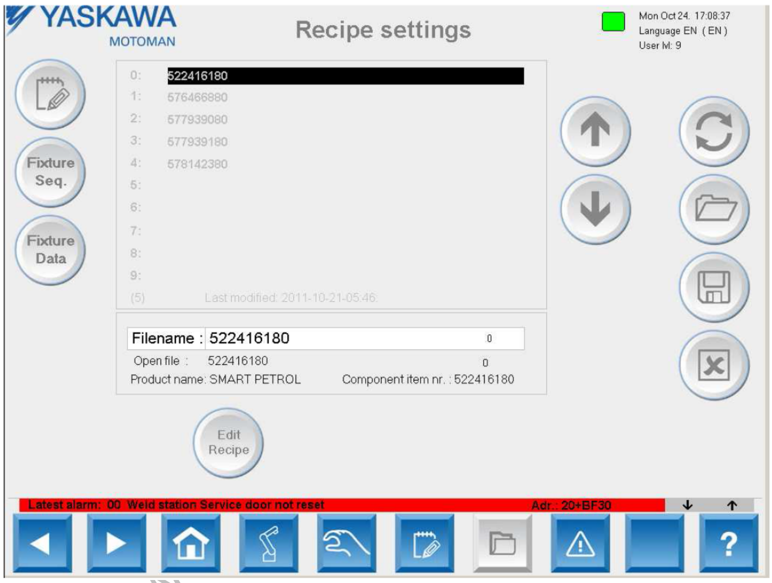
SE laddat i HMI

0:	0_master
1:	1313350107001
2:	1313350107002
3:	1313350659
4:	1313350742
5:	1313350750
6:	1313351200
7:	3115028800
8:	3115135300
9:	3115165100

Filnamn:

Senaste aktiva larm: 6, System Galdabini kommunikationsfel

(a) Developed using Beckhoff's TwinCAT 2



Recipe settings

Mon Oct 24. 17:08:37
Language EN (EN)
User kl: 9

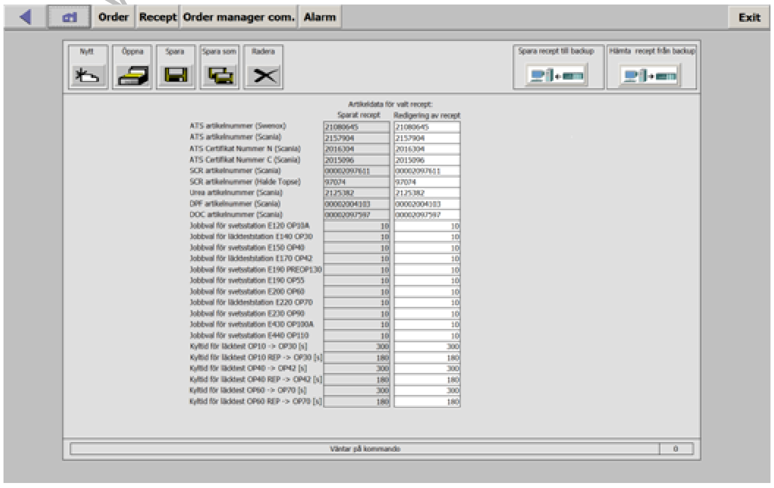
0:	522416180
1:	576466880
2:	577939080
3:	577939180
4:	578142380
5:	
6:	
7:	
8:	
9:	

(5) Last modified: 2011-10-21-05:46

Filename : 522416180 0
Open file : 522416180 0
Product name: SMART PETROL Component item nr. : 522416180

Latest alarm: 00 Weld station Service door not reset Adr.: 20+BF30

(b) Also developed using Beckhoff's TwinCAT 2



Artikeldata för valt recept:

	Sparat recept	Redigering av recept
ATS artikelnummer (Swenox)	21080645	21080645
ATS artikelnummer (Scania)	2157904	2157904
ATS Certifikat Nummer N (Scania)	2016304	2016304
ATS Certifikat Nummer C (Scania)	2015096	2015096
SCR artikelnummer (Scania)	00002097611	00002097611
SCR artikelnummer (Halde Topse)	97074	97074
Urea artikelnummer (Scania)	2125382	2125382
DPF artikelnummer (Scania)	00002004103	00002004103
DOC artikelnummer (Scania)	00002097597	00002097597
Jobbval för svetsstation E120 OP10A	10	10
Jobbval för läcktestsstation E140 OP30	10	10
Jobbval för svetsstation E150 OP40	10	10
Jobbval för läcktestsstation E170 OP42	10	10
Jobbval för svetsstation E190 PREOP130	10	10
Jobbval för svetsstation E190 OP55	10	10
Jobbval för svetsstation E200 OP60	10	10
Jobbval för läcktestsstation E220 OP70	10	10
Jobbval för svetsstation E230 OP90	10	10
Jobbval för svetsstation E430 OP100A	10	10
Jobbval för svetsstation E440 OP110	10	10
Kyltid för läcktest OP10 -> OP30 [s]	300	300
Kyltid för läcktest OP10 REP -> OP30 [s]	180	180
Kyltid för läcktest OP40 -> OP42 [s]	300	300
Kyltid för läcktest OP40 REP -> OP42 [s]	180	180
Kyltid för läcktest OP60 -> OP70 [s]	300	300
Kyltid för läcktest OP60 REP -> OP70 [s]	180	180

Väntar på kommando 0

(c) Developed using Siemens's WinCC Flexible

Figure 4.1: The pictures show the recipe page from three different HMI applications. As displayed, being developed with the same tool does not guarantee that the products will have similar looks.

and it reduces the time the operator has to spend in order to learn how to use the interface.

For the developers, a standardised design principles means that less time has to be spent on redesigning each application. This also reduces the impact of individuals taste of what is good design on the application. When a standard is chosen, custom components can be carefully developed for it. More time can be spent on developing each component once, since they will to be reused in multiple applications and this can increase the quality of the assets.

A standardised principle for layout and component design helps to promote proven design principles, the same design that was proven successful can easily be reused in multiple projects and the wheel does not have to be reinvented every time. With a common standard, the collaboration between team members can be easier. When everyone is working with the same program, tips and ideas can be shared as well as code.

A standard for the look of the HMI applications also opens up the possibilities to hire consults for a project. This is something that is usual for a company this small, and it would make the work of the consults much easier if the company could come and give a clear set of directives of how the implementation and layout should look.

4.5 Code reuse

The possibility to reuse code is important and it can help to shorten the time-to-market and improve the quality buy reusing rigorously tested assets. By applying a design pattern, like MVC of PAC [8], the base functionality can be separated from the individual applications and it can be reused across multiple applications. This is requires more effort from the developers in the beginning to build up the code base, but when it is done it can improve the development.

This is however not an important factor if the development environment is generating the code for the developer. In that case, the possibility to reuse asserts in an easy way is more important. This can be the possibility to apply a custom style to all the components in a easy to manage fashion.

4.6 Non-extensive coding

It is important to choose a development tool which does not require deep programming knowledge if the employees which are going to develop HMI applications does not have a PC programming background. Some of the developers may be electric technicians which are familiar with PLC programming but maybe not with PC programming, which is quite different.

If this is the case, it is important to have a development platform that does not require an extensive amount of programming knowledge and as much should be virtualised so that the code is written in the background, generated by the visual design tool.

4.7 Making a custom development tool

By making a custom development toolkit for HMI development, the tool can be tailored after the needs of the developers. It can be hard to make improvements to the development tools available, because they are owned by a company. Making a custom development tool is a solution to avoid the limitations and fees that often comes with using development tools provided by PLC manufacturers, or to make the development in more general toolkits more accessible and streamlined for the common needs of HMI applications.

A lot of time can be saved by making the development tool more focused on one task. It also opens up the possibility make the development more accessible for a wider or a specific target group, for example people who are not necessary used with advanced programming.

One example of a custom made HMI development tool is the OP-Touch. It is a fairly old environment that was developed by a company that later was bought by Yaskawa. We got the chance to talk to the person who continued to develop this software for his own company. We discussed the topic of why the HMI development tool looks like it does. The UI is designed via a text file were a limited number of controls can be added, for example buttons or lamps. These can be assigned functionality, but the design and layout is not customizable. Instead the controlls are put in a preset grid. OP-Touch is used to update older systems which were built with a earlier version of the software. To make the conversion easier, the same structure was kept in the newer version of the software. OP-Touch is a good example on the positive and negative aspects of making a custom development tool. A good thing is that it is made for a specific purpose and it is therefore developed to make these tasks easy to preform. The downside is that it is hard to keep it updated and modern. If a big update is needed, the backwards compatibility and upgrade possibility has to be considered.

It also shows that limiting the choice of the design via the development tool can work, but it can also hinder improvements or evolution of the applications. The OP-Touch can be compared to the modifiable page which is

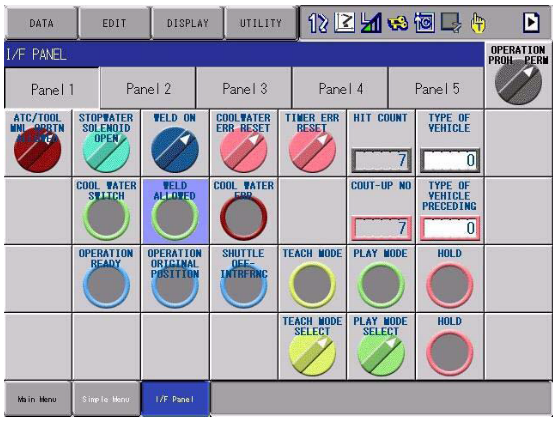


Figure 4.2: The modifiable page on a teach box. The possibilities of custom design on the teach box is very limited.

available on the teach box for the Yaskawa robots. This also has limited customization options which can be a problem when a more advanced HMI is necessary.

4.7.1 Summary

There are a lot of factors which has to be taken into consideration when choosing a development tool for HMI applications. This study has highlighted some of the important factors, like reusability, easy to use and templates for design. These factors are of different importance depending on the target group of developers. In some situations, the programming requirements might not be a problem and in other cases it might not be necessary to do more advanced or customized components than what is provided within the development tool.

The aim of the study was to, in a structured and objective way, evaluate the usability of HMI development tools. This could provide guidance for developers who are standing in front of the choice of which HMI development tool to use. It is not efficient to try out each development tool by yourself, because of time and costs of licenses. The table with results from the evaluation of HMI development tools used in this study could give the developers a chance to compare the development tools. This can help them to make a well-informed dissension on which tool to use, depending on how well it matches their needs.

Chapter 5
Conclusions and future work

The study highlights the HMI development process and identifies criteria which may be important for the development tool to fulfil to make it easier to develop high quality interfaces.

In the study we also performed a case study to evaluate two HMI development tools based on the selected evaluation criteria. This were presented in structured way which could be useful for making equal comparisons of development tools. It is important that the development team chooses a development tool based on their needs and it is therefore not possible to say that one development tool is always the best.

5.1 Answers to research questions

- RQ1: How does the current process of HMI development for industrial robotics look on Yaskawa Nordic AB?
  This question is answered by the results of the interview study, presented in section 2.2.2 The employees say that the HMI development is lacking in standardisation, both in the development tools used and the look of the final product. This is a big issue for the effectiveness of the development.

- RQ1.1: In what way may the HMI development process be improved to shorten development time and required programming knowledge of the HMI developer?
  The unified answer from the developers at the studied company were that one HMI development tool has to be chosen. All their thoughts on possible improvements are presented in section 2.2.4 This is further discussed in the analysis of the same chapter, which highlights five important criteria for a HMI development tools which has to be taken into consideration. It is important to choose a development tool which fits the developers knowledge.

- RQ2: What is the applicability of the selected HMI framework to the current HMI needs?
  In section 3.3 the experiences from two case studies are mapped to the qualifications that are important for HMI development. In this study, we have

tested how well Qt and iX Developer meets the needs of a HMI development tool. It shows that Qt is very versatile but it requires good programming knowledge to use it. iX Developer on the other hand is very simple to use and it is specialised for HMI development, which makes it easy to develop standard HMI applications. However, this can also be its limitation since it is more strict and much of the control is taken away from the programmer.

- RQ2.1: What are the experiences from applying a HMI framework matching the current HMI needs?
  The results from the case study of the two selected development toolkits are presented in section 3.2. The experiences from developing the same application in two different toolkits are described. The experiences gained are mapped to the identified criteria.
- RQ2.2: What are the pros and cons of the selected HMI framework applied to the current needs?
  The pros and cons of the selected toolkits are presented and discussed in section 3.3. The evaluated qualities, like the possibilities to reuse code and make templates, the amount of coding and coding knowledge and how easy the software is to use are discussed.
- RQ2.3: What improvements to the HMI framework can be made?
  No bigger improvements can be made directly to the development tools available. One option is to make a custom development toolkit which is tailored for the special needs of HMI development 4.7. This can streamline the development so that important tasks are made easier, but it also puts the burden of developing and updating the toolkit on the development team.

5.2 Future work

Because of time limitations, only a few of the development toolkits available today could be examined. In a future work, we would like to conduct a bigger case study with a wider range of toolkits. The test applications could also be more advanced to give more insight in the pros and cons of the toolkit. This would result in a better overview of the usabilities of the toolkits and the table that we started filling in could be extended. This could be helpful information for companies in the situation where the development has to be more standardised and a choice about which development tookit to use has to be made.

The results from the study shows that a custom made development toolkit could be useful. The toolkits available does always not fully meet all the requirements that are important for companies developing HMI applications. This suggest that a custom made toolkit, which is developed in close collaboration with the industrial needs could fill the gaps identified. The pros and cons of developing such a toolkit has to be further researched to ensure that the project is giving more in return to that company than it costs to develop.

We would also like to do future research in how to make reusable templates for HMI applications. These would focus on creating a standard look and feel for the products of the company and also reduce the development time that is spent by programmers designing the HMI layout over and over again.

Appendix A

Information om undersökning i syfte att utvärdera utvecklingen av HMI för robotar på Yaskawa

Jag genomför mitt examensarbete på Blekinge Tekniska Högskolan i samarbete med Yaskawa. Målet är att utvärdera hur HMI-utvecklingen ser ut idag och komma med förslag på hur det skulle kunna förbättras. Jag är därför intresserad av att få veta hur utvecklingsprocessen har sett ut i de senaste HMI-projekten du deltagit i. Jag vill även höra vilka idéer och tankar du har på vad som var problematiskt under utvecklingen och hur det skulle kunna förbättras.
Din identitet kommer inte att redovisas i resultatet. Det är helt frivilligt att delta i intervjun och du kan när som helst avbryta den. Jag skulle gärna se att jag fick spela in intervjun. Ljudinspelningen kommer användas av mig (och ingen annan) för att underlätta sammanställning av resultaten. När jag är klar kommer inspelningen att raderas. Resultaten kommer att rapporteras i en sammanställning i mitt arbete och användas för att överblicka HMI-utvecklingen på Yaskawa idag.

Om du har några frågor innan eller efter intervjun kan du kontakta mig via min mail lindaandersson92@hotmail.com eller ringa till 076 2477 203.

Under intervjun kommer jag ställa följande frågor:

- Hur länge har du arbetat på Yaskawa?
- Vilken avdelning tillhör du och vilken är din position där?
- Vilket är ditt huvudsakliga arbetsområde?

- I hur många projekt har du deltagit i HMI-utvecklingen?
- Vilket var det senaste HMI-projektet du var med på? (De följande frågor besvaras för det här projektet, alternativt om du har deltagit i flera: välj ut de två som skiljer sig mest med avseende på utvecklingsmiljön)
- Under vilken tidsperiod pågick projektet?

- Jobbade du ensam eller i ett team?
- Vilka HMI-plattformar omfattade projektet?
- Vilken utvecklingsmiljö användes för HMI-designen?
- Hade du tidigare erfarenhet från den utvecklingsmiljön? Om inte, hur långt tid tog det att lära sig?
- Var det enkelt att arbeta med den utvecklingsmiljön?
- Om du har arbetat i flera olika miljöer, vilken föredrar du? Varför?
- Lades det till någon ny funktionalitet till under projektet? Hur påverkade det HMI-utvecklingen?
- Om HMI-projektet omfattade flera plattformar (teach box/PLC/PC), skilde sig något i utvecklingsprocessen för dessa plattformar? Fick man göra några specialanpassningar?
- Återanvändes delar från tidigare projekt? Varför/varför inte? Vad återanvändes?
- Tänkte ni på återanvändbarhet för framtida projekt då ni jobbade med det här projektet? Om ja, hur påverkade det projektet?
- Fanns det en kodstandard under projektet? Om ja, hur följdes den?
- Användes några design mallar (patterns) eller design principer för att strukturera programkoden? Om ja, vilka och hur påverkade det projektet?
- Vad skulle man kunna göra för att öka återanvändbarheten av komponenter för HMI-utveckling?
- Hur skulle man kunna göra det lättare att designa ett användarvänligt HMI?
- Vad, tror du, är den viktigaste förändringen som bör göras för att förbättra HMI-utvecklingen på Yaskawa?
- Är det något annat du tror skulle vara intressant för mitt arbete?

Det var alla mina frågor. Jag vill tacka så mycket för att du tog dig tid att besvara dem. Har du några frågor till mig eller övriga kommentarer innan vi avslutar?

Acknowledgements

I would like to thank Mikael Svahnberg for his supervision and advices which guided the project in the right direction.

The study were done on behalf of Yaskawa Nordic AB, and I would like to thank them for a great collaboration and for the time they invested in my interviews .

I would also like to thank everyone who helped correcting the English and kept my mood up during the project. It would not have been possible to complete the thesis if you did not cheer me up and kept pushing me.

References

[1] J. Katzel, "HMIs in robotics. (cover story)," *Control Engineering*, vol. 51, pp. 30–34, Oct. 2004.

[2] R. Schilling and R. a. Norvig, "Fundamentals of robotics," Jan. 2013. Contents related to RAI.

[3] L. Erickson, "Programmable logic controllers," *IEEE Potentials*, vol. 15, pp. 14–17, Feb. 1996.

[4] W. Bolton, *Programmable Logic Controllers.* Newnes, Sept. 2009.

[5] B. Myers, "Challenges of HCI design and implementation," *interactions*, vol. 1, p. 73–83, Jan. 1994.

[6] H. Berger, *Automating with SIMATIC: Controllers, Software, Programming, Data.* John Wiley & Sons, Oct. 2012.

[7] M. Fowler, "Separating user interface code," *IEEE Software*, vol. 18, pp. 96–97, Mar. 2001.

[8] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, A System of Patterns.* Wiley, Apr. 2013.

[9] P. Falkman, E. Helander, and M. Andersson, "Automatic generation: A way of ensuring PLC and HMI standards," in *2011 IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–4, Sept. 2011.

[10] W. B. Frakes and K. Kang, "Software reuse research: Status and future," *IEEE transactions on Software Engineering*, vol. 31, no. 7, p. 529–536, 2005.

[11] L. T. Eriksson and F. Wiedersheim-Paul, *Att utreda, forska och rapportera.* Liber, 9 ed., 2011. Att utreda, forska och rapportera.

[12] J. W. Creswell, *Research design : qualitative, quantitative, and mixed methods approaches.* Thousand Oaks: Sage, 2003.

[13] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering.* Springer, June 2012.