

Intuitive Programmierung von kollaborativen Mensch-Roboter Prozessen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Bernd Hader, BSc

Matrikelnummer 01427748

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dr.-Ing. Sebastian Schlund

Mitwirkung: Christina Schmidbauer, MSc

Dr.-Ing. Tudor Ionescu, MSc, MA

Wien, 10. März 2021

Bernd Hader

Sebastian Schlund



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Intuitive Programming of collaborative Human-Robot Processes

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Bernd Hader, BSc

Registration Number 01427748

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dr.-Ing. Sebastian Schlund

Assistance: Christina Schmidbauer, MSc

Dr.-Ing. Tudor Ionescu, MSc, MA

Vienna, 10th March, 2021

Bernd Hader

Sebastian Schlund



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Bernd Hader, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. März 2021

Bernd Hader



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich während der Anfertigung dieser Diplomarbeit fachlich sowie persönlich unterstützt und motiviert haben.

Ein besonderer Dank gebührt meinen Betreuern Univ.-Prof. Dr.-Ing. Sebastian Schlund, Christina Schmidbauer, MSc und Dr.-Ing. Tudor Ionescu, MSc, MA. Sie hatten immer ein offenes Ohr und unterstützten mich mit hilfreiche Anregungen bei der Erstellung dieser Arbeit. Außerdem möchte ich mich für das wertvolle Feedback und die Ideen bedanken.

Weiters danke ich Bettina Mosch, BA für das Korrekturlesen meiner Diplomarbeit.

Meinen Eltern, Renate und Gerhard, sowie meinen Geschwistern, Judith und Gerd, danke ich, dass sie mich im Laufe des Studiums unterstützt haben.

Zu guter Letzt ein herzliches Dankeschön an meine Freundin Claudia für die motivierenden Worte sowie ihre Geduld in den letzten Jahren.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Seit der Entwicklung der ersten kollaborativen Roboter, kurz Cobots, in den 90er Jahren gab es hohe Erwartungen, dass diese neue Generation von Robotern die Arbeitssituation ganzer Industrien grundlegend verändern könnte. Tatsächlich konnten jedoch diese Erwartungen bis heute nicht erfüllt werden. Hierfür gibt es eine Reihe von Gründen, ein großes Problem sind nach wie vor die Programmierumgebungen. Obwohl Cobots dafür vorgesehen sind, mit dem Menschen kollaborativ an Prozessen zu arbeiten, berücksichtigen aktuelle Benutzerschnittstellen den Menschen nicht und konzentrieren sich nur auf die Programmierung des Roboters. Das Ziel der Arbeit ist es, eine workflow-basierte Programmierumgebung zu schaffen, welche die beiden unabhängigen Bereiche der Cobot Programmierung und der Arbeitsassistenzsysteme in einem System vereint. Dadurch soll die Möglichkeit geschaffen werden, dass Menschen mit geringen oder keinen Programmierkenntnissen auf einfache Weise kollaborative Mensch-Roboter Prozesse erstellen können. Diese erstellten Workflows sollen in weiterer Folge sowohl den Cobot steuern als auch dem Menschen als Werkerassistenzsystem dienen. Für die Umsetzung der Diplomarbeit wird ein benutzerzentrierter Ansatz verwendet, damit die Anforderungen der Benutzer bereits während der Entwicklung berücksichtigt und somit bestmöglich integriert werden können.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Since the rise of the first collaborative robots, short cobots, in the 90s, there were really high expectations, that those robots will change working situations of whole industries fundamentally. In fact, that didn't happen so far. One major problem are their programming environments. Although cobots are intended to work together with humans in a shared working space, current programming interfaces don't consider the human being in their process. The aim of the work is to create a workflow-based system, which combines the two independent areas of cobot programming and worker assistance systems into one system. This should enable people with little or no programming skills to create collaborative human-robot processes, which control the cobot and serve simultaneously as a worker assistance system. A user-centred approach is used in order to meet the user requirements as well as possible.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Motivation & Problem Statement	1
1.2 Expected Results	3
1.3 Methodology & Approach	4
1.3.1 Understand and specify context of use	5
1.3.2 Specify the user requirements	6
1.3.3 Produce solution	7
1.3.4 Evaluation	7
1.4 Structure of the Work	9
2 Theoretical Foundations	11
2.1 Robotics	12
2.1.1 Historical Evolution	12
2.1.2 Industrial Robots	12
2.1.3 Collaborative Robots (Cobots)	17
2.1.4 Worker Assistance Systems	24
2.2 Human-Computer Interaction	28
2.2.1 Graphical User Interfaces	29
2.2.2 Intuitive Use - Usability	30
2.2.3 User Interface Design, User Experience Design and Mental Models	32
2.3 (Business) Process Modelling	34
2.3.1 What is a process?	34
2.3.2 Types of process modelling	35
2.3.3 Business Process Model and Notation (BPMN)	36
2.3.4 Reasons for the use of BPMN 2.0 in the context of the thesis	41
3 State of the Art	43
3.1 Programming Robots/Cobots	44

3.1.1	Types of Robot Programming	44
3.1.2	Programming Robots/Cobots with Graphical User Interfaces (GUIs)	47
3.2	Systematic Literature Review	50
3.2.1	Summary State of the Art	53
3.3	Cobot programming in practice	74
3.3.1	Universal Robots	74
3.3.2	Kuka	75
3.3.3	Fanuc	76
3.3.4	Fruitcore Robotics	77
3.3.5	Franka Emika	77
3.3.6	Techman (TM) Robot	78
3.3.7	ABB	79
3.3.8	Drag&Bot	80
3.3.9	Robot Operating System (ROS)	80
3.3.10	Conclusion	81
4	Implementation	83
4.1	Iteration 1	84
4.1.1	Understand and specify context	84
4.1.2	Specify the user requirements	84
4.1.3	Produce solution	88
4.1.4	Evaluation	92
4.2	Iteration 2	93
4.2.1	Understand and specify context	93
4.2.2	Specify the user requirements	93
4.2.3	Produce solution	94
4.2.4	Evaluation	96
5	Software Architecture	97
5.1	(Camunda) BPMN Engine	97
5.1.1	Interaction with the BPMN Process Engine	100
5.1.2	Camunda REST API [171]	102
5.1.3	Conclusion	108
5.2	Franka Emika Panda Cobot	109
5.2.1	Authentication	110
5.2.2	Start a Task	110
5.2.3	Get Status of a Task	111
5.3	External Node JS Task Client	112
5.4	Web-based User Interface	115
5.4.1	What is bpmn.io?	115
5.4.2	Req 1: Creation of collaborative BPMN processes	116
5.4.3	Req 2: Definition of additional parameters (name of the robot tasks, which should be executed)	117

5.4.4	Req 3: Storage of additional information (work instructions as image and text)	118
5.4.5	Req 4: Execution of the created processes (start/stop)	118
5.4.6	Req 5: Representation of the current progress	119
5.4.7	Req 6: Representation of additional information (work instructions) during execution	120
5.4.8	Req 7: Detailed representation of robot tasks	120
5.4.9	Req 8: User interaction (confirmation that the user has completed a specific task)	120
5.5	Code	121
6	Evaluation	123
6.1	Study Design	123
6.1.1	Hypothesis	124
6.1.2	Study Procedure	124
6.1.3	Further Evaluation Data	130
6.1.4	Technical Implementation	130
6.2	Participants	131
6.2.1	Data Cleansing	131
6.2.2	Demographic Data	132
6.3	Results	135
6.3.1	Task solved correctly, Duration	135
6.3.2	Usability	136
6.3.3	Further Results	137
6.3.4	Qualitative Data (Comments and Issues)	138
7	Conclusion, Discussion & Outlook	139
7.1	Conclusion	139
7.2	Discussion & Outlook	142
A	Focus Group Invitation	143
B	Questionnaire	145
	List of Figures	151
	Bibliography	155



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Introduction

1.1 Motivation & Problem Statement

Human-Robot-Interaction has become a popular topic in the media, industry and research. It is usually associated with the terms Industry 4.0 and collaborative robots, short cobots. Collaborative processes between humans and robots represent the highest level of Human-Robot-Interaction, where humans and cobots work together in a shared working space to achieve a common goal [1, p.4-6], [2, pp.8]. Automation by the use of robots was only of interest for large companies in the last few decades. This should change with the rise of cobots in the 90s [3]. Due to their flexible application areas and their low price compared to classical industrial robots, there were really high expectations, that cobots will change the working situations fundamentally especially in manufacturing for small and medium-sized companies [2, pp.12]. Despite the high media attention and the low prices of cobots starting below €20,000, they are still a niche product. In the World Robotics Report 2020 of the International Federation of Robotics (IFR) cobots had only a share of 4.8% [4]. This shows, that in reality those high expectations have not been met by far.

There are many possible reasons why there is such a large discrepancy between expectation and reality. One problem is the programming of collaborative robots. The flexible application possibilities should allow cobots to be used weekly, monthly or even daily as needed for other purposes. This requires a simple programming interface that enables people with little or no programming knowledge to program collaborative robots quickly and easily. According to a study by TM Robotics, simple programming is of paramount importance for 79.31% of the customers [5].

The manufacturers promise intuitive programming environments, but there are some problems. On the one hand, these programming environments such as Desk by Franka

Emika [6] lack essential elements such as if-/else conditions and loops. These constructs are necessary to program common tasks and to enable a dynamic division of tasks between human and machine [7]. On the other hand, these programming environments only offer the possibility to program the tasks of the robot. The idea of cobots is to support the worker, but not to replace him. For this reason it would be also important to be able to model the tasks of the worker. With this approach it would be possible to combine the programming of the cobot with the creation of a worker assistance system. A worker assistance system is a system that guides the worker through a process and provides him with necessary information. The current cobot programming only refers exclusively to the programming of the robot, but completely ignores the human being.

Therefore, the thesis addresses the following research questions:

RQ1: What does an intuitive robot programming interface look like that can simultaneously serve as a worker assistance system?

RQ2: What is an appropriate means to implement such a system?

Some parts of this work, especially the user interface of the created prototype (see Chapter 4 & Chapter 5), were published in advance in [7].

1.2 Expected Results

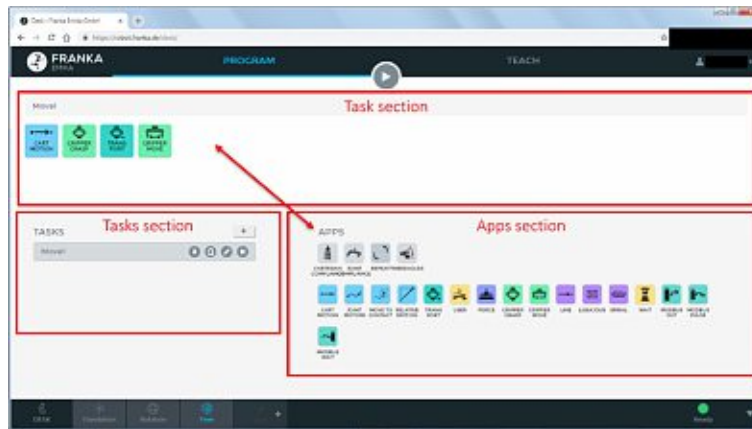


Figure 1.1: Programming interface of cobot Panda by Franka Emika [8]

The programming environment "Desk" of Franka Emika's cobot Panda (see Figure 1.1) appears very simple at first glance. With this kind of programming it is possible to program the cobot. However, it is not possible to create entire collaborative human-robot processes. The problem is that there is no way to integrate tasks of the worker into the workflow. Therefore, a possibility should be created both to programme the robot and to map entire collaborative human-robot processes. Furthermore, it should be possible to enrich the human tasks with further information. For this purpose, the idea was to create an intuitive programming interface based on a process modelling language, as shown in Figure 1.2.

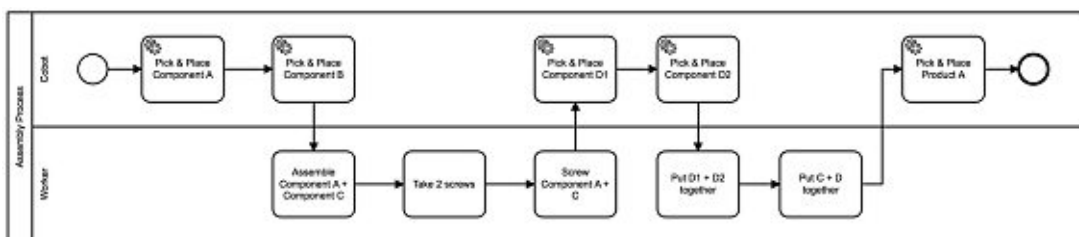


Figure 1.2: Collaborative Human-Robot Assembly Process with BPMN 2.0 (own Figure)

Summarized, the expected result should be an intuitive programming interface based on a process modelling language, which

- is easy to use, also for people with no or less programming experience,

- maps the entire collaborative process,
- controls the robot,
- provides the possibility to enrich the human tasks with additional information
- and serves simultaneously as a worker assistance system.

The results should also show to what extent a combination of cobot programming with existing process modelling tools is useful and which limitations exist. As usability and intuitiveness are central elements of the thesis, the evaluation should show to what extent this approach is considered as simple and intuitive for the users.

1.3 Methodology & Approach

As the title of the thesis already states, intuitiveness and usability are central elements. The aim of the work is to create an intuitive programming environment for human-robot processes. Since usability always depends on the context, the specific users, which are using the artifact, play an important role [9, p.9]. In order to integrate the users as an essential element, a user-centered approach seems to be most obvious and reasonable to create such an artifact.

User-Centered Design (UCD) is a methodology, which should ensure, that the created product has a good usability, which satisfies the requirements of the users. Therefore, it is important that the users of the developed product are integrated in all steps and that there is a so-called feedback loop [10, pp.89]. In the literature there are various process models which implement the basic concept of UCD. Examples are the Usability Engineering Life Cycle by Deborah Mayhew [11], the Goal Directed Design by Alan Cooper et al. [12], Contextual Design by Hugh Beyer and Karen Holtzblatt [13] and many more.

A very widely used User-Centered Design approach for the development of interactive systems is defined in the ISO standard 9241-210, which is applied in this thesis [14].

As Figure 1.3 shows, the ISO approach consists of the four main elements: understand and specify context, specify user requirements, produce solution and evaluate. The process steps are run through iteratively several times as required [14].

The next section provides a brief explanation of each of the four phases as well as a detailed description of the research methods used.

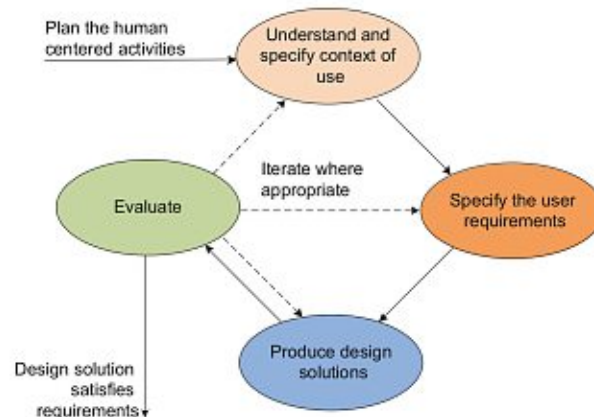


Figure 1.3: ISO standard 9241- 210, User-Centered Design process for interactive systems [10, p.15]

1.3.1 Understand and specify context of use

Systematic Literature Review

In order to obtain the necessary knowledge for the field, it is necessary to understand the basic knowledge about the different topics. A systematic literature review (SLR) will be carried out to summarise the existing literature to obtain knowledge about the state of the art [15, p.3].

The SLR is conducted on the basis of the review process by Kitchenham, which consists of the following three steps [15, p.6].

- **Planning the review**

The research question addressed by this thesis is:

What does an intuitive robot programming interface look like that can simultaneously serve as a worker assistance system?

The research question consists of the following areas: intuitive programming, robot programming and worker assistance system. Furthermore, the following research areas are of interest: human-machine interaction, collaborative processes, collaborative robots, workflow modelling.

As inclusion and exclusion criteria there are the following restrictions. Cobots have only been around since the end of the 90s [3]. For this reason, only papers published after 1996 are of interest. Furthermore, the main topic is programming. Papers which do not have programming as content will be dropped. There are different approaches to enable intuitive programming like learning from demonstration, skill based programming, and so on. The content of the paper addresses intuitive programming with graphical user interfaces, so all papers with other approaches

will be dropped. Since there has been lots of research in the area of cobots in the German-speaking area over the last years, both German and English literature will be considered. The search is performed manually. Since a good SLR protocol is fundamental for the result of the whole literature review, an evaluation should take place as recommended by Kitchenham [15, p.13].

- **Conducting the review**

The databases used for the search are IEEE Xplore [16], ScienceDirect [17], Springer-Link [18], Google Scholar [19], Citeseer Library [20], ACM Digital Library [21] and Sage Journals [22].

- As described by Kitchenham [15, pp.14], in this step queries are created using the keywords defined in step 1 in combination with different operators like AND, OR, NOT. One possible example would be: (COBOTS OR COLLABORATIVE ROBOTS) AND (PROGRAMMING) AND (WORKFLOW).

- **Reporting the review**

Based on the SLR, a report including the summarized content will be prepared, which will be part of the thesis.

1.3.2 Specify the user requirements

Before the user requirements can be specified, it is necessary to determine the potential users of the system [23, p.117]. The following three groups of stakeholders have been identified. The workers who will program cobots in industry in the future. Researchers who develop these systems and provide the necessary knowledge. The third group consists of interested persons who use so-called Makerspaces or Fablabs to realize their own projects.

The project Cobots Meets Makerspace, short CoMeMak, is a research project co-financed by the Austrian Research Promotion Agency and lead by the TU Wien. Besides TU Wien, an Austrian Makerspace (Grand Garage), a research institute (Joanneum Research) and a manufacturing cooperate (BRP Rotax) are working together towards the project goal. The idea of the CoMeMak project is to give interested people access to cobots in makerspaces and to address various problems regarding robots, intuitive programming and robot safety [24].

Focus Group Interviews

To get the requirements of the potential stakeholders and to develop an appropriate prototype in an iterative way so-called focus group interviews with the users from the groups above are conducted. The interview is led by a moderator [25, p. 10]. To avoid an unstructured discussion, a previously developed guideline is used as a basis [25, p. 18]. This guide is intended to provide a certain structure but should still allow an open discussion within the group [25, p. 18]. The great advantage of interviews with several participants is the possibility that new ideas/solutions can emerge, which would otherwise

remain hidden [26, p.12]. Furthermore, this qualitative research method enables the collection of individual opinions of the participants as well as group statements [25, p. 11]. To obtain appropriate results, the group size should not be too small, but to ensure that each participant has a certain amount of speaking time, the group should also not be too large. Usually a group size of 6-12 people is recommended [26, p.12], [25, p. 10].

1.3.3 Produce solution

Agile development

Agile software development describes a method of software development, which flexibly responds to the requirements of the customer through iteration and short cycles. Only small parts of the software are developed and tested afterwards [10, p.13].

To combine high usability with flexible software development, it makes sense to combine both approaches (user-centered design and agile methods). One possible approach is described by Silva et al. [27].

1.3.4 Evaluation

Functional Testing

With functional blackbox testing it can be evaluated whether the requirements are fulfilled or not [28]. This is very important, because a working software is the basis for a good usability.

System Usability Scale (SUS)

The System Usability Scale, short SUS, is a quantitative research method developed by John Brooke [29]. Usually the evaluation of the usability of a system depends on the context. Therefore, it would be necessary to define specific measures to test the usability for a specific context. This is costly and prevents the comparison of the usability of different systems. For this reason John Brooke developed a quantitative method to test the usability independent of the context and the used technology. The test consists of a questionnaire with 10 short questions, which are evaluated using a Likert scale. The questions have a different weighting, which results in a overall score between 0 and 100 [29].

A quantitative analysis of the usability of a system only makes sense after a functional prototype has been created. As soon as a functional prototype has been developed, this evaluation will be carried out. The result of this evaluation will be an indicator of how intuitive the system is and whether further iterations should be carried out or not.

User Experiment

At the end of the thesis the system will be evaluated by conducting a user experiment. The experiment will be performed as follows. The participants are given a task to program

1. INTRODUCTION

a specific collaborative process. Afterwards, they are asked to fill out a questionnaire based on their experience with the system. The questionnaire will consist of state of the art methods for evaluating the usability of software such as the SUS questionnaire [29] and NASA-TLX [30] as well as some open questions. The results will then be used to determine the user-friendliness and problems/limitations of this new interface.

1.4 Structure of the Work

Chapter 2 Theoretical Foundations explains the main subject areas and basic concepts, which are relevant for the work. As the title of the thesis "Intuitive Programming of collaborative human-robot processes" already suggests, this roughly covers the subject areas:

- **Robotics**
- **Human-computer interaction** and
- **Process modelling.**

Additionally the concept of intuitiveness is also of great importance for the work.

First of all there will be an introduction to the discipline of Robotics, starting with a short historical excursion to the emergence of robots. Then we will deal with the topics of industrial robots, collaborative robots and worker assistance systems.

After clarifying these terms we will focus on the topic of human-computer interaction (HCI). Human-computer interaction is a very broad field, which deals with interfaces, how people can communicate with computers. Since the goal of the thesis is to create a graphical programming environment, the focus is on graphical user interfaces (GUIs). After a general description of graphical user interfaces, the meaning of the term "intuitive" will be clarified, especially for the context of the thesis. For this purpose, we first consider the concept of intuitiveness in general. Then we will look at intuitiveness in relation to graphical user interfaces. What characteristics must a graphical user interface have in order to be considered as intuitive? How do you ensure good usability? To understand intuitiveness and usability in the context of graphical user interfaces an excursion to the following topics is necessary: **User Interface Design**, **User Experience Design** and **Mental Models**.

Since the mapping of the entire human-robot process is of interest, the last section of this chapter deals with process modelling. After a general introduction to this area follows a short description of the basic elements of the Business Process Model and Notation language (BPMN). Finally, it is explained why BPMN was used in the course of the thesis.

In **Chapter 3** a systematic literature review is conducted to determine the state of the art in cobot programming. Since collaborative robots have been an issue since the 1990s, there are already various solutions from different manufacturers on the market, which we will also look at in detail.

Chapter 4 is devoted to practical implementation. With the help of the user-centred design approach, a prototype is created iteratively in several runs with the involvement of potential users. The basis for the implementation is formed by the theoretical foundations (see Chapter 2), the systematic literature review (see Chapter 3) and group interviews with experts.

In **Chapter 5** we look at the technical part of the implementation. The software architecture as well as important parts of the implementation of the prototype are described here in detail.

Chapter 6 deals with the evaluation of the prototype. The basis for this is a user experiment, which is conducted online. The participants have to solve a specific task with the created web interface. A questionnaire, based on state of the art methods for determining the user-friendliness of a system like the SUS questionnaire and NASA-TLX, will be used for the evaluation. This questionnaire in combination with other user data like the execution time and the correctness of the solution will then provide results on how far the prototype meets the requirements of intuitiveness and usability.

In **Chapter 7** we look at the research questions and answer them with the help of the findings and results of the thesis.

Chapter 8 summarizes the problem statement and the results of the thesis. Finally, an outlook on future work in this field is given.

Theoretical Foundations

In this chapter the essential basics are explained, which are needed as background knowledge for the work. Since the topic of the thesis deals with the improvement of the programming of collaborative robots and the use of worker assistance systems, the three subject areas **Robotics**, **Human-Computer Interaction** and **Process Modelling** are of interest.

Robotics

In this section, first there will be a short explanation of the differences between classical industrial robots and collaborative robots. Furthermore, the different types of collaboration are explained. The different methods how cobots can be programmed are not part of this chapter, but are described in detail in Chapter 3 State of the Art.

From the field of robotics we are also interested in the topic of assistance systems. Possible types of worker assistance systems and their current use in industry are considered briefly.

Human-Computer Interaction

The field of Human-Computer Interaction (HCI) is an area that has become more and more important in almost all areas of human life in recent decades due to increasing digitalisation. It is a very broad field, but we are specifically interested in the possibility of HCI via Graphical User Interfaces (GUIs). Here we are interested in the possibilities for designing user interfaces that are especially good and easy to use, or in other words "intuitive". For this we will first try to clarify what intuitiveness means and then we will dive into the areas of User Interface Design, User Experience Design and Mental Models.

Process Modelling

After a short introduction to the field of process modelling and the clarification of terms, we will look at different types of process modelling. Then an introduction to the Business Process Model and Notation Language (BPMN) follows as well as a short explanation why this process modelling languages was used for the development of the prototype.

2.1 Robotics

2.1.1 Historical Evolution

From a historical point of view it can be seen that long before the development of the first robot, this topic was already of great interest in the science fiction scene. An important contribution, regarding the naming, was made by the Czech author Karel Capek, who coined the term "robot" in his play "Rossum's Universal Robots" in 1921 [31, p.14]. Probably the word "robot" was derived from the Czech word "robota", which means "to work unfree". The play [32] is about the company "Rossum's Universal Robots", which uses a substance to create artificial humans (robots). Rossum's idea was to create cheap labor that would relieve people of heavy labor. This works quite well at the beginning and the robots serve the people. But at the end of the play there is a riot and the robots outlive the humans.

Two decades later, in the 1940s, the term "Robotics" was used for the first time by a science fiction author named Isaak Asimov. He defined "Robotics" as the study of robots, which deals with the design, implementation as well as the control and use of robots [33, [31, p.VII, p.15], [34, p.10].

Inspired by the first science fiction novels, in which robots were artificial humans, nowadays there are two meanings of the term robot in the everyday language [31, p.15], [35]:

- a machine based on the human being, which can execute human functions (artificial human)
- an automaton, which executes certain functions based on sensor data and/or programming, which are usually performed by humans (e.g. industrial robots in assembly, vacuum cleaning robots)

With the increasing spreading of robots in everyday life (e.g. vacuum cleaning robots, mowing robots) the first meaning, especially the shape, lost importance during the last decades and usually the term humanoid robot is nowadays used for human-like robots. Whereas the second meaning, that a robot takes over human tasks, regardless of its figure, becomes more important.

From an industrial point of view, two types of robots are particularly important for production, so-called classical industrial robots and collaborative robots, which will be discussed in detail below.

2.1.2 Industrial Robots

The first industrial robot, called "Unimate", was introduced in 1960 and was already used in production by General Motors (GM) in 1961. The two developers of the "Unimate" Engelberger and Devol are often referred to as the fathers of the industrial robot. Due to constant further developments, the falling price and the high benefit in production,

especially in the automotive industry, the industrial robots spread very fast. Typical applications are welding and pick&place tasks. Examples of welding show that a human being would not be able to produce a comparable welding seam in the same time and with such a high precision and a constant quality like an industrial robot. [36], p.11f]

This potential was already recognised at GM in the 1960s. By using industrial robots, the production time was halved in 1969, doubling the number of units produced. In figures, GM was able to increase the production to 110 cars per hour. For this reason (the enormous increase in production due to the usage of industrial robots for welding as well as pick&place tasks), other car manufacturers such as BMW and Daimler have also very quickly adopted this approach for their production lines. [37]

Already in 1986 Todd defined in the "Fundamentals of Robot Technology" [34] a list of characteristics, which a classical, or as Todd called it, a true robot, should have [34], p.9f]. Haun [31], p.15] revised this list and defined the following five elements as crucial for a classical (industrial) robot:

- an arm or gripper (to move objects, not mandatory)
- wheels, chains or something similar to move the position (not mandatory)
- sensors (e.g. for determining the position of the robot as well as the position of the gripper or for measuring distances to objects or persons, etc.)
- a power unit to move itself and/or to control the gripper
- a memory to store tasks/commands/applications that the robot should execute

With the help of these five elements, a classic robot according to Haun [31], p.16] as well as Todd [34], p.9] can perform the task for which it was initially created: to move objects to different places and, if it is a mobile robot, to be able to move itself.

Due to the development in manufacturing in recent years as a result of increasing digitalisation, keyword Industry 4.0, today's robots in industry require a number of additional capabilities, such as the ability to communicate [31], p.16]. On the one hand, this is important for classical industrial robots, so that the robots can communicate with each other and with different IT systems, which are used for planning and controlling the production. On the other hand, the ability to communicate is also very important with regard to collaborative robots. This is exactly where the work should contribute to the research by the creation and evaluation of a new user interface, which enables the user to easily communicate with the cobot by the use of a graphical user interface based on a process modelling language, but more on this later. First we look at further definitions of industrial robots and typical applications where they are used.

Definitions

In the literature various definitions for the term "Industrial Robot" can be found, in the following we look at two of them, which are particularly widespread.

The DIN EN ISO 10218-1 specifies the term industrial robot as an

"automatically controlled, reprogrammable multipurpose manipulator(s) (3.6), programmable in three or more axes (3.16), which can be either fixed in place or mobile for use in industrial automation applications" [38, p.4]

The Association of German Engineers (VDI) specifies in the VDI 2860 standard:

"Industrial robots are universal handling systems with several axes those motions with respect to movement sequence and paths or angles are freely programmable or sensor-guided. They can be equipped with grippers, tools or other means of production and can perform handling and / or production tasks." [39] [1] [2]

In summary, all three definitions (VDI 2860 [39], ISO 10218-1 [38], list of necessary characteristics of robots by Haun [31]) are basically the same, but with different levels of detail. They agree, however, that a typical industrial robot must consist of a gripper/manipulator, which must be able to be programmed/reprogrammed (Haun specifies this as the ability to communicate via e.g. user interfaces). Furthermore, the purpose is defined in the VDI and ISO standard with the adoption of typical applications in the industrial environment, e.g. handling or production tasks like pick and place or screwing tasks. Haun/Todd specify this in a more general way with the aim that the robot must be able to move objects.

Fields of Application and Current Developments

Typical examples of classical industrial robots in manufacturing can be seen in Figure 2.1 and 2.2, where the robots are used to move objects (typical pick and place tasks) as well as for welding. Figure 2.2 shows the production plant of Audi in Ingolstadt, where the assembly of the bodywork is completely automated.

¹Translated by [39]: "Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkel frei programmierbar oder sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen." [40].

²Note: The VDI norm 2860 has currently the status "withdrawn", but as the new revised version is not expected to be published before November 2021 [41], this (outdated) version can be regarded as the currently valid one.



Figure 2.1: Automated production using industrial robots [42]



Figure 2.2: Production line of the Audi A4 in Ingolstadt [43]

After the great success of industrial robots in the automotive industry in the 1970s, other industries also changed their production lines. There are two main reasons for this evolution:

- **Economic savings:** The main reason for the success of industrial robots initially in the automotive industry in the 1970s is based on the savings potential that results in comparison to manual work. Compared to humans, robots can carry out a task (e.g. welding) at a constant (high) speed with constant accuracy for up to 24 hours, 365 days a year. In addition, rising labour costs and falling costs for technology also contribute significantly. If we compare the development of labour costs e.g. in Germany in the last decades with the acquisition costs of industrial robots, it shows that the wage development is continuously increasing, while the acquisition costs of industrial robots are decreasing. This trend further increases the effect of the savings potential. [44, p.5f]
- **Flexibility:** Due to the design, based on the human arm, with several axes and the possibility of programming this arm, a classical industrial robot can be used for a wide range of applications and for different sectors such as the electronics and metal industry [44, p.4].

This flexibility makes it possible to adapt industrial robots to almost all use cases by means of programming. The only question that remains is whether a correspondingly high number of units can be produced so that the high investment costs are worthwhile. It must be taken into account that the high costs are usually not caused by the purchase of the robot itself. A typical industrial robot can be bought for about 20,000 € - 60,000 € [44, p.7]. The high costs result from the high integration effort, i.e. the programming, the adjustment of the end effector and the implementation of the safety concept [44, p.7]. Therefore, the actual costs are usually many times higher. As a guideline, the multiplication of the robot price by a factor of 4-5 is often used [44, p.7]. The actual

2. THEORETICAL FOUNDATIONS

acquisition costs of a 50,000 € robot, for example, amount to approx. 200,000 € - 250,000 €. Considering that an industrial robot usually only performs one very specific task and that normally (see Figure 2.1 and 2.2) a large number of robots are needed, the costs rise very quickly to millions of euros.

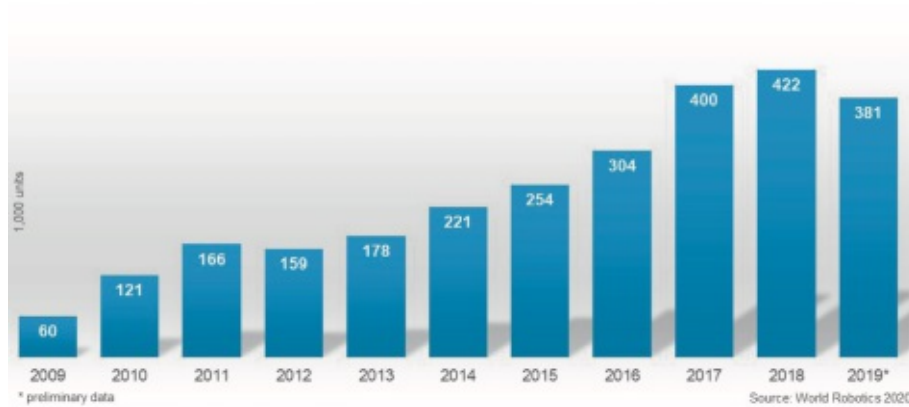


Figure 2.3: Annual installations of industrial robots 2009-2019 45

As shown in the World Robotics Report 2020 45 of the International Federation of Robotics (IFR) (see Figure 2.3), 381,000 industrial robots were installed worldwide in 2019. Furthermore, the development since 2012 shows very nicely the annual growth (between 2012 and 2018), which is on average about 12% per year. Asia is by far the largest customer, followed by Europe and America (see Figure 2.4). Looking at the application areas (see Figure 2.5), it shows that the automotive and electronics industries are still the major players, which account for about half of all robot installations.

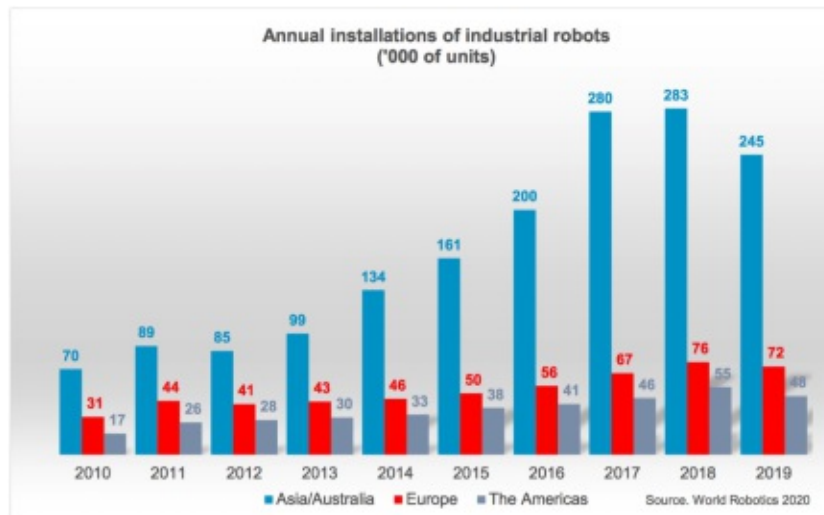


Figure 2.4: Annual installations of industrial robots by regions 46

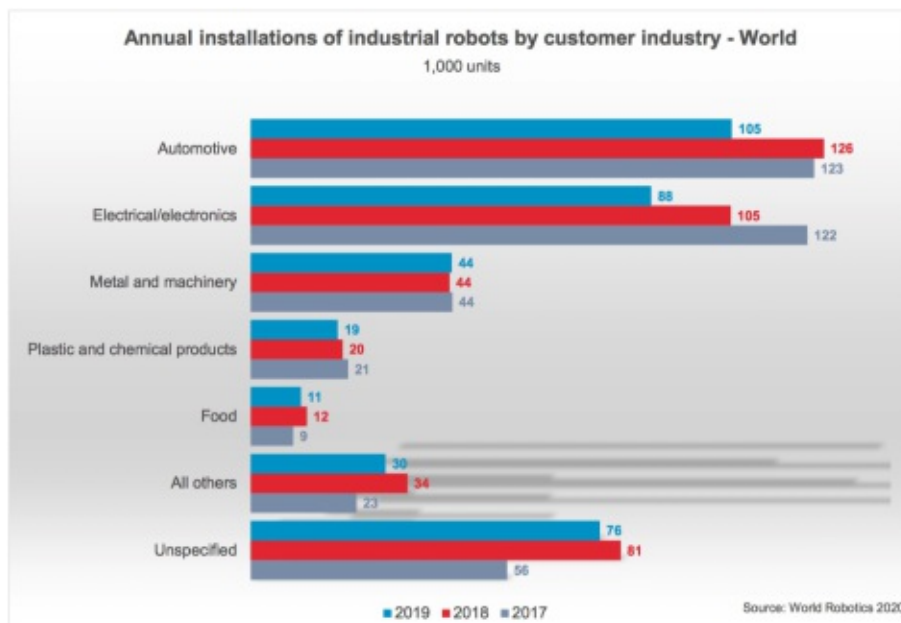


Figure 2.5: Annual installations of industrial robots by industries [46]

Due to the increasing individualisation towards batch size one in production and the fact that industrial robots are not of interest for small and medium sized companies, there was a need for a special type of industrial robot - the so-called collaborative robot was born.

2.1.3 Collaborative Robots (Cobots)

The term "Cobot", which is the abbreviation for collaborative robot, was coined in 1996 by Colgate et al. [3]. The original research project was funded by General Motors and aimed to find a way to make robots safe enough to work hand in hand with humans without the use of a safety fence, which is usually used for industrial robots [47].

Colgate et al. [3] define a cobot as follows:

"A "cobot" is a robotic device which manipulates objects in collaboration with a human operator." [3, p.433]

The original cobot has little in common with the collaborative robotic arms that are now used in the industry. The following graphics from the ISO standard 10218-1 show the difference between the design of a traditional industrial robot cell and a collaborative workplace very well.

Figure 2.6 shows a typical application of a classical industrial robot. For safety reasons, access within the working area of the robot is not permitted for humans. This is necessary

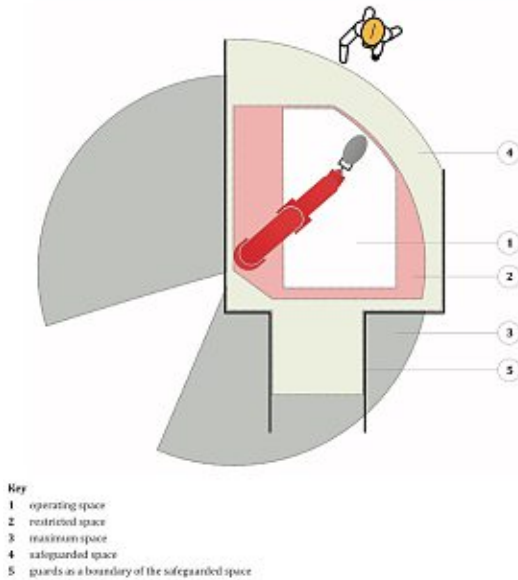


Figure 2.6: Typical cell of an industrial robot [38, p.42]

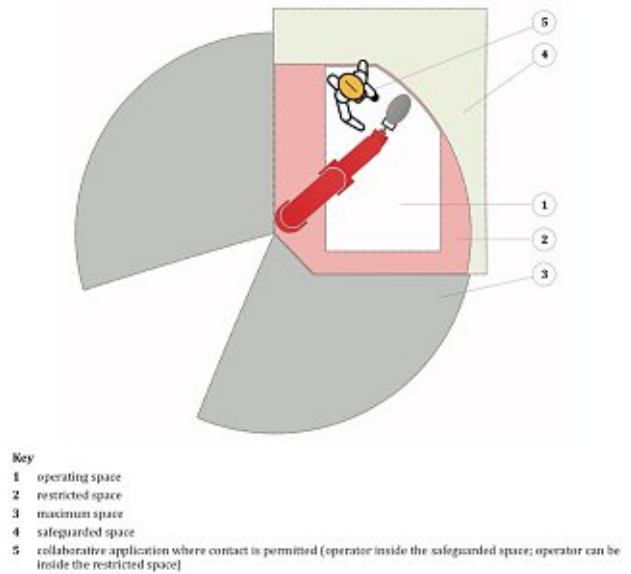


Figure 2.7: Design of a collaborative workplace [38, p.43]

because, on the one hand, traditional industrial robots work at a very high speed and on the other hand they are not equipped with sensors to avoid collisions. As you can see in Figure 2.6, the area numbered 4, shows the protected area, which extends beyond the maximum working area of the robot to ensure the safety of the human workers. The black lines represent a safety fence surrounding the robot to ensure that no human can accidentally enter the working area of the robot.

In comparison, Figure 2.7 shows a collaborative workplace, where humans and robots work hand in hand. As you can see, there are no guards and the human being and the cobot share their workspace.

So the main difference between a classical industrial robot and a collaborative robot is "the direct interaction with human workers" [48, p.14]. To enable this cooperation between human and robot special safety concepts are needed to prevent injuries.

Summarized this means that cobots are in principle industrial robots that work together with humans and to make this possible they must be equipped with special safety concepts to ensure a safe collaboration [49]. The ISO standard 15066 [50] provides a basis for this and therefore we will have a look at some parts in the next section to clarify how such collaborative use cases can be implemented in practice.

Types of Human-Robot-Interaction

In the last section we have clarified the definition of the term "Cobot", but what we have not yet considered is the question of what exactly does "collaboration" mean? In the

literature as well as in the media the term "cooperation" is often used instead. So what is the difference between collaboration and cooperation and are there also other types of collaboration? What distinctions exist?

Bauer et al. [2, p.8-10] have defined a classification based on the interaction between human and robot, where a distinction is made between the following five types:

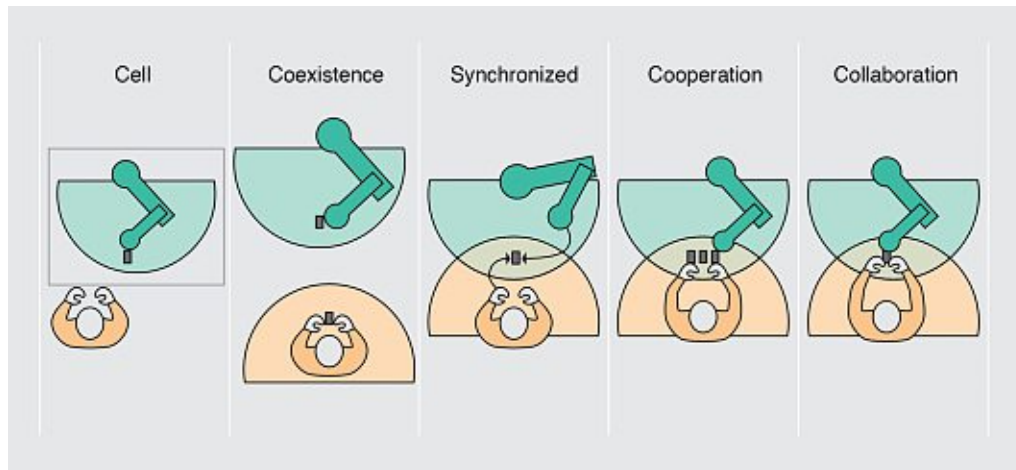


Figure 2.8: Types of Human-Robot-Interaction by Bauer et al. [2, p.9]

- **Cell:** In this scenario the robot and the human being work completely independent of each other. The example shows the scenario of a traditional industrial robot, where the robot is surrounded by a protective fence.
- **Coexistence:** This scenario is similar to that of the cell, with the difference that the robot has no protective fence and man and robot work side by side. This means that certain safety concepts are required from this level of interaction onwards, as the absence of guards allows the employee to enter the robot's working area. Both human and robot have their own working area and they work independently on their tasks.
- **Synchronized:** In this type of collaboration, human and robot share the workplace. It is important that only one actor is active in the workspace at one time, this can be either human or robot.
- **Cooperation:** The human being and the robot work in the same workspace as in type "Synchronized", with the difference that both actors work simultaneously on their tasks. However, the two actors work on different components.
- **Collaboration:** In this scenario, which represents the highest form of collaboration, the human and the robot share the workspace and they work on the same product at the same time.

In the classification by Bauer et al. [2] the focus is on the degree of collaboration between human and cobot. What is not taken into account, are the dependencies between the actors, e.g. in the scenario "Cooperation" the human and the robot work simultaneously on their components. It is not specified if there is a dependency between the worker and the cobot or not, both use cases are possible. The classification of Zaatari et al. [51] also takes the processes and the created work pieces into account and distinguishes between scenarios with and without dependency as well as if human and robot work on the same work piece or not.

The result is a classification with four categories, which essentially corresponds to the two types "Cooperation" and "Collaboration" by Bauer et al. [2] with additional differentiation whether a dependency between the processes exists and whether the two actors are working on the same work piece or not.

The four types of human-robot-collaboration by Zaatari et al. are [51]:

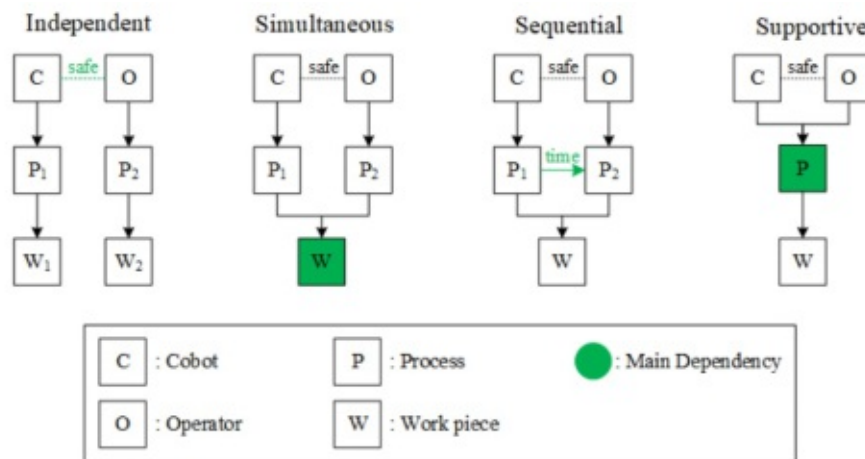


Figure 2.9: Types of Human-Robot-Collaboration by Zaatari et al. [51]

- **Independent:** The worker and the cobot share the workspace, but they work on different processes and workpieces. This scenario corresponds to the type "Cooperation" according to Bauer et al. [2].
- **Simultaneous:** Both human and cobot work on the same workpiece, but on different processes. According to Bauer et al. [2], this would also correspond to the category "Cooperation".
- **Sequential:** This scenario is similar to the category "Simultaneous", with the difference that there is a time dependency between the process P1 and P2. An example could be the assembly of a workpiece, where the cobot provides all the parts and the worker assembles them. If the robot does not deliver the necessary

parts on time, delays will occur. According to the classification of Bauer et al. [2] this would also correspond to the type "Cooperation".

- **Supportive:** This corresponds to the definition of "Collaboration" according to Bauer et al. [2]. The human and the robot share the workspace and work on the same process and workpiece at the same time.

A further classification, that deals with safety concepts for the implementation of collaborative systems, can be found in the ISO standard 15066 [50].

A distinction is made between the following four types [50]:

- **Safety-rated Monitored Stop:** In this type of collaboration the worker and the cobot share the same workspace, but only one actor is working at a time. A typical use case would be the loading of the robot's end effector by a human operator. The scenario corresponds to the type "Synchronized" described by Bauer et al. [2]. As soon as the worker enters the collaborative workspace, the system must detect this and stop the robot. Once the worker leaves the workspace, the robot can resume work. Since the robot only works when the operator is outside of the workspace, no collision between cobot and human can occur.
- **Hand Guiding:** In this scenario, the operator controls the robot by means of a manual controller, which is located directly on or near the robot. When a person enters the collaboration room a safety-rated monitored stop is triggered. Example: The operator enters the collaboration room, due to a safety-rated monitored stop the robot stops working. By using an input device e.g. teach-pendant, tablet or something similar, the operator can control the robot manually. Once the person has completed his tasks, the robot is stopped again by a safety-rated monitored stop. After the operator has left the collaborative workspace, the robot can continue the work.
- **Speed and Separation Monitoring:** This concept enables that human and robot can work simultaneously in the shared workspace by keeping a safety distance. The greater the safety distance between man and robot, the faster the robot can perform its activities. If the safety distance decreases, the robot has to reduce the speed. If the safety distance is less than the minimum safety distance, the robot has to stop. In respect to the classification of Bauer et al. [2] this concept allows the implementation of the scenario "Cooperation", where human and robot work in the same workspace but on different tasks.
- **Power and Force Limiting:** In this type of collaboration man and robot work hand in hand. To make this possible the robot must have special (precise) sensors that can detect collisions. Furthermore, these robots work usually with a lower speed, so that in case of a collision no injuries are caused. Typical collaborative robots, like the Panda by Franka Emika [6], which enable "real" collaboration

between human and robot as described in the scenario "Collaborative" according to Bauer et al. [2] or "Supportive" according to Zaatari et al. [51], are of this type.

Current Developments and Problems of Cobots

In the section "Industrial Robots" we have already seen statistics about the sales figures, which have shown a high growth in the last years. Looking at these figures in detail (see Figure 2.10), divided by the type of robot (classical industrial robots and collaborative robots), it can be seen that 355,000 traditional industrial robots and only 18,000 collaborative robots were sold in 2019 [52]. In percentage terms this means, that cobots had only a share of about 4.8% of all industrial robots sold in 2019.

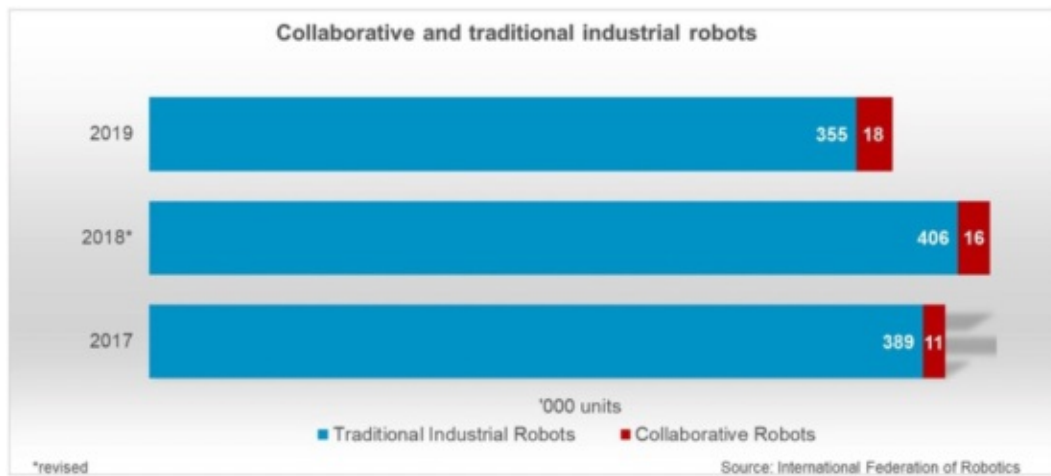


Figure 2.10: Sales of classical industrial robots vs. cobots (2017-2019) [52]

Compared to the development of the classic industrial robot (see section Industrial Robots), which became very widespread after only 10-20 years of invention, cobots were in 2019, about 25 years after their emergence, still a niche product [52], although their prices, starting at around €15,000-€20,000 [53], have fallen drastically in recent years. In theory cobots should be of particular interest for small and medium-sized companies, which currently still rely on manual assembly [2, p.12]. Before the rise of the cobots, the automation of such (manual) processes with small lot sizes was not economically meaningful [2, p.12]. This should have changed now, but the sales figures (see Figure 2.10) show the opposite.

There are several reasons for this. One major problem is that the actual costs (including the installation) are many times higher, than the €20,000. Similar to the integration of classical industrial robots, the real costs for implementing a cobot in an industrial setting are also about 4 times higher than the costs for the cobot itself [54, p.101]. One problem are the safety aspects, which have to be fulfilled. For the integration of a collaboration robot in e.g. a factory, depending on the country, it is necessary to certify each application [55]. The ISO standards 10218 and 15066 are often used for this purpose. In the course

of the certification, a risk assessment is necessary to evaluate dangerous situations [55], [50]. The problem is that every collaborative application (even if the same application has already been integrated in another company) has to go through this procedure again [56, p.93]. This results in a high effort and high costs.

Besides the problem with certification there are also issues with the "intuitive" programming interfaces of cobots. It turns out that these systems are often not as intuitive as they are advertised by the manufacturers. The evaluation of the usability of three programming environments in [57] showed that there is still considerable room for improvement. This is important so that no special experts are needed for programming. Thereby the economic efficiency can be increased, which is the primary goal of using cobots [2, p.9]. The importance of simple programming is also shown by a study of TM Robotics, which showed that simple programming is of paramount importance for 79.31% of the customers [5]. The reason for this seems obvious, because if it's possible to program cobots by non-experts, the programming sequences can be easily adapted, which will be necessary due to the increasing individualisation towards batch size one. Also interesting is the fact that human beings are not considered when programming cobots in current solutions of different manufacturers [58, p.4]. The focus is on programming the cobot, but more on this later in the section "State of the Art".

In summary, the current developments show that cobots have now reached a price level that should greatly increase their distribution. However, as the IFR report shows [4], collaborative robots are still a niche product. This is partly due to the high cost of integration and the current safety regulations, but there are also problems with the programming environments, as for example they do not take humans into account. This is exactly the problem that will be addressed in the course of the thesis.

2.1.4 Worker Assistance Systems

This section clarifies the basics of what a (worker) assistance system is, what it is used for and why it is also becoming more and more important in industry.

According to the dictionary, the term assistance system means: "electronic control system, which increases driving safety" [59]. In general, an assistance system is therefore an electronic system which is intended to support the human being in carrying out activities.

Assistance systems are often associated with driver assistance systems. This is because the first assistance systems (ABS - Anti-lock Braking System) were installed in cars in the 1970s to improve driving safety. Nowadays, vehicles are equipped with a variety of assistance systems such as lane keeping assistant, brake assist, etc. [60, p.1-2]

But assistance systems are also becoming increasingly common in other areas, such as in industry [60, p.7].

An assistance system in an industrial environment, also known as a worker assistance system, has the task of supporting the employee so that he/she can concentrate on his/her essential core competencies. For this purpose different technologies are used, e.g. tablets as input/output devices to capture input and provide information. However, such a system can also be expanded as desired depending on the use case, for example with cameras that automatically recognize the activities of the worker, sensors which can be used for locating positions, augmented reality glasses for the visualization of information instead of tablets, and so on. [61, p.90]

Particularly important for the acceptance of assistance systems is their operation [60, p.7], which we will discuss in detail in Chapter 2.2 Human-Computer Interaction.

As an example, in its simplest form, an assistance system in production could look like following: The user receives information (text/pictures/videos) about the current assembly step on a tablet and confirms that this step is completed via touch input. Then the next process step appears and so on. The input of the worker is necessary so that the right information is always available at the right time.

In comparison, a more complex implementation could look as follows: By means of a camera, it is automatically recognized which work step the employee is currently performing. In addition, the worker receives information on the current work step on a screen. The system also recognizes whether the step has been carried out correctly or not and visualizes this to the worker via light signals.

The Figures 2.11 and 2.12 show examples of worker assistance systems in practice. In the following we will take a closer look at the example shown in Figure 2.11. A screen is used to display the individual work steps and instructions using text and images. The worker can interact with the system via foot switch, touchscreen and the system automatically recognizes which components have been removed or deposited from the black boxes. In addition, a smartphone can be used to show augmented reality instructions directly on



Figure 2.11: Modular assistance system for manual assembly [62]



Figure 2.12: Worker Assistance System "Computer Aided Works" [63]

the components. Furthermore, the system is a modular system that can be expanded as required. [64]

Necessity of Assistance Systems in Manual Assembly

The reason for the necessity of assistance systems in manual assembly is that the change to even smaller batch sizes increases the number of variants. This means that a worker in manual assembly has to carry out a large number of different assembly processes. By increasing the number of variants, the mental effort increases. This in turn leads to lower productivity, but the decrease in productivity can be counteracted with the help of assistance systems. The employee can thus concentrate on his/her core competences and is relieved. [65, p.265]

Summarized the use of assistance systems in assembly can/should:

- help to avoid errors, which in sum increases the quality [61, p.91-93]
- reduce the mental and physical effort of the employees, which increases productivity [66, p.28], [61, p.91-93]
- shorten the training period of new employees [61, p.91-93].

To ensure that these positive effects emerge an essential factor is the acceptance of the employees [61, p.91-93]. Beyond that, it is important that the necessary information is presented in a suitable form at the right time, otherwise it can be misinterpreted, which increases the error rate and reduces productivity [66, p.24].

Based on two case studies, Hinrichsen and Bendzioch [67] have identified the following five typical problems in the representation of information in worker assistance systems [67, p.341]:

- **Missing information:** The authors identified that necessary information for the execution of the next work step is often missing, so that other employees or the superior must be asked.
- **Display of unnecessary information:** Another problem appears if there is too much information provided, although it is not needed. This leads to a so-called information overload.
- **No process orientation:** This occurs, when the information is not represented as a process and does not correspond to the individual sub-steps.
- **Outdated information:** This error occurs, when outdated graphics or representations are used, e.g. in assembly instructions. If the deviation is too big, this could lead in the worst case to the problem that workers are not able to solve their tasks.
- **Wrong way of displaying information:** This happens, when the information is displayed in a wrong place and with a wrong representation type. This makes it difficult for the employee to perceive the information.

On the basis of these errors, they have developed the following design recommendations, which should lead to better (more efficient) worker assistance systems and consequently to a higher productivity [67, p.341]:

- **Display all information that is required:** The authors recommend to show all information for each step which is really needed to execute the task. Otherwise this leads to the problem of missing information, where, in the worst case, the employees are not able to solve the tasks.
- **Display only relevant information:** As mentioned in the first recommendation, it is problematic if too little information is available. On the other hand, too much information is not helpful either, which is why a trade off must be found. Therefore, all necessary information should be provided to solve the tasks but not more.
Show only the information that is absolutely necessary for the task.
- **Use step by step instructions:** Since assembly processes are processes consisting of various sub-steps, use step-by-step instructions as declaration.
- **Integrate the worker assistance system into the IT system:** In order to keep the product data up-to-date, the assistance system should be integrated into the IT system.
- **Use pick-to-light systems for commissioning and in-situ projection for displaying information:** The authors recommend to use a suitable technology for the representation of the information, such as pick-to-light systems for commissioning processes instead of bills of material.

In order to avoid these typical errors, which lead to a decreasing efficiency and a lower user acceptance of the worker assistance system, special attention will be paid to this recommendations during implementation.

2.2 Human-Computer Interaction

Human-Computer Interaction, or short HCI, is the field, as the name suggests, of how humans and computers (or more generally: machines) interact with each other. The need for this arose from the development and widespread of electronic computers over the last 50 years. After all, they had to be operated in some way by the user. Historically, the foundation for the concept of human-computer interaction was laid in 1946 with the development of the first general-purpose electronic computer called ENIAC (Electronic Numerical Integrator and Calculator) [68, p.2].

There are numerous definitions of the term "Human-Computer Interaction" in the literature, such as:

"Human-Computer Interaction (HCI) is the study of the way in which computer technology influences human work and activities." [69]

"Human-Computer interaction is the command and information flow that streams between the user and the computer. It is usually characterized in terms of speed, reliability, consistency, portability, naturalness, and users' subjective satisfaction." [70]

"HUMAN COMPUTER INTERACTION is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." [68, p.1]

To understand HCI, it is important to realise that human-computer interaction is not just about the design of human-computer interfaces - it is an interdisciplinary field that encompasses a variety of areas. Becker [71] defines the following areas which are important for HCI (see Figure 2.13):

- **Design & Media:** How should "good" user interfaces for HCI be designed?
- **Computer Science & Engineering:** How can such systems be implemented?
- **Human Factors & Ergonomics:** The integration of the human being. What are the ergonomic requirements to be considered?
- **Behavior Science & Psychology:** Human behaviour and psychology also play an essential role in development and must be taken into account.
- **Other Professions:** Furthermore, there are numerous other fields that contribute to HCI, e.g. cultural anthropology, user research, etc.

HUMAN-COMPUTER INTERACTION

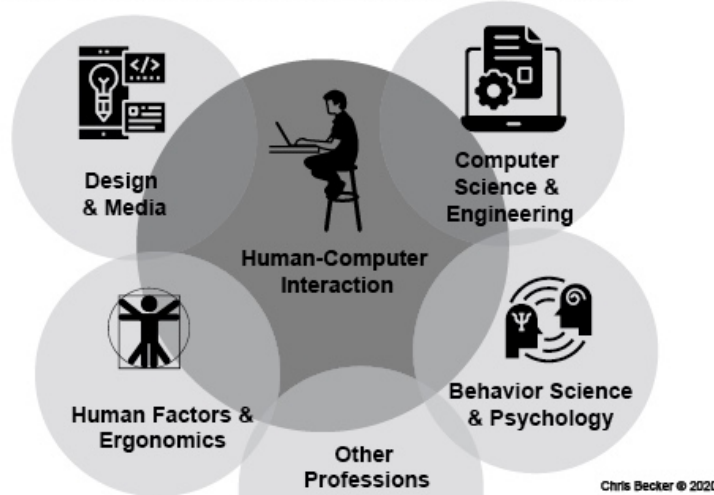


Figure 2.13: Interdisciplinarity of HCI [71]

In summary, human-computer interaction describes the exchange of information between humans and computers/machines and thus deals with the design, implementation and evaluation of interactive systems. For this communication, an interface is subsequently required. The so-called human-computer interface, short user interface (UI) is the software that enables this communication between human and computer [70].

There are various ways to implement this human-computer interface. One type of user interface that has become established in HCI in almost all areas where computers are used in recent decades is the so-called graphical user interface (GUI).

2.2.1 Graphical User Interfaces

Before we look at graphical user interfaces, we must first clarify what a user interface is.

The iso standard 9241-110 defines the term user interface as

"all components of an interactive system that provide information and controls for the user to accomplish specific tasks with the interactive system" [72]

This means that the user interface (UI) provides information and possibilities to control the interactive system. Now there are many different ways in which this information can be provided and the system controlled, e.g. by means of light signals, screens, keyboard, mouse, hand movements and so on.

The main reason why graphic user interfaces are so successful and widespread is based on the principle of direct manipulation. This is a style of interaction, introduced by Ben Shneiderman [73], which is characterised by the following three properties [73, p.57]:

- Visibility of the objects: All objects that can be changed are visualised graphically, e.g. files or folders.
- Rapid, incremental and reversible actions: The user should be able to carry out the actions quickly and it is important that they are reversible.
- Simple direct manipulation of objects instead of text-based commands: The user can change the elements directly, for example change the state of an element by moving it instead of using complex commands.

The idea behind this concept was essentially to enable users without special knowledge to use human-computer interfaces, so that the user does not have to remember any commands, as the functions are self-explanatory due to their visual presentation [74, p.89f]. GUIs have thus made computers much easier to use, in other words, systems have become more intuitive, but what exactly does intuitive mean?

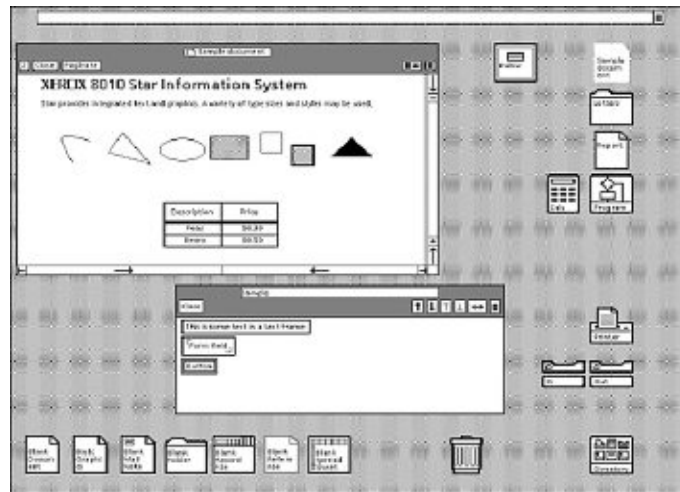


Figure 2.14: The first commercial GUI, developed by Xerox [75, p.28]

2.2.2 Intuitive Use - Usability

Many manufacturers advertise their software in various areas by saying that it is intuitive to use. But what does that actually mean? The term "intuitiveness", usually referred to as "intuitive use", is not uniformly defined [75, p.21], there are many different definitions available.

The definition by Naumann et al. [76] defines intuitivity as follows:

"A technical system is, in the context of a certain task, intuitively usable while the particular user is able to interact effectively, not-consciously using previous knowledge." [76, p.129]

Another definition by Hurtienne et al. [77] defines the term "intuitive use" as follows:

"A technical system is intuitively usable if the users' unconscious application of prior knowledge leads to effective interaction." [77]

Looking at these two definitions, they agree that

- already existing knowledge is used (unconsciously)
- to effectively solve a task

The definition by Naumann et al. [76, p.129] goes one step further. The authors are of the opinion that the term "intuitive use" must always be related to a specific task. In the context of the following work, this specific task would be the programming/modelling of collaborative human-robot processes.

What is the difference between "intuitive use" and usability?

The ISO standard 9241-11 [9] defines usability as the

"extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [9]

The main difference between usability and intuitive use is that intuitive use applies existing knowledge. This is particularly emphasised in the definitions (see above). Indirectly, however, this can also be assumed in the usability definition, since the definition refers to "special users".

At this point, one could still ask the question: why should user interfaces be easy to use (intuitive)?

The main reason is to reduce mental effort. The user should be able to solve tasks efficiently, and the simpler the operation, the better. Intuitive UIs reduce the user's cognitive load because the user solves the tasks unconsciously with the help of already existing knowledge. Through this subconscious application of pre-existing knowledge, the cognitive load is reduced and the user can thus fully concentrate on other things. [76]

How do you achieve good usability or intuitive systems/products? Two main areas have developed to deal with this problem: **User Interface Design** and **User Experience Design**. For easy-to-use systems, the concept of **Mental Models** is also of great importance, which we will also briefly look at in the following section.

2.2.3 User Interface Design, User Experience Design and Mental Models

User Interface Design

User Interface Design (UID), as the name suggests, deals with the design of user interfaces. UID is a sub-field of human-computer interaction [78, chapter 1]. The question that now arises is how do you create a "good" user interface? In the course of the last decades, numerous guidelines have been developed to answer this question. A good overview of existing and widespread guidelines is provided by Johnson [79]. However, when using such design guidelines, it is always important to bear in mind that they are not a step-by-step guide on how to design the perfect user interface. Such guidelines are usually a relatively general description of goals that are important, e.g. consistency, prevent errors, etc [79, p.vi-vii]. However, how to achieve and implement them is a matter of interpretation and depends on the specific application [79, p.vi-vii].

User Experience Design

Now, for the so-called user experience, it is not only the user interface that is important, but a multitude of areas. The ISO norm 9241-11 [9] defines user experience as the

"user's perceptions and responses that result from the use and/or anticipated use of a system, product or service" [9, section 3.2.3]

"Users' perceptions and responses include the users' emotions, beliefs, preferences, perceptions, comfort, behaviours, and accomplishments that occur before, during and after use." [9, section 3.2.3 (Note 1)]

This means that in order to achieve a good user experience, a good user interface is necessary, but this is only one part. Furthermore, it is essential to know the emotions, preferences, etc. of the potential user already before development in order to create a system that meets these requirements. The user should be involved in the development process of the system. An approach which makes this possible is the so-called User-Centered Design (see Chapter 1.3). This approach was also applied in the course of the thesis to achieve the best possible usability and user experience.

Mental Models

A mental model is a representation of reality that helps people to understand how things work [80, p.112]. When designing intuitive graphical user interfaces, an important concept is to understand the mental model of the users as well as the use of metaphors. By the use of metaphors it is possible to transfer a mental model from one area to another area [74, p.79].

A very popular example is the office metaphor. The aim was to represent the mental model of an office employee and to use it for the graphical user interface of a personal computer. All actions that can be performed in the office, e.g. sorting documents into folders, removing documents by moving them to the paper basket, etc. were digitally mapped. One reason why PCs with such GUIs are so easy to use is that the mental model of a classical office worker has been perfectly mapped. [74, p.79]

In summary, this means that for intuitive GUIs it is important to understand the mental model of the future user and to design the user interface based on this mental model. Furthermore, the use of metaphors can help to design an "intuitive" user interface, as the user is already familiar with these elements from other areas. Successful examples of metaphors are, for example, the use of brushes and paint buckets in image editing programmes or knobs and sliders in audio editing programmes.

2.3 (Business) Process Modelling

2.3.1 What is a process?

Before we have a look at process modelling, it is first necessary to clarify what a process is. Different definitions can be found in the literature.

The ISO 9000 standard defines a process as a "set of interrelated or interacting activities that use inputs to deliver an intended result" [81, p.33].

Mertens defines a process as follows: "A process consists of a sequence of individual functions (function sequence) and has a defined starting point (trigger of the process) and end point (end state)"¹ [82, p.24].

Davenport specifies a process in the following way: "A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action" [83, p.5].

There are numerous other definitions. In the literature, the terms process and business process are often used as synonyms [84, p.5]. This is due to the fact that processes and especially process modelling play a particularly important role in companies. Therefore, people think of process modelling at first sight that business processes are modeled. In the course of this work it is insignificant whether we are dealing with processes or business processes. Nevertheless, the importance of business processes and the need to model, optimize and simulate them has led to the development of a number of different modelling methods and tools that can also be applied to other areas and disciplines, what will also be done in the course of this work. But more on that later.

Summarized from the definitions above, a process consists of:

- tasks,
- which are executed in a certain order
- to achieve a specific goal and
- each process also has a defined starting and end point.

In order to map and optimize processes, there are various models and tools that have been established over time. In the next section we will take a look at different types of process modelling and we will discuss them in more detail.

¹Translated by the author: "Ein Prozess entsteht aus einer Folge von einzelnen Funktionen (Funktion-sablauf) und weist einen definierten Anfangspunkt (Auslöser des Prozesses) sowie Endpunkt (Endzustand) auf." [82, p.24]

2.3.2 Types of process modelling

In the area of process modelling a distinction can be made between the following two types [85, p.89]:

- **Scripting Languages:** Scripting languages use a formal notation to describe the models. This is done in textual form, similar to programming languages and has the main advantage that the models can be described in great detail [85, p.89]. The big disadvantage is that this kind of modelling requires appropriate knowledge to understand the process models [85, p.89]. Therefore, this variant is not suitable for the representation of an intuitive method for modelling collaborative human-robot processes.
- **Diagram Languages:** These are modelling languages that use graphical objects (diagrams) for representation. This has the great advantage that such models are easier to understand than scripting languages, which is why this type of modelling is more widespread in practice [85, p.89]. For this reason, a diagram language is clearly more suitable for a worker assistance system than a scripting language.

Diagram languages can be divided into the following three categories according to their focus [85, p.89f]:

- **Data flow oriented:** As the name suggests, these methods focus on the data flow. Thereby it is modeled, which data are important in the course of the process, how they are generated and so on.
- **Control flow oriented:** In this approach the focus is on the control flow. The aim is to present the process in the best possible way so that it is easy to understand which sub-steps the process consists of and in which order they are executed by whom. Examples for control flow oriented modelling languages are the Business Process Model and Notation (BPMN) [86] and the Event-driven Process Chain (EPC).
- **Object oriented:** The object-oriented approach uses objects to describe the process. This approach comes from the field of software engineering. A very well-known representative of this type is the Unified Modelling Language (UML) [87], which can be seen as the standard for the modelling of software systems.

Since the sequence of the individual tasks as well as the assignment of who is responsible for executing the tasks is important for the mapping of collaborative human-robot-processes, the control flow oriented modelling languages seem to be best suited for this purpose.

For the implementation of the prototype in the course of the thesis, the Business Process Model and Notation (BPMN) is used, which is briefly explained in the following chapter.

2.3.3 Business Process Model and Notation (BPMN)

As the OMG (Object Management Group) describes in their specification of the BPMN (Business Process Model and Notation) Language [86], the goal is to create a simple notation for mapping business processes, which is easy understandable for the following three user groups:

- Business Analysts: Analyze the processes within a company and map them using BPMN.
- Technical Developers: Are responsible for the technical implementation of the mapped business processes.
- Business Users: Are persons who monitor and manage the implemented processes.

The idea is to have a tool that helps to model, implement and understand business processes, both for business users/analysts who are familiar with the business domain of a company and for technical staff who is familiar with the implementation of information systems. Typically, these two groups have a different language within their domain and they use different modelling tools. BPMN tries to solve this problem by defining a good and easy understandable notation with the goal to create "... a standardized bridge for the gap between the business process design and process implementation" [86, p.1].

Historical Background

BPMN was originally developed at the beginning of the 2000s in the course of the Business Process Management Initiative (BPMI) [88]. The idea of BPMI, a non-profit organization with members of different companies like IBM, Hewlett-Packard, Fujitsu and Sun Microsystems, was to develop standards for Business Process Management [89]. Stephen A. White from IBM created the first version of BPMN for this purpose in 2004 [90]. Finally, BPMN was adopted and further developed by the Object Management Group, short OMG, since 2005 [88]. Since 2014 the current version of the Business Process Model and Notation language is version 2.0.2 [91].

The OMG [92] is an international consortium, which pursues the goal of developing and maintaining standards in the area of information technology. Besides BPMN, one of the most known standards from the OMG is the Unified Modelling Language (UML) [87].

Example of a business process mapped with BPMN

To get a better understanding of what BPMN is and what BPMN looks like, we will have a look at a typical example [93] of a business process modeled with the Business Process Model a Notation language.

As shown in Figure 2.15, the BPMN model is a representation of a process for credit application.

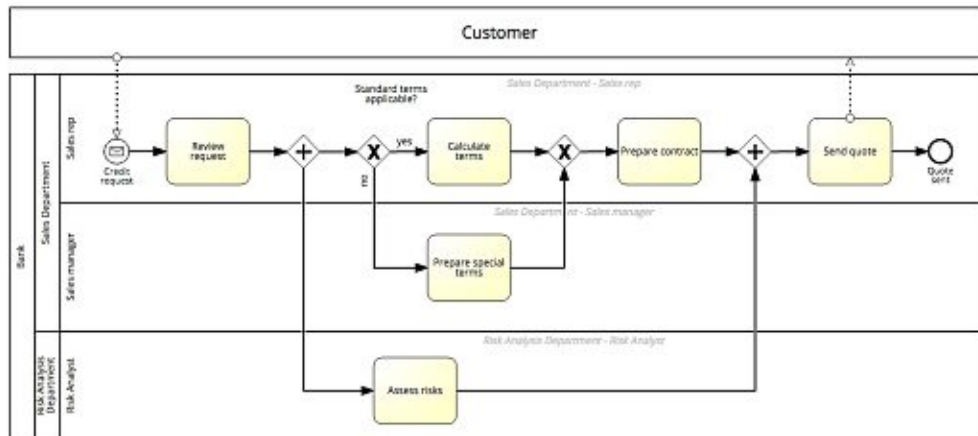


Figure 2.15: BPMN example: bank credit quote creation process 93

The two big rectangles represent the two actors, on the one hand there is the customer, who wants to apply for a credit and on the other hand there is the bank, which checks the granting of the credit. These rectangles are so-called pools, but more about this later.

The process in the example runs as follows. The customer first sends a request to the bank. This can be done e.g. by telephone call or e-mail. This is symbolized by the dotted line and the symbol with the envelope. The customer's inquiry starts the process at the bank. As can be seen in the bank's pool, the process runs across several departments and responsibilities. The two departments involved in the process are the Sales Department and the Risk Analysis Department. Within these departments employees with particular responsibilities are needed: a sales representative, a sales manager and a risk analyst.

The sales representative first checks the request. Afterwards, the branching with the plus sign shows that tasks afterwards run parallel. A risk analyst evaluates the risks, meanwhile the sales representative decides whether the standard conditions can be applied or not, represented by the branch with the X. This means that only one path is followed afterwards. If the standard conditions can be applied, the sales representative calculates the conditions. If the standard conditions can not be applied, a sales manager must define special conditions. The sales representative then prepares the contract. Once the contract is ready and the risk analyst has completed the risk assessment, the offer can be sent to the customer. The circle with the thick outline finally shows that the process is completed.

In the example above (see Figure 2.15) a very simple one was used, which at the same time represents all the essential basic elements of BPMN that will be needed later.

Summarized the following elements were used in the example above:

- several parties/departments represented by so-called pools and swimlanes

- tasks, shown by the rounded rectangles
- branches, pictured by the rhombuses
- connections between the tasks, parties, branches and events, represented by the arrows
- start as well as end events, which initiate or end the process, shown as circles

On the following pages these and other important BPMN elements and concepts are described in detail.

BPMN Elements

In this section all basic elements and concepts are explained which are needed for the use of the BPMN prototype later on. Basically there are only five different categories of elements [86, p.27]:

- Swimlanes
- Flow Objects
- Connecting Objects
- Data
- Artifacts

However, in the course of the work and for using the prototype to program the collaborative robot, we only need three categories: Swimlanes, Flow Objects and Connecting Objects.

Swimlanes

The "Swimlanes" concept is used to improve/simplify the representation of the processes. Consider the following scenario: a process consists of a number of activities/tasks and the activities are performed by a number of different people (with different skills). Now the question arises, how can you easily map which person/department is responsible for which task? This is exactly what the swimlanes concept is used for.

The "Swimlanes" concept in BPMN consists of the following two elements [86, p.28]:

- Pools
- Lanes

A pool can consist of one or more lanes and represents a coherent unit. Thus, a pool can be used for example as in Figure 2.15 for a company or for a customer. A lane represents an actor, which has a certain function/role. In Figure 2.15 the swimlanes are used for departments and subsequently for special roles like the "Risk Analyst". This concept allows a clear presentation of responsibilities in a very simple way and is also very well suited for the representation of collaborative human-robot processes (see Figure 2.16).



Figure 2.16: BPMN "Swimlanes" concept for collaborative human-robot processes (own Figure)

Flow Objects

Flow Objects represent the basic elements that are needed to model the processes [86, p.27].

A distinction is made between the following three types [86, p.27-29]:

- **Activities:** An activity is used to model tasks/activities and is represented by a rounded rectangle. An example could be the task: Move component X from position A to B.



Figure 2.17: Activity [86, p.29]

- **Events:** This type of flow objects is used to trigger something, e.g. start the process. In Figure 2.15 the process is triggered by the mail event of the customer.

There are a lot of different types of events in BPMN. For the prototype we will only need the following two: start event and end event (see Figure 2.18). As the name already suggests the start event defines the beginning of the process and the end event the end. Events are represented by a circle.



Figure 2.18: Start/End event [86, p.31]

- **Gateways:** The control flow for BPMN processes is usually sequential. With the help of gateways, represented by rhombuses (see Figure 2.19), e.g. parallel as well as alternative process sequences can be mapped [86, p.34]. For examples how to use these elements see Figure 2.15.

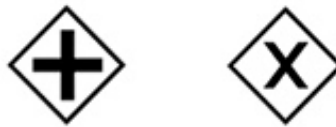


Figure 2.19: Parallel/Exclusive Gateway [86, p.34]

Connecting Objects

In order to illustrate the sequence of execution of the process, so-called connecting objects, represented as arrows, are used to link events, gateways and activities.



Figure 2.20: Sequence Flow [86, p.29]

This was a very brief introduction to the main concepts and elements of BPMN. The Business Process Model and Notation language offers lots of other elements and features, which were not mentioned because they are not important in the course of the work. For interested persons a detailed description of BPMN 2.0 can be found in [86], [94].

2.3.4 Reasons for the use of BPMN 2.0 in the context of the thesis

- **Distribution:** A big advantage of BPMN is its wide distribution. Lübbe and Schnägelberger [95] came to the conclusion in their study, in which 64 companies were surveyed, that BPMN is the most frequently used notation for process modelling. The spread is also shown by the huge number of modelling tools that support the Business Process Model and Notation language [94, p.10]. This can be seen e.g. on the website bpmn.org [96], which currently refers to more than 60 applications. There are also various open source projects which can be used as a basis for own projects. One example is the github project bpmn.io [97], which provides a web-based editor for modelling BPMN 2.0 compliant models. This project will also be used as a basis for the implementation of the prototype.
- **Easy to Understand:** Two essential characteristics that contribute to the success of software are low threshold and high ceiling [98, p.6]. The threshold (high/low) defines how difficult it is to learn how to use the system and ceiling (high/low) describes what kind of problems can be solved [98, p.6]. The combination low threshold/high ceiling means that it is easy for the user to use the system, but it is also possible to solve really complex problems. A very well-known example, which implements this concept perfectly, is Microsoft Excel. Everyone can do simple calculations in a short time, but Excel also offers the possibility to solve really complex problems. BPMN offers a similar concept. As we have seen above, only very few elements are needed to create and understand simple process models. But due to the multitude of functions provided by BPMN, even very complex processes can be mapped and automated. How extensive the Business Process Model and Notation language is, can be seen very well in the specification [86], which has more than 500 pages.
- **OMG [92] Standard:** Compared to other modelling languages for business processes such as Event-Driven Process Chains (EPCs), BPMN is a defined standard [94, p. 10]. This offers the great advantage that all BPMN models use the same notation.
- **Swimlane Concept:** The use of the "Swimlane" concept allows a very clear and simple presentation of responsibilities (see section BPMN Elements), which is also very well suited for the representation of collaborative human-robot processes.
- **Execution of Models:** The initial idea of BPMN was, as already mentioned at the beginning of the chapter, to create a modelling language that combines the worlds of IT and business. While in BPMN version 1.0 only graphical modelling was possible and for automation a transformation into an executable/computer readable language (e.g. BPEL) was necessary, this changed with version 2.0 [94, p.11-13]. For this purpose a so called BPMN-engine is used, which is responsible for the automation/execution of the process models. There are many software solutions

available on the market that offer such BPMN 2.0 engines, both commercial and free ones like Signavio [99], Camunda [100], etc.

Zalando can be mentioned as a practical example. Parts of the ordering process were mapped and automated using BPMN 2.0. This means that with every order a new instance of the BPMN ordering process is executed in the background. [101]

Summarized one big advantage of BPMN 2.0 models is therefore that they can be executed without transformation using a BPMN engine. This is important for the implementation of the prototype, because with BPMN 2.0 and the use of a BPMN engine we can both model and execute the collaborative processes.

State of the Art

In this chapter we will discuss the current state of the art in programming industrial and collaborative robots. However, it is important to consider the current state of the art from several points of view, because cobots are not only interesting in the research area and there are also lots of different solutions already available on the market. Therefore, it is necessary to have a look on the current state in research and also in industry/on the market.

First, there will be a general introduction into methods how robots and cobots can be programmed.

In the second section we will deal with the research question "What does an intuitive robot programming interface look like that can simultaneously serve as a worker assistance system? By means of a systematic literature research, short SLR, the current state of research in the development of programming environments, which on the one hand try to simplify the programming of cobots and on the other hand offer the possibility to represent the complete collaborative human-machine process, will be clarified. Furthermore, we will look at whether there are programming environments that offer the possibility of storing additional information during the creation process, so that during the execution of the programs the system can also serve as an assistance system for the worker.

Last but not least, current developments in the business world should not be neglected, since the programming of cobots has been gaining ground in the industry for several years. For this purpose, we take a closer look at various programming environments from well-known manufacturers such as Universal Robots, Kuka, Franka Emika and we will also have a look at other development environments for the programming of collaborative robots, which were found in the course of an online research.

3.1 Programming Robots/Cobots

3.1.1 Types of Robot Programming

In the course of time, different methods of how robots can be programmed have evolved. These can be roughly divided as follows.

Online/Offline Programming

Online programming is a method where the robot is programmed by direct interaction [102], [103]. There are different ways in which this interaction can take place, e.g. by the use of a teach-pendant, via programming by demonstration (PbD), etc., but more on this later. The majority of manufacturers rely on the use of teach-pendants in combination with programming by demonstration (see Figure 3.1). With the aid of these teach-pendants and PbD, the operator carries out each work step. The different positions, motions and actions of the end effector are recorded and thus create the robot program [102, p.88]. The main advantage of this type of programming is its simplicity [102, p.88]. Furthermore, compared to offline programming, changes can be made very quickly. This is particularly important for flexible systems with small batch sizes.

In comparison, offline programming is not performed directly on the robot (see Figure 3.2). With the help of e.g. 3D simulations, the scenario is simulated and tested [103, p.2352]. Afterwards the simulation has to be converted into code that can be executed by the robot and finally it must be transferred. An advantage of this method is that the code can be altered and tested offline, while the robot carries out its work, which in turn minimizes the downtime [102, p.88]. Disadvantages are on the other hand that the programming is very complex and requires experts. Compared to online programming, changes take a very long time, which means that this method is not as flexible in adapting to rapidly changing processes. Furthermore changes are much more costly, due to the need for experts. For these reasons, offline programming is more suitable for high volumes and processes that remain unchanged for a long time [102, p.88]. This type of programming is typically used for classical industrial robots such as in automotive industry.



Figure 3.1: Online Programming via teach-pendant [104]

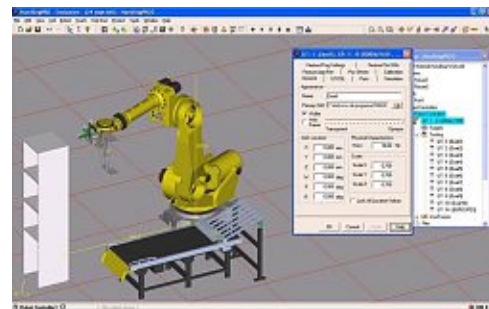


Figure 3.2: Offline Programming with 3D simulation software [105]

Automatic/Manual Programming

Biggs and Macdonald [106, p.1-2] divide the methods of programming robots into automatic (see Figure 3.3) and manual programming (see Figure 3.4). Automatic methods are those where humans have little influence on the code, such as programming by demonstration or instructive systems (e.g. gesture recognition). As manual programming, the authors specify all those methods in which the user creates the robot code by means of graphical or textual programming.



Figure 3.3: Automatic programming with Programming by Demonstration [107]

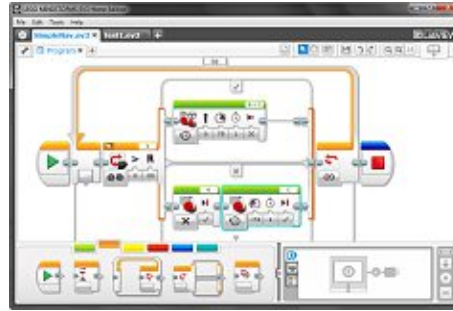


Figure 3.4: Manual programming (graphical) with Lego Mindstorms [108], [109]

A distinction is made between the following automatic programming methods [106]:

- **Programming by Demonstration (PbD):** In this method the operator moves the robot by hand, by using a control panel/teach pendant or via teleoperation [110, p.213]. When the user guides the robot, the respective positions are stored and afterwards the tasks can be performed automatically [110, p.213]. Especially with regard to the programming of collaborative robots, PbD is a very popular and widespread method [51, p.173]. This can be seen by lots of research, which is done in this field. Furthermore almost all cobot manufacturers on the market use PbD, as we will see in section 3.3. The reason for this is the simplicity and intuitiveness compared to other programming methods [51, p.173].
- **Learning Systems:** In such systems, the robot is provided with examples for solving a task. By using machine learning methods, such as neural networks or reinforcement learning, the robot programs "itself". This method is also very well suited to improve already programmed sequences to maximize the performance. [106, p.6]
- **Instructive Systems:** Instructive Programming is a method which uses, for example, natural language or gestures to control the robot in real time [106, p.7f]. However, these commands must first be taught in some way (e.g. via programming by demonstration) [106, p.7f]. By using natural language and/or gestures such

systems are very intuitive on the one hand. On the other hand, the problem is that the intuitiveness is at the expense of reliability, which is why Zaatari et al. consider graphical user interfaces (GUIs) and haptic solutions as more suitable for industrial applications [51, p.167f].

The manual programming methods can be divided into the following two groups [106]:

- **Textual Programming:** As the name suggests, this type uses textual language for programming. This offers the advantage that the flexibility is very high, so that a very wide range of scenarios can be implemented with this method. Classical (textual) programming, however, has the disadvantage that a lot of knowledge is required and therefore experts are needed for implementation. [106]

There are many different text-based robot programming languages available, such as VAL, ARLA, KRL and Rapid. Which one is finally used depends on the specific application and/or the robots supported. Some of these languages (e.g. KRL, VAL) were developed by robot manufacturers, which means that the respective robots of the manufacturer are particularly well supported and others are not or only partially supported, resulting in a manufacturer lock-in. In addition to these robot specific programming languages, there are also frameworks, such as the Robot Operation Systems (ROS), which can be used with classic programming languages (e.g. Java). But more on this in Chapter 3.3.

- **Graphical Programming:** Compared to textual programming this method uses graphic elements, which can be adjusted by the use of parameters, instead of text. To create programs, these graphic elements are then combined to e.g. blocks or flowcharts. This has the great advantage that the programming is very simplified. The disadvantage, however, is a lower flexibility. [106]

A very well-known example of a graphical programming interface is Lego Mindstorms (see Figure 3.4). After building a robot using Lego bricks it can be programmed with a graphic (flowchart-based) interface on a computer or tablet. By combining the different blocks and specifying their parameters, the user defines how the robot should behave. There are many ways in which the system can be expanded, so that, for example, it is possible to implement voice control via Amazon Alexa to trigger the execution of the programs and thus control the robot. The age recommendation is for children from 10 years and no programming skills are required. This example shows very well how easy (graphical) programming can be.

Now one could argue that graphical programming environments are already automatic programming. Depending on the abstraction this is partly true, graphical methods are somewhere in between. Nevertheless, there is a direct connection between the graphical elements and the programming statements [106]. With automatic programming (e.g. gesture recognition) this is no longer the case.

In summary, this overview has shown that many different methods exist for programming robots. In terms of simplicity and intuitiveness, which is especially important for easily adaptable collaborative systems, text-based systems are not very well suited. Both automatic methods, especially Programming by Demonstration (PbD), and Graphical Programming are very well suited for this purpose and often they are used in combination. This is because every type of programming, even if it is carried out via direct interaction with the robot arm, requires a possibility to subsequently edit and change the created programs. This is where typically the graphical user interfaces come in. In the course of time various graphical approaches/methods have been developed, which we will now look at in more detail.

3.1.2 Programming Robots/Cobots with Graphical User Interfaces (GUIs)

There are several ways/concepts in which cobots can be programmed by means of graphical user interfaces. Zaatari et al. [51] distinguish between the following four types.

Cobot teaching pendant

This is the most common variant, used for programming collaborative robots, available on the market. The teaching pendant essentially consists of a screen and numerous buttons and is connected with the robot, either wired or wireless. Looking at, for example, the user interface of Universal Robot, programming is done by guiding the robot and defining waypoints (see Figure 3.5). The teaching pendants of other manufacturers essentially follow the same concept (see Chapter 3.3). The big advantage is its simplicity but how intuitive and simple the usage is finally depends on the GUI as it can be seen in [57].

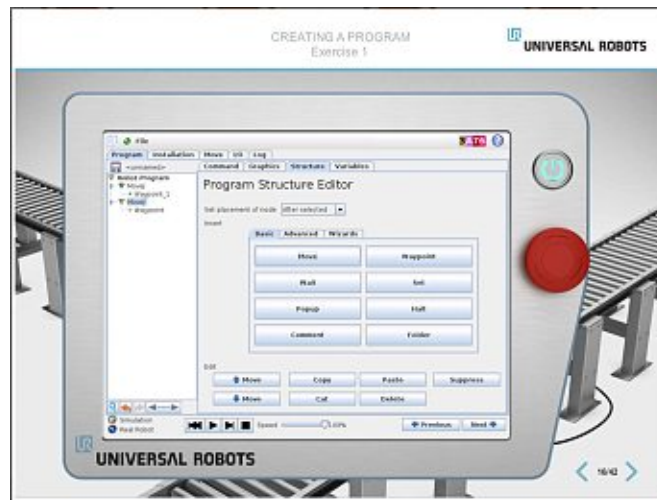


Figure 3.5: GUI of Universal Robots teaching pendant [111]

Icon-based programming

It was recognised very early on that the programming of robots in the industrial environment can also be considerably simplified by means of graphic elements (icons) instead of text-based solutions [112]. In the 1990s, a project called AMIRA (Advanced Man Machine Interfaces for Robot System Applications) was launched with the aim of developing a new generation of graphical user interfaces for programming robots in industry [113]. In the course of the project prototypes were evaluated and a style guide for user interfaces was created [113]. As a result, the MORPHA style guide for icon-based programming was developed [112]. In addition, the ISO standard 15187 was published in 2002, but it was already withdrawn six years later in 2008 [114]. Although the original idea of creating a standard that makes programming independent of syntax and can be used by all manufacturers was very good, the MORPHA approach for icon-based programming has never really been accepted in the industry [115]. Possible reasons for this are the difficulty in debugging, maintaining and modifying the created programs as well as their limitations compared to classical text-based methods [51].

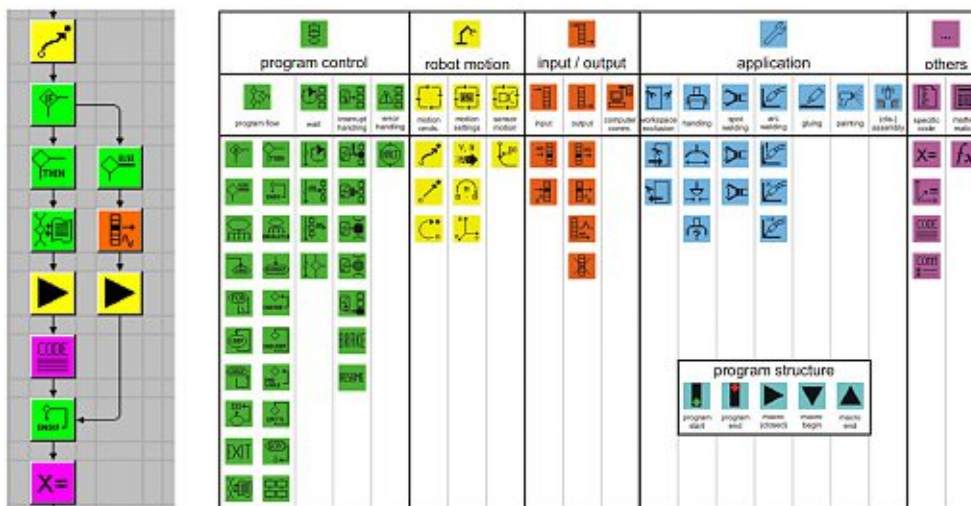


Figure 3.6: Example of a Morpha program (left), Kuka specific implementation according to the (withdrawn) ISO standard 15187 (right) [112]

CAD-based programming

Computer-aided design, short CAD, refers to the creation of products with the help of (3D) graphic programs [116]. For modelling and simulating industrial scenarios there are various programs from different manufacturers on the market available, such as CoppeliaSim [117] (see Figure 3.8), RobotStudio [118] (see Figure 3.7) and Autodesk [119]. The main disadvantages are the complexity and the high effort involved in creating and altering programs [51]. For these reasons this method is typically used for applications with high volumes in mass production but it is not suitable for collaborative human-robot

scenarios with small batch sizes, where it is necessary to change processes easily and quickly.

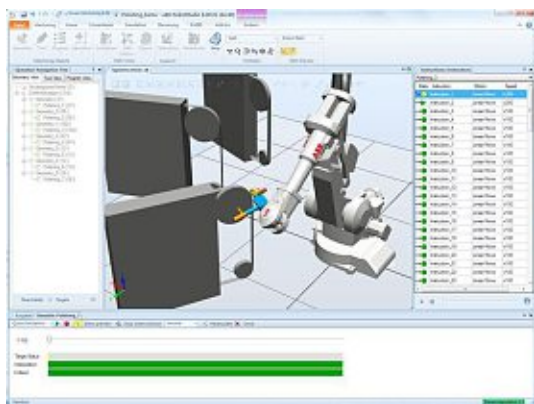


Figure 3.7: GUI of RobotStudio [120]

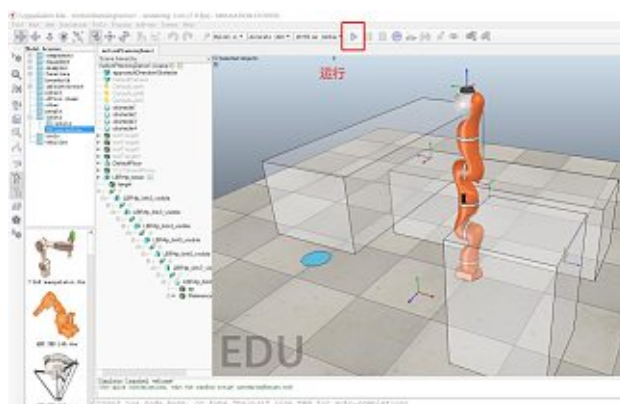


Figure 3.8: GUI of CoppeliaSim [121]

Task-based programming

Task-based programming is a paradigm in which, in contrast to e.g. icon-based programming, the focus is on object-oriented goals instead of individual motions of the robot. This is done by using primitives, skills and tasks. Primitives are the basic movements of a robot like move or grasp and they describe the individual motions of a robot. To make programming easier and more intuitive, one possibility is to abstract the motions to so-called skills. A skill represents an object-oriented goal to be achieved, such as rotate, pick, etc. It is important that on the one hand the skills are abstract enough to be easily understood by humans and that on the other hand they are general enough to be used for a variety of tasks. A task describes an overall goal to be achieved in a scenario such as Pick&Place Object A and is created by combining skills. [122]

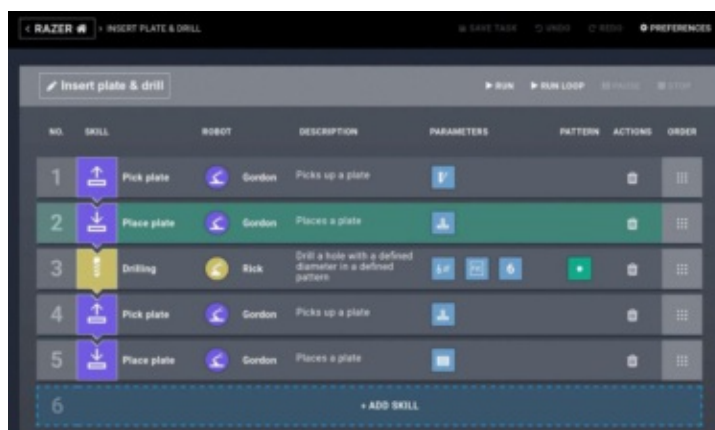


Figure 3.9: Task-based GUI Razor [123]

The task-based approach seems to be the most promising in terms of intuitive cobot programming, which is why this research direction is currently also the most popular [51].

After this brief introduction to different methods and approaches how robots can be programmed, we will now look at the state of the art in research in the following section.

3.2 Systematic Literature Review

As a starting point for the Systematic Literature Review the following research question was considered: **"What does an intuitive robot programming interface look like that can simultaneously serve as a worker assistance system?"**. With the help of the SLR it should be clarified to what extent articles already exist which deal with this topic. The guidelines defined by Kitchenham [15] served as a basis for conducting the review.

Looking at the research question in detail, the following areas/terms that are relevant for the search can be identified:

- **Interfaces:** The goal of this work is to design and implement a graphical user interface that facilitates the programming of collaborative human-robot processes. Since the term interface is a very general word, a more appropriate choice is the term: user interface. Since this limited the search too much, this term was finally omitted.
- **Robot:** Also the term robot is a bit too general, because robots could also be classical industrial robots or toy robots or vacuum cleaning robots. In our concrete case only collaborative robots are of interest, which is why the term cobot is used instead of robot.
- **Programming:** The search should deliver results that deal with the programming of cobots, so this word must be a fixed part of the search.
- **Work(er) assistance system:** At first glance, the term worker assistance system seems a good choice. However, this is not the case, as we will see soon.

Taking into account possible synonyms or abbreviations, the search string looks like following:

**("user interface" OR UI) AND
(cobot OR "collaborative robot") AND
programming AND
"worker assistance system"**

Searching various databases such as IEEE Xplore [16], Sage Journals [22], SpringerLink [18] and ScienceDirect [17] provided an interesting result. All databases mentioned

above returned 0 results. After intensive error search, the "problem" was the term: worker assistance system. Also the usage of the synonym work assistance system and the replacement by "assistance system" provided only very few results. The conclusion at this point is that the current search string restricts the query too much, which is why it needs to be adapted further.

In order to determine the state of the art in intuitive graphical programming of collaborative human-robot processes it was necessary to remove the term worker assistance system, nevertheless, the representation of the whole human-machine process should be taken into account. For this reason, the terms process modelling and workflow were added in the query.

Furthermore it was necessary to define a criterion, how to determine if the query is appropriate or not. As Kitchenham explains, it is important to conduct trial searches and to check whether literature already known in this field is part of the results [15, p.14]. Therefore, the criterion was to determine if already known studies like [124], [56], [57] were included in the results.

The final query, which was used for the Systematic Literature Research, looks like following:

**(cobot OR "collaborative robot") AND
programming AND
("process modelling" OR "process modeling" OR "workflow")**

Inclusion/Exclusion Criteria

As Kitchenham states so-called inclusion and exclusion criteria should be used to refine the search and to filter out irrelevant studies [15, p.18 f.].

- The term Cobot was coined by James Edward Colgate in 1996 [3]. For this reason only articles from 1996 onwards are of interest.
- Since there has been lots of research on cobots in the German-speaking area in recent years, both German and English literature is accepted.
- Furthermore only articles with full text will be considered, only abstract results are excluded.
- Only articles are of interest which can be accessed via open access or via the VPN of the TU Wien.
- A major focus of the whole work is the programming of cobots via graphical user interfaces, so all studies that do not deal with the programming of cobots or the intuitiveness of user interfaces are excluded.

- As already explained in detail in section 3.1, there are various methods of programming collaborative robots, such as programming by demonstration. The general possibilities of programming cobots have already been discussed in this section. The thesis focuses on the programming via graphical user interfaces (GUIs). Therefore, all results that do not deal with GUIs are sorted out.

Databases

The following databases were used for the search:

- IEEE Xplore [\[16\]](#)
- ScienceDirect [\[17\]](#)
- SpringerLink [\[18\]](#)
- Google Scholar [\[19\]](#)
- Citeseer Library [\[20\]](#)
- ACM Digital Library [\[21\]](#)
- Sage Journals [\[22\]](#).

Conducting the search

The search yielded the following results: IEEE Xplore: 62 results, ScienceDirect: 125 results, SpringerLink: 84 results, Google Scholar: 370 results, Citeseer Library: 0 results, ACM Digital Library: 16 results and Sage Journals: 10 results. It must be considered that especially Google Scholar found literature which was already found in other databases. Therefore there were also some duplicate entries.

In total this gives a result of 667 hits.

In the first step all titles were scanned. After that, 138 results were left over. For example, results were removed which do not deal with collaborative robots or which focus already in the title on AR or VR or programming by demonstration.

These 138 results were further sorted out on the basis of the abstract and if necessary also on the basis of the conclusion. In this way 33 articles were selected as relevant to the state of the art of intuitive programming collaborative human-robot processes using graphical user interfaces or some kind of process modelling language or workflow. Finally, reading these 33 results showed that only 11 were really relevant for the diploma thesis.

The content of this selection (11 results) is briefly summarized in the next section.

3.2.1 Summary State of the Art

The search results show that, on the one hand, human-robot interaction (HRI) is becoming increasingly important in research. On the other hand, it can also be seen that the field of research is very large and very diversified.

The summary is divided into three parts. We will first look at current trends and recommendations, followed by various (intuitive) graphical user interfaces that researchers have already developed. As we will see in this section, human-robot collaboration is not or only very little considered in the development of intuitive GUIs, the focus of the last section is on the research that deals with HRC.

The importance of Cobots, UIs and Assistive Systems in the future

Wolfartsberger et al. describe in [125], [58] current developments, limitations, trends as well as technologies, which will be important and necessary for manufacturing in the future. The authors focus on two main topics, which in their opinion will be of great importance in modern manufacturing towards lot size one, these are **Collaborative Robots** on the one hand and **Assistive Systems** on the other hand [58]. This is based on the assumption that small batch sizes require a higher flexibility in production, which can be achieved by the use of collaborative robots. Due to the increased flexibility the workers need additional guidance. In the future in extreme cases each product in manufacturing could be different. This is where assistance systems come into play to support and relieve the workers.

Concerning human-robot collaboration (HRC) Wolfartsberger et al. see among others a necessity for the following two areas [58]:

- **New (intuitive) methods of programming/controlling robots**

Practical experience with industrial partners showed that the workers, for example, find it difficult to cope with the three dimensional coordinates, that are needed for positioning the robot. For this reason, new methods will be needed in the future to eliminate these problems. [58]

The authors specially emphasize the importance of interfaces, when creating new intuitive methods for programming cobots, with the following statement:

"One of the most important aspects in the field of collaborative robotics is the human-machine interface." [58, p.4]

Zaatari et al. [51, p.176] also underline the importance of user interfaces in their recommendations for future developments. It is also possible that future HRC systems will use other technologies, such as Artificial Intelligence (AI), to define tasks and parameters automatically, without user input. However, also in such scenarios UIs will be of great importance to alter or adapt the created workflows, tasks and parameters [51, p.176].

The fact that user interfaces are especially important for HRC may seem obvious, since the communication between humans and robots has to be done in some way. In contrast, the question of which type of user interface is particularly suitable for this purpose can't be answered so easily. To answer this, first it has to be clarified what is basically meant by the term "User Interface"? A detailed description of user interfaces (UIs) and especially graphical user interfaces (GUIs) can be found in Chapter 2 Theoretical Foundations.

As a short repetition, the ISO 9241-110 standard defines the term user interface as

"all components of an interactive system that provide information and controls for the user to accomplish specific tasks with the interactive system" [72]

This means a user interface is any interface that allows the user to interact with a system. This could be for example a graphical user interface or speech recognition, gesture recognition by cameras, haptic devices like wristbands or something else. In the simplest case a simple switch, for example a light switch, is already a user interface. Which user interface is best suited for humans depends on the application.

Looking at HRC and assuming that collaborative robots will be used for assembly especially in industrial environments, it turns out that voice control and body language are intuitive systems, but not reliable enough for industrial use cases [51, p.168]. Reasons for this are disturbing noises as well as the fact that these technologies are not yet mature [126], just think of the hit rate when using systems like Siri [127] or Alexa [128]. Therefore, it seems that at the moment graphical user interfaces and haptic devices/interfaces are better suited for industrial scenarios [51, p.168].

- **New methods to model collaborative assembly workflows**

Due to the increasing flexibility in assembly and the use of collaborative robots, Wolfartsberger et al. [58] identified a need and also a lack of currently existing methods, which enable the modelling of collaborative assembly workflows.

"There are different ways of modeling and optimizing specific workflows, but none of these methods consider HRC." [58, p.4]

By moving towards lot size one, it is necessary to create new modelling approaches that take both human and robot into account. Subsequently these workflows should be as flexible as possible. So, Wolfartsberger et al. [58] suggest that future research should also deal with modelling approaches that allow dynamic assignment of tasks between human and robot during runtime. [58]

With regard to assistive systems, it can be summarized that Mixed Reality (MR) and Virtual Reality (VR) will be of great importance in the future. However, current developments in these areas show that these technologies are not yet suitable for industrial

applications. For this reason, instructions on tablets as well as projections should be used for current developments. [58]

Zaatari et al. [51] also identified that there is currently a large gap between current developments of cobot programming in industry and research. To close this gap, research should focus more on cooperation with industrial partners and the creation of ready-to-use prototypes. In addition, more user studies should be conducted, to prove whether the developed systems and methods are really perceived by the users as simple and intuitive or not. One essential problem at the moment is that researcher assume that systems and interfaces are perceived by users as intuitive, without having conducted corresponding studies. [51]

After this short introduction about current developments, trends and recommendations, we will now look at different approaches, that try to improve the programming of cobots by the creation of intuitive graphical user interfaces.

Overview of intuitive GUI prototypes in research

In the following section we will look in detail at a number of research projects that aim to improve the programming of cobots.

Pentikäinen and Richard have investigated in their work [129] how to make the programming of cobots a bit easier and more intuitive for people with little programming knowledge. More specifically the cobot YuMi from ABB was used. The goal of the work was not to replace the existing programming environments of YuMi, but to create an additional graphical possibility to program the robot very easily. So the focus was on the creation of a graphical user interface and the consideration of how the interaction with the program could be particularly intuitive for the user. Finally the decision was made to create a web-based application with HTML5, CSS and Javascript for a tablet. The use of this variant has proved to be positive in that tablets and the use of touch screen gestures are very widespread and can be easily operated by anyone.

The GUI (see Figure 3.10) essentially consists of two areas. The left area is used to create/edit the workflow and the right area is used to visualize the workflow. Since YuMi is a collaborative robot with two arms, a representation by means of two swimlanes was chosen in both the left and the right area. By adding different tasks like pick or assemble the workflow can be created on the right by using touch gestures. When executing the program, the black horizontal bar on the right shows which task the cobot is currently executing. This presentation using swimlanes was perceived as good and intuitive by the users. Furthermore, the evaluation showed that this double presentation (creation on the left and visualisation on the right) seems to be a good approach. Nevertheless, for users working with the interface for the first time, this division of creation and visualization also caused confusion at the beginning. So the authors mention that for example a tutorial for users who work with the UI for the first time could be useful to improve the usability.

The final prototype of the user interface looks like following:

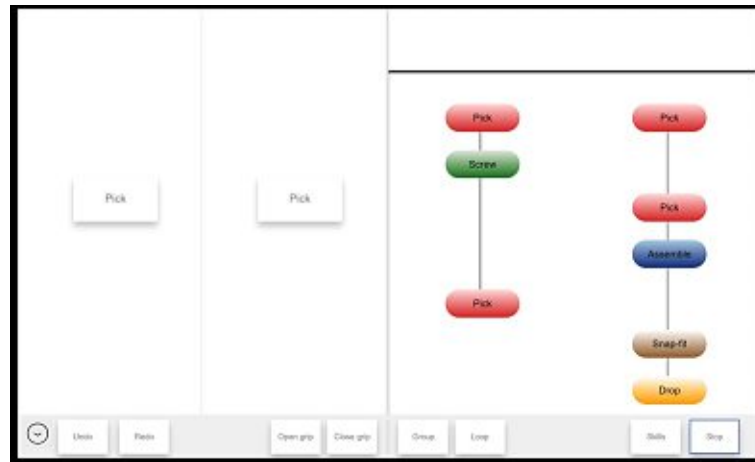


Figure 3.10: Intuitive GUI for the cobot YuMi [129]

The main conclusion of the thesis is that graphical programming is very suitable for people with little programming knowledge, however, it must be remembered that simplified graphical programming usually comes at the expense of flexibility. Furthermore this division into two areas, creation and visualization, was perceived as good by the users as well as the skill-based approach, which allows to program the collaborative robot by the usage of predefined skills like pick and assemble. [129]

Stenmark et al. [130] have conducted a study, also using ABB's YuMi collaborative robot, to investigate the extent to which a graphical interface for creating reusable parameterizable skills, combined with lead-through programming, simplifies programming for both experts and non-experts. For this purpose, a graphical user interface was created, which allows the use, combination and parameterization of robot instructions such as move and open gripper commands to create more or less complex skills such as e.g. pick&place. Figure 3.11 shows the graphical user interface, where the elements on the top left show the robot instructions and the pink buttons below show the created and already existing skills. An important aspect was also the compatibility. Since the prototype does not support all actions, the created program is stored in a form that allows further editing in the other existing programming environments of the robot. In this way, programs can be created very easily and quickly with the help of this interface and, if necessary (in the case of more complex tasks), can then be adapted further in ABB's traditional programming environments. The evaluation showed that the time saving for experts was 80% compared to the usage of existing programming interfaces/environments to solve the same tasks. Furthermore, a user study was conducted with non-experts. This study showed that the use of (reusable) skills significantly simplifies the programming of robots for this target group as well.

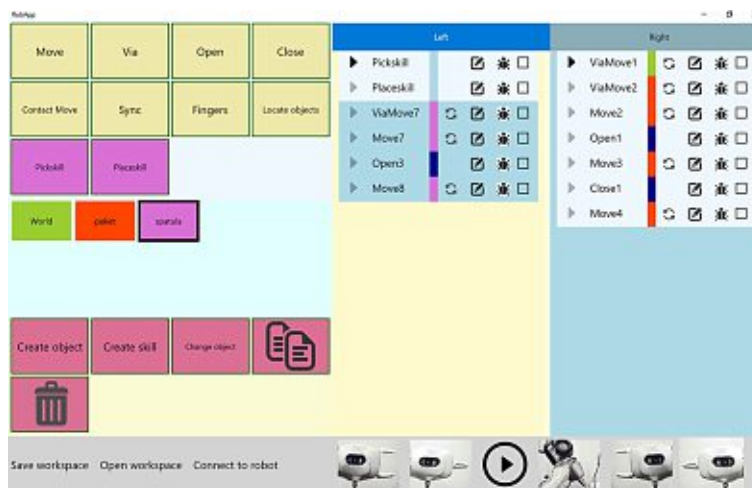


Figure 3.11: Intuitive skill-based graphical user interface [130]

Hanai et al. [131] describe a workflow-based approach, that allows to reuse and parameterize existing tasks. The model was implemented as a plugin for Chorenoid [132], which represents a GUI environment for robots, that provides basic robotic functions as well as easy expandability through plugins. The problem that the authors wanted to solve with their approach is to enable users without programming knowledge to create and alter collaborative workflows by the combination of existing skills, which can be parameterized. During the last years the teaching process of many cobots, which currently exist on the market, has been simplified. However, the problem is still how to reuse and alter existing tasks.

This problem could also be identified with the cobot Panda by Franka Emika [6]. For example, teaching a pick&place task is very easy. A big problem, however, is that with the smallest change to a component or the workspace, it is not really possible to change individual motions easily. For this reason the usual way to solve this is to retrain the affected parts of the workflow. In the long run this is quite troublesome.

The approach by Hanai et al. [131] distinguishes between experts and users. Experts are those persons who have programming knowledge and experience with robots. Therefore, the experts are responsible for creating the so-called reusable task models. For example a task model could be the task "Move a plastic bottle". Users are the people who search the database for existing task models, then they configure and combine them to workflows. So for example, a user wants to create a workflow for labeling bottles. A task model that could be used in this scenario would be, for example "Move a plastic bottle", because the bottle has to be transported to the "labeling jig" and afterwards transported into a transport box. The user can use this existing task and customize it with necessary parameters for this specific application, e.g. pick and place positions, bottle size, etc. As already mentioned, this approach was implemented as a plugin for Chorenoid [132], with different interfaces for experts and users.

3. STATE OF THE ART

Figure 3.12 shows the interface for the users, which consists of the following areas [131, p.8]:

- **Scene view:** The scene view shows a 3D visualization and animation of the programmed workflow.
- **Workflow view:** This is the section where the user combines the different task models and configures them. The combination of all tasks, connected by arrows, finally results in the workflow.
- **Search view:** The search view is used to search the database for existing task models created by experts.
- **Metadata view:** The metadata view uses text and images to describe the functionality of the task model. On the one hand this information is then used to help the user to understand what this task model does in detail and on the other hand this data is the basis for the search.

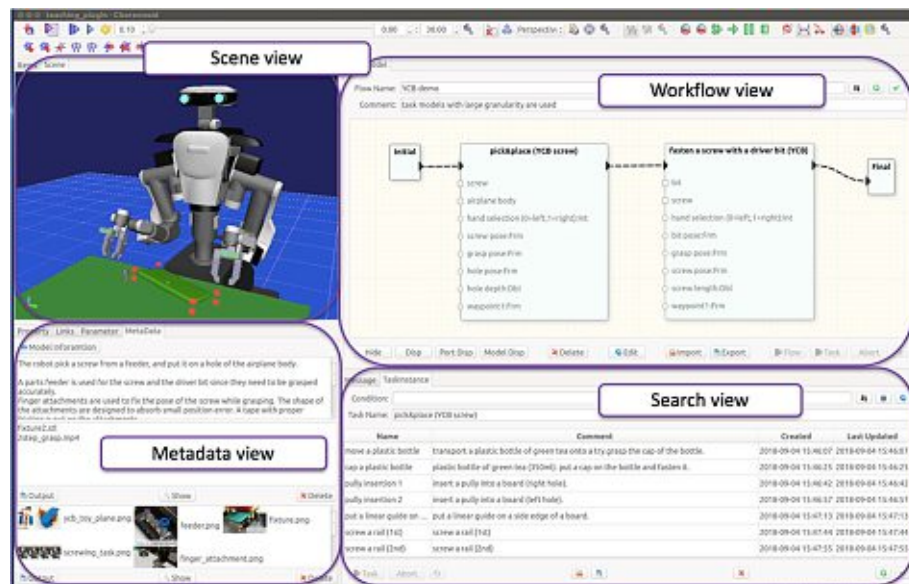


Figure 3.12: Workflow-based programming with Chorenoid [131, p.8]

With an experiment, using a UR3 cobot with dual arms, this prototype was finally tested. The use case was the assembly of the main wing of a toy airplane, consisting of one pick&place task which attaches the main wing, two pick&place tasks for the screws and two screw tasks. [131]

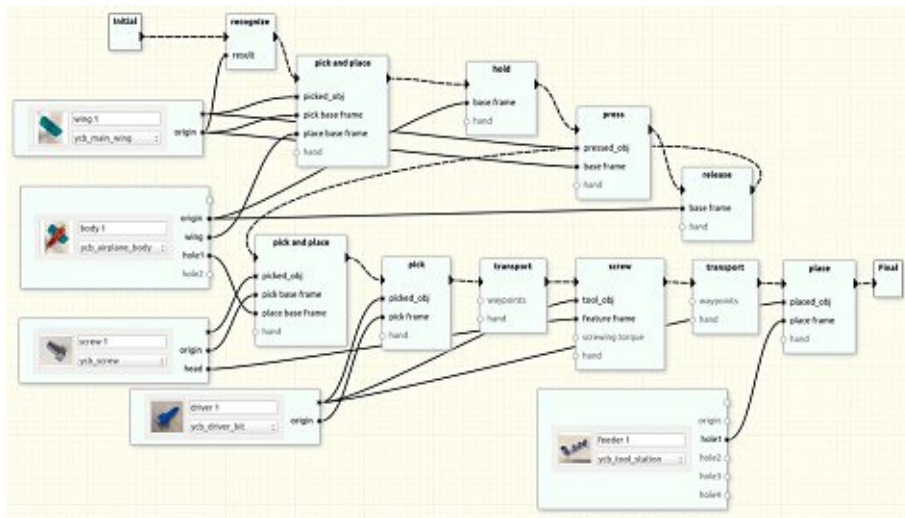


Figure 3.13: Example workflow for the assembly of the main wing of a toy airplane [131, p.13]

Figure 3.13 shows the final workflow, which seems very complex for only five tasks. Therefore, the authors mention that it is necessary to simplify the representation of the workflow in a next step [131].

Nevertheless, the approach looks very promising, as the problem of reusability and new teaching is really tedious and time consuming. A question that was unfortunately not answered is how good is the usability? For this purpose it would be interesting to conduct a user study to check, in how far this method is perceived as simple and intuitive for users without programming experience and to identify the limitations of this approach.

Ionescu and Schlund [56] describe an approach how to simplify the programming of cobots. A major point that the authors want to address with this method is the fact that cobots still do not seem to be economical to use. The authors identified several reasons for this. On the one hand, expert knowledge and experience is often necessary. This leads to high costs in planning and integration as well as in programming such systems. Second, there is little public information about the use of cobots (typical use cases, best practices). For this reason, Ionescu and Schlund present a three-layer approach, which should enable laypersons or in other words people with little or no programming experience to program cobots or adapt processes efficiently. Subsequently they propose to use this 3-layer model in so-called fablabs and makerspaces.

By the spreading of cobots (including simple ways of programming) in fablabs, people from other areas should also get access to this technology. This could democratize the programming of cobots what could lead in further consequence to the fact that these so-called makers could find new use cases and applications, which could later also be used in industry [56].

In the following we will now look at the approach of Ionescu and Schlund [56] in detail. The three programming layers differ in the possibilities (simple vs. complex use cases) that can be implemented as well as in the skills that are needed [56]:

- **Layer 1:** On this layer, people without programming experience (e.g. assembly workers) can create and edit workflows. An example would be the standard programming environment Desk of the cobot Panda by Franka Emika. Users can combine given tasks via drag and drop and thus program the collaborative robot. When used in industry, however, the use of only this application very quickly reaches its limits, since only relatively simple sequences can be programmed. For this reason, Ionescu and Schlund suggest two additional necessary layers to address this problem and to enable the possibility to create also complex applications for different scenarios in industry.
- **Layer 2:** In order to create more complicated programs, concepts such as loops, variables and branches are required. This is made possible by layer 2. Basic programming knowledge is required for this. In an industrial environment this would apply for example to an industrial engineer, who has a basic understanding of logical thinking and programming. The prototype of Ionescu and Schlund uses Blockly [133] for layer 2, which is a visual programming language developed by Google. Usually it is used to learn the basic concepts of programming, especially in schools. As Figure 3.14 shows, the user interface essentially consists of three areas. In the left area all objects are arranged, which can be used for programming, e.g. loops, if/else statements, variables and so on. These elements can be combined via drag and drop in the middle area, which represents the working area. In the right area it is possible to choose different programming languages like Javascript or PHP and the corresponding program code to the visual program appears. Summarized Blockly thus enables visual programming and helps to understand basic concepts and the corresponding program code.

As you can see in Figure 3.14, the following program was created. In the first block a variable "Count" is defined and set to the value 1. The next green block is a loop that repeats the part in the "do" section as long as the variable count is a value less than or equal to 3. Within the loop (in the "do" section), "Hello World!" is output and then the variable "Count" is increased by 1. In our example the loop is run through three times. If you select Javascript as programming language in the right section and then press Play, the browser will display three times the message "Hello World!". While creating program code requires programming knowledge, even laymen can quickly understand and map programming logic with the help of visual editors such as Blockly without knowing the concrete syntax of a specific programming language.

Ionescu and Schlund have extended Blockly and added a new category "Franka" with specific elements to the Blockly toolbar (see Figure 3.15). This makes it possible to use existing tasks of the desk environment in the Blockly code. Furthermore they

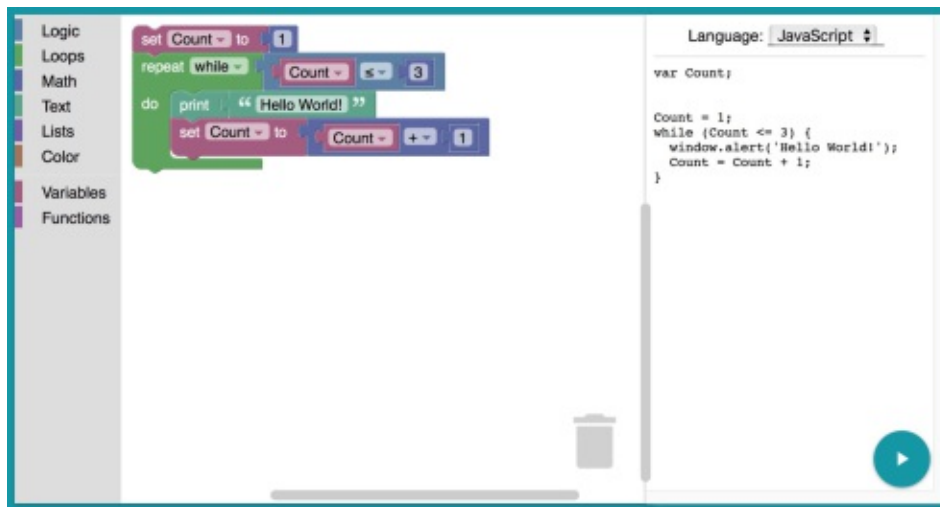


Figure 3.14: User Interface of Blockly [133]

have extended Blockly so that the created programs can be saved as bookmarklets. With the help of these bookmarklets the code created in Blockly can finally be executed in Desk. Bookmarklets are Javascript programs, which enable additional functions in the browser, such as changing the appearance of a website. From a technical point of view, code is injected into the existing website. This is used in this specific case to start and control the execution of tasks in Desk as you can see in Figure 3.16. This means that the program created in Blockly is saved as a bookmarklet (Javascript code) in the toolbar of the browser. If you open Desk and click on this bookmarklet, this code will control the Desk interface. In this way, with the help of basic programming knowledge, more complex use cases can be implemented in a relatively simple way.

- **Layer 3:** Finally, Layer 3 is used to enable special, extended functionalities. At this level, very good programming skills are required, which is why typical users of this layer are software engineers. The idea of this 3-layer approach is to cover as many use cases as possible with the help of the first two layers, thus saving the need for a software engineer. Only for very special use cases (e.g. the implementation of keyboard input) this should become necessary by using the third layer. This makes it possible in the shown prototype to control via keyboard input whether the user or the robot should execute a specific task.

So the idea of the approach is to combine the three layers to provide the highest possible flexibility, whereby layer 1 does not require any programming knowledge, layer 2 requires basic programming skills (e.g. understanding of loops, if/else statements and so on) and layer 3 requires very good programming skills. However, layer 3 is only needed to implement special advanced functionalities, like keyboard interaction. In addition,

3. STATE OF THE ART

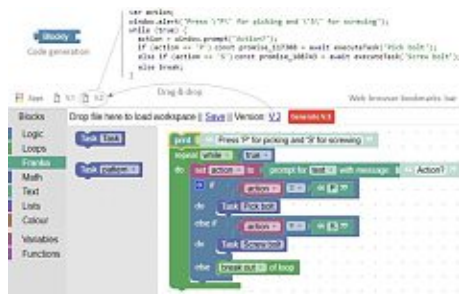


Figure 3.15: Extended Blockly App [56, p.96]



Figure 3.16: Injecting Blockly Code into Desk [56, p.97]

through the collaboration of the users of the different layers (workers, industrial engineers and software programmers) the knowledge of all should be combined in the best possible way. [56]

Another method how to make the programming of collaborative robots easier and more intuitive was proposed by Blume et al. [134]. With their approach they have shown that by combining different existing technologies, such as visual programming, speech and object recognition, teach-in, etc. even users without programming experience are able to program cobots easily. The goal was the conception and implementation of a flexible HRI cell, which allows the packing of products in different sizes and shapes. Specifically, they used the following technologies: a cobot from FerRobotics in combination with a flexible gripper by Tekniker, Microsoft Kinect for activity recognition of the worker, a GUI based on Blockly, speech and gesture recognition and a software for object recognition by Profactor. Another goal was to develop a graphical user interface, which enables users with little or no programming experience to create new workflows and modify existing ones. The user interface (see Figure 3.17) uses Blockly and was extended so that users can create new blocks using teach-in and voice control. The order of these blocks can be modified in the user interface, using e.g. a tablet, via drag and drop. Finally, a study was carried out to evaluate the usability of the system. For this purpose the System Usability Scale (SUS) was used. All participants were able to solve the exercise and the system achieved a SUS score of 89.4%, which speaks for a particularly good usability. Furthermore, the statement "90% stated that it was easy or very easy to program the workflow." [134, p.86] also underlines the outstanding SUS score.

All of the above-mentioned works attempt to make the programming of collaborative robots more intuitive by the usage of different approaches. Changing requirements in manufacturing, especially the trend towards small lot sizes and an increasing number of different variants of a product, require as already mentioned new flexible ways of

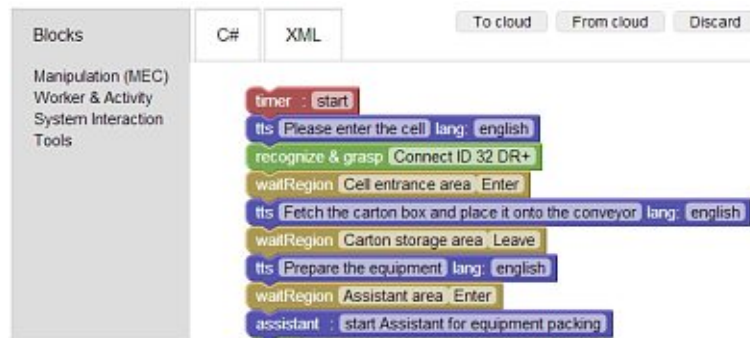


Figure 3.17: Cobot programming with Blockly [134]

assembly [125, p.289]. One possibility to achieve this is human-robot-collaboration (HRC), which means that humans and robots work collaborative together - they share their work. Nevertheless, the focus of all approaches mentioned so far is only on improving the programming of the cobot. However, humans also play an important role in the field of HRC, therefore we will also have a look at current research, which involves the human being, in section three. Before that a short summary of two studies follows that have evaluated and compared the usability of systems which are already available on the market.

Usability evaluation of different GUIs

Schmidbauer et al. [57] investigated three systems for programming collaborative robots and evaluated them with respect to usability in the course of an experiment. In detail, the following three robots were compared (pictures and a detailed description of the interfaces can be found in Chapter 3.3):

- UR5 by Universal Robot: The programming was done via teach pendant.
- Panda by Franka Emika: Compared to most of the other manufacturers on the market, the Panda is programmed via a web interface called Desk, which can be used in a browser on a laptop or tablet for example.
- CR-7iA by Fanuc: Similar to the UR5, the CR-7iA is programmed via teach-pendant.

The System Usability Scale (SUS) by John Brooke [29] was used to measure and compare the usability of the three systems. The average SUS score of all systems provided a value of 59%. As the authors state, this score can already be seen as an indicator that such systems have usability problems. For explanation: a SUS score below 60% is usually an indicator for bad usability, while in the range between 60-80% the usability of the system is ok. A score above 80% generally means good/excellent usability. The systems in detail delivered the following results: Universal Robot UR5 55.4%, Fanuc CR-7iA

52.3% and Franka Emika Panda 70.6%. This detailed result of the individual systems shows that both robots with teach-pendant delivered a worse value by far compared to the web-based interface of the Panda. [57]

Ferraguti et al. [135, p.1075] also evaluated the SUS score of the Universal Robot UR5 and Franka Emika Panda. The results were 62.4% for the UR5 and 85.4% for the Panda. The values are much higher compared to Schmidbauer et al. [57]. Nevertheless, the results of both experiments show that the Panda has a higher usability compared to the UR5, which ranges from acceptable to good. While the usability of the UR5 is in need of improvement.

Further interesting results from Schmidbauer et al. [57] were also that the participants were dissatisfied with the use of the teaching pendant and therefore the authors recommend the use of an user interface on a screen (as in the case of the Panda UI) or the combination of several options, for example using a graphical user interface in combination with speech recognition.

The following statement also underlines the importance of ("good"/intuitive/simple) UIs, also in the area of programming cobots:

"From a learning factory's perspective, it can be stated, that quality UI design increases the acceptance of a cobot as an assistance system in manufacturing and serves as a facilitator for the training of assembly operations – even for non-experts, laypersons and people that are usually not interested in industrial processes." [57, p.403]

These studies show that the usability of the GUI of the cobot Panda is by far better than the teach pendant solutions of the other manufacturers. There are several possible reasons for this. On the one hand this could be due to the way the cobot panda is operated (web-based PC/tablet solution vs. teach pendant). On the other hand possible reasons for this result could also be the simple design of the Desk UI as well as the task-based approach, that is used.

(Workflow-based) Approaches for programming cobots

In the last section, we will now look at research work that is not only related to the programming of the cobot and also considers the human tasks.

Mangler et al. presented in their work [136] a framework, called *centurio.work*, which enables the orchestration of manufacturing processes on the shop floor level. For this purpose a BPMN process engine, inspired by the Reference Architecture Model Industry 4.0 (RAMI) [137], was developed. As can be seen in Figure 3.18, the production processes are mapped on the basis of the Business Process Model and Notation Language (BPMN). Subsequently, the developed process engine takes over the execution of the process.

The BPMN process model in Figure 3.18 on the left shows a use case in manufacturing where the worker starts the production of a component by scanning a product code (1). The curved brackets (2) define conditions (like `data.state == 'Cancelled'`) and thus enable the mapping of decisions. Subsequently, various sub-processes are executed (3). An example of a sub-process is represented by (4). The example shows very well the scope of *centurio.work*, especially how detailed the processes in production can be mapped.

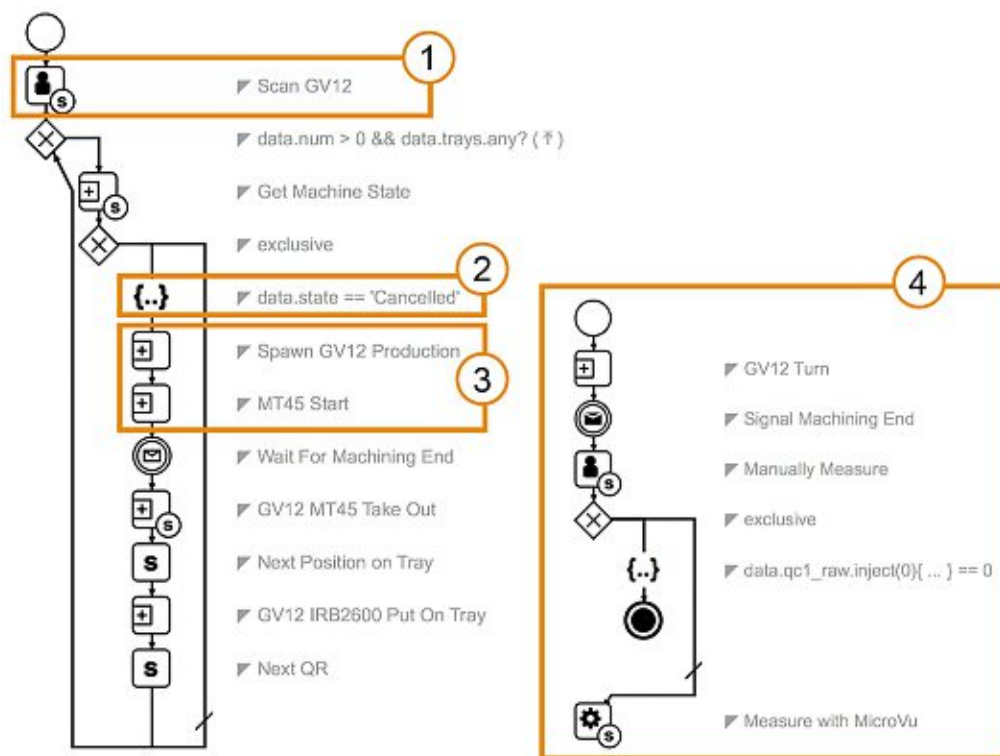


Figure 3.18: Example of a *centurio.work* BPMN Process (adapted from [136])

Schönberger et al. present in their work [138] a method called Human Robot Time and Motion, short HRTM, which enables the modelling of collaborative workflows between humans and robots. The focus lies on the integration of the human being. So, the goal of this approach was to enable the creation of collaborative workflows regardless of who is executing the tasks - human or robot. The authors first examined existing modelling languages for workflows, specifically: BPMN (Business Process Model and Notation), EPC (Event driven Process Chain) and AD (Activity Diagram). However, none of the three methods is suitable out of the box for modelling MTM (Methods Time Measurement) and RTM (Robot Time and Motion).

Excursion: Methods Time Measurement (MTM)

Methods-time-measurement (MTM), originally developed by Maynard et al. [139], is a method for performance rating for manual tasks. It breaks down the course of human movements into small parts (so-called motions, e.g. move, grasp, release) and uses empirically determined values to define time standards [140, p.99]. The times of the different motions are then added up and time targets are created for the individual tasks as well as for the whole process [140, p.99]. On the one hand the use of MTM increases the efficiency of the production and on the other hand, if implemented correctly, ergonomics are also taken into account, which means that the use of this method is also beneficial for the employees [141, p.981].

Excursion: Robot Time and Motion (RTM)

To compare the task performance between humans and robots, Paul and Nof have developed a methodology called Robot Time and Motion (RTM). In their work they have tried to model robots with the use of MTM. This is possible in principle, but requires a different set of work elements. For this reason they developed RTM, which represents MTM for robots. [142]

A major problem is that one requirement of MTM and RTM is that certain basic elements, which can be executed by robots as well as by humans, must exist. However, BPMN, EPC and AD are very general modelling languages, which should be as flexible as possible. For this reason, this new approach, HRTM (Human Robot Time and Motion), was developed, which essentially consists of 17 elements, divided into 5 groups: Motion elements, Tool elements, Sensing elements, Collaboration elements and Time elements. For the visualization BPMN was then used as a basis and extended. An example of a simple collaborative workflow, modeled with the HRTM approach is shown in Figure 3.19. [138]

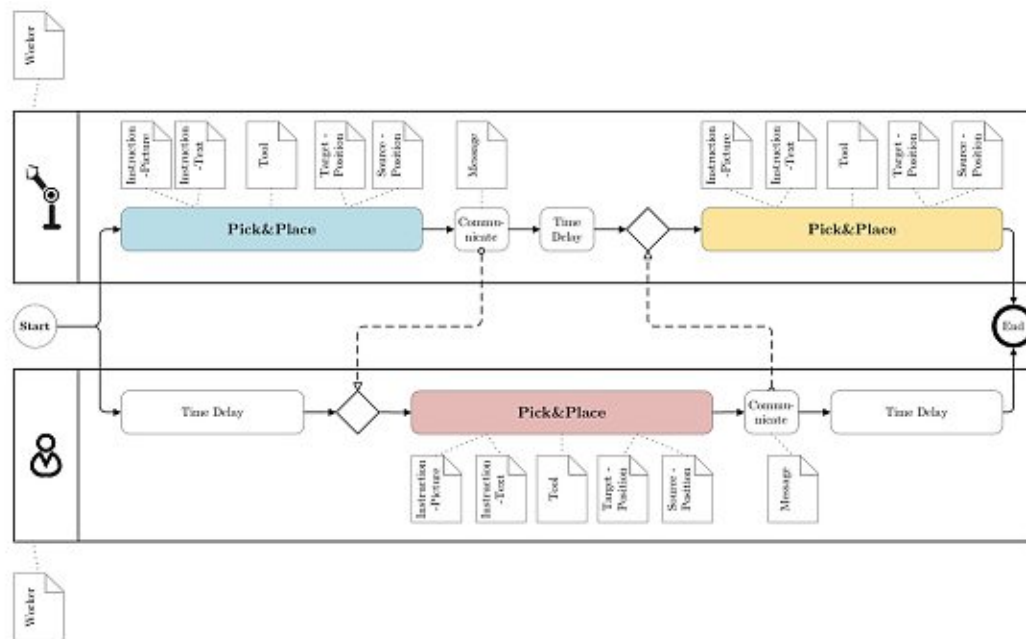


Figure 3.19: Collaborative workflow modeled with HRTM [138]

As shown in Figure 3.19 so-called pools and swimlanes are used to represent the tasks of humans and robots. A detailed description of BPMN as well as the concept of pools and swimlanes can be found in Chapter 2 - Theoretical Foundations. In this example the upper pool represents the robot and the lower one the human being.

The special thing about the approach from Schönberger et al. [138] is that the pick and place tasks between human and robot could be exchanged simply by moving them to the other pool. This is possible because each Pick&Place sub-workflow contains all the information that the robot needs to execute as well as the information (for example, a picture) that the human being needs to execute the task. As one way to implement this approach in practice, the authors suggest that the robot's activities are directly forwarded to the robot controller while the human tasks are converted into HTML and displayed in a web browser.

Lindorfer et al. presented a Meta-Meta-Model [143], which also takes into account the modelling of human tasks. As in the solution by Schönberger et al. [138] MTM and RTM are used as a basis. Additionally, a possibility was integrated which allows decision based execution offline as well as at runtime. This functionality allows, for example, to model several variants in a workflow (multiple paths) and to make decisions at runtime, which path should be executed. Figure 3.21 shows the Meta-Meta-Model, called ADAPT (Asset-Decision-Action-Property-Relationship).

Before we look at ADAPT in detail, a short digression follows to clarify the terms Meta-Meta-Model, Meta-Model and Model.

Excursion: Meta-Meta-Model/Meta-Model/Model

A model is a simplified representation of the reality [144, p.1]. Depending on necessity, reality is abstracted and thus omits things that are not important for observation. An example would be the modelling of a collaborative process. A possible model to describe an assembly process could look like Figure 3.19 represents. This model describes the workflow in detail and contains necessary information for human and robot how to execute the tasks. But a detailed description of the workplace for example is not part of this model, because it has no relevance for describing the workflow. In comparison, the same example could be modeled using a CAD tool. The result would be a detailed 3D representation and simulation of man, machine and workplace. The level of detail is much higher for the latter, but this additional information is not necessary for the purpose of the first model. By omitting unimportant details, also known as reduction feature, the models become simpler and the focus can be directed to the essential things [144, p.2].

Looking at the examples above, it can be seen, that one specific model is the representation of one use case in the real world. It's possible to create different models for different use cases, but it requires an overarching set of rules that determines which elements are available and how a conforming model can be created. So to speak, a language is needed, a so-called modelling language or also called meta-model, which defines the basic elements, their relationships and how to create a conforming model.

"Therefore, one can define models of the reality, and then models that describe models (called metamodels) and recursively models that describe metamodels (called meta-metamodels)." [144, p.15]

The meta-model is therefore a model that describes all models of this specific modelling language. The meta-meta-model is then another generalization of the meta-model. The meta-meta-model describes all possible meta-models, which can be created. Now one could come to the conclusion to continue this process of model creation endlessly: to create meta-models, meta-meta-models, meta-meta-meta-models and so on. This is also possible in theory, but "... it has been shown, in practice, that meta-meta-models can be defined based on themselves, and therefore it usually does not make sense to go beyond this level of abstraction." [144, p.15].

For a better understanding of the interrelationships, see Figure 3.20. The different levels are also referred to as the levels M0-M3, where M0 represents the real elements of the world, in our case e.g. human workers, robots, pick and place tasks. M1 denotes concrete models like Figure 3.19, which reflect the real world. The layer M2 describes the meta-models, as mentioned above, this are the modelling languages, examples would be UML, BPMN. Finally, M3 represents the meta-meta-models, which are the models with the highest level of abstraction that define the meta-models.

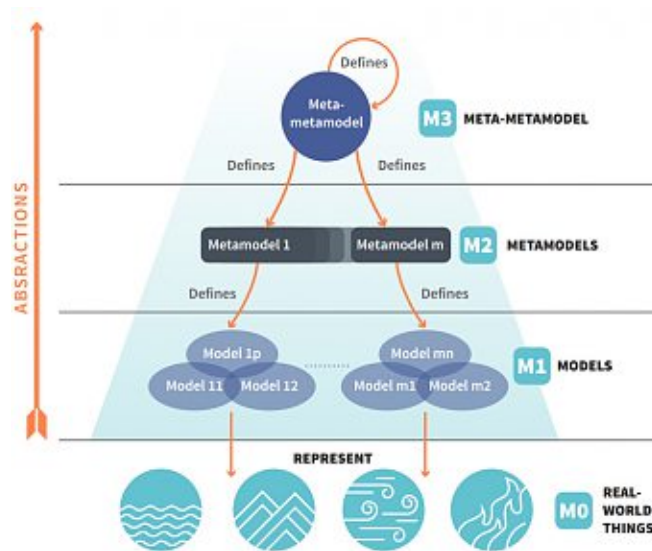


Figure 3.20: Models, Meta-Models, Meta-Meta-Models 145

Conclusion Models/Meta-Models/Meta-Meta-Models

Summarized and explained with a practical example and the M0-M3 layers this means if we look at the example above, Figure 3.19 represents the model (M1) for a concrete, simple collaborative human-robot process with three pick&place tasks.

Imagine this process in the real world, this could be in a factory, for example, where the assembly is ultimately carried out according to exactly this scheme. This is then the level M0.

The modelling language, which provides all elements (branches, tasks, connectors) and defines how the model has to look like to be conform, is then the modelling language/meta-model (M2).

Let us assume that this modelling language only allows the modelling of pick and place tasks, because this is perfectly sufficient for collaborative assembly in a specific domain. But in another domain more than pick&place is needed. Then a meta-meta-model (M3) could be used to develop a new domain-specific modelling language (meta-model (M2)) that is capable of modelling e.g. pick&place tasks as well as screw tasks. Nevertheless, both meta-models (M2) would be conform to the same meta-meta-model (M3).

On the next few pages this should become even clearer when we look at the ADAPT meta-meta-model (see Figure 3.21) and the specification of one possible meta-model (see Figure 3.22).

As you can see in Figure 3.21 the ADAPT Meta-Meta-Model consists of only five elements [143](#):

- **Action:** This element represents tasks in general, it could be a manual task or a robot task. Who executes the task is irrelevant at this point. An example would be a screw task, regardless of whether it is performed by man or machine. To come back to the connection between Meta-Model and Meta-Meta-Model, which actions/tasks are finally available in the modelling depends on the Meta-Model. As explained in the section Conclusion Models/Meta Models/Meta Meta Models, a Meta-Model could, for example, only define pick and place tasks. Important is the connection of the action with itself. This means that an action can consist of several actions. This makes it possible to create so-called sub-tasks. Pick&Place tasks are a classic example of this. A Pick&Place task consists usually of several actions. These could be grouped together to form a superior Pick&Place Task to improve clarity (see Figure 3.19).
- **Asset:** Assets are used for any kind of information. For example, a human (worker) could represent an Asset, which can be used to model that a certain action must be performed by a human. Also, information needed to execute an action can be represented as assets, e.g. work instructions, videos, etc.
- **Relationship:** As you can see in Figure 3.21, Action and Asset are connected by the Relationship element. This element can be used to express that e.g. an Action must or can own a certain Asset. An example would be: a certain task e.g. a screw task can only be executed by a human, because the robots that are used (e.g. Panda by Franka Emika) cannot execute a screw task. So you can use a Relationship element which defines that the screw task must have an Asset "worker". Another example: An Action can, but must not, contain a video. This can also be modeled using the Relationship element. As Lindorfer et al. [143](#) describe, every Relationship element also has a boolean condition. This can be used to incorporate logic during the development of the process or at runtime. With the help of these conditions, it can be checked if certain conditions are fulfilled. This can be used to check whether a certain condition is fulfilled for a specific Relationship element, the data for this could be provided by a sensor. Only if the condition is fulfilled a specific action would be executed. For example, a sensor could provide information about whether the robot is available or not. If yes, the action should be performed by the robot, if no, by the worker. With the help of these conditions, the action flow of the process can be controlled at runtime.
- **Property:** Properties are used in the Meta-Meta-Model to describe the Action, Asset and Decision elements in more detail. An example could be the Action "Place". It is necessary to store the information where the element should be placed. This can be achieved with the help of the Property element.

- **Decision:** Decision elements are used to model workflow conditions. For example, a branching, depending on the evaluation of a specific condition, a certain path should be executed - also known as a classical if/else condition. The data for checking the condition can either come from information stored during the creation of the process or during runtime from assets (e.g. sensor, camera, user input).

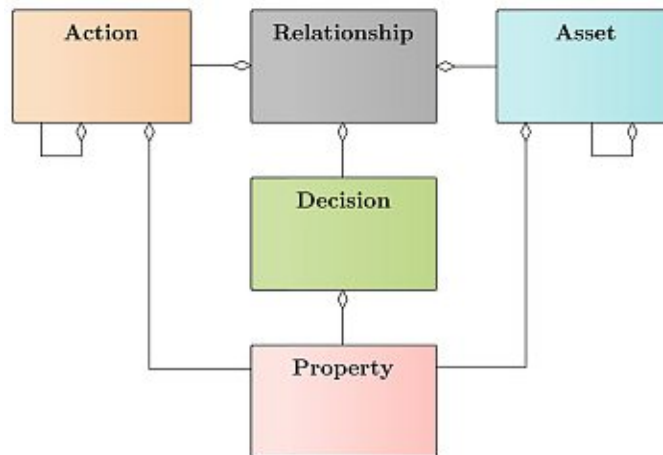


Figure 3.21: Meta-Meta-Model ADAPT [143, p.561]

As shown in Figure 3.22 this is one possible Meta-Model that uses ADAPT as Meta-Meta-Model. This Meta-Model [143] consists of only seven Action elements like Move, Grip, Place. The assets that represent additional information, as described above, are in this specific example: Worker, Skill profile, Work instruction, Image, etc. This Meta-Model shows very well which modelling languages can be created based on the Meta-Meta-Model.

The Relationship "Requirement" is used in the example to express that an Action requires a specific Asset. A concrete example could be defined as follows: The Action "Grip" requires a certain skill to be executed, so a specific skill profile (Asset "Skillprofile") is required. Only if the Asset "Worker" has an appropriate Asset "Skillprofile", he or she can perform the "Grip" Action. [143, p.561]

This specification is possible because an Asset can consist of several Sub-Assets. There a "Worker" Asset can be created that contains various Sub-Assets, such as the Asset "Skillprofile". Examples of Properties used in this Meta-Model are "Coordinate" and "Text". "Coordinate" can thus be used to store additional information about the coordinates for various actions. Or the "Text" Property can be used for storing additional text for e.g. the work instruction. As Decision elements "InputDecision" and "VisionDecision" can be found in the Meta-Model. "InputDecision" would thus be a possibility in the workflow to model conditions, whereby a different path is executed depending on the input.

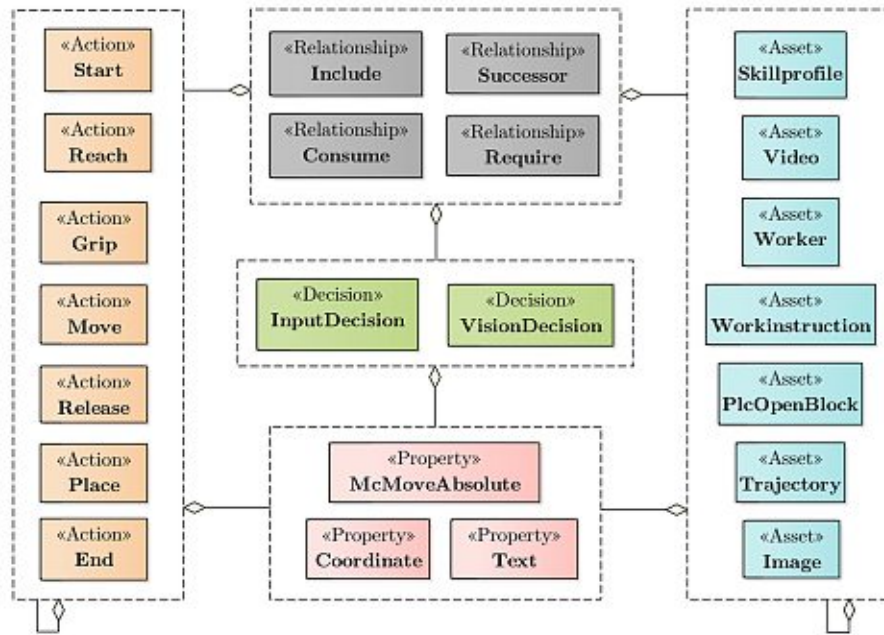


Figure 3.22: Example of a Meta-Model based on the ADAPT Meta-Meta-Model [143, p.561]

In summary, Lindorfer et al. [143] have developed a Meta-Meta-Model, which allows the creation of Meta-Models, that take both humans and robots into account. They also showed how an exemplary meta-model based on ADAPT (Asset-Decision-Action-Property-Relationship) could look like. By integrating validity conditions into the element Relationship, they showed that it is also possible to map so-called conditional action flows. This means that the workflow can be controlled during execution by conditions fed by asset data, e.g. by sensor, camera or user input. Based on this, they have also created a software prototype called ADAPT designer, which allows the intuitive creation of meta-models and subsequently the creation of workflows (see Figure 3.23). The designer essentially consists of five sections: Modeler, Gantt, Logger, Architect, Workspace- and Adapt Explorer. The basis is the modeler in the middle, which is based on BPMN. The Architect is located in the lower left area. This is where the Meta-Model is defined. It was also shown that platform-specific visualizations could be generated on the basis of the XML model from the ADAPT designer. The authors propose to use code generators subsequently to generate code to control the robot on the one hand and to visualize the work instructions for the worker (for example using HTML or augmented reality) on the other hand.

The ADAPT Meta-Meta-Model [143] provides a good basis for the development of platform-independent collaborative human-robot systems and is a really good top-down approach. This means, that first a domain-specific meta-model is defined and the workflows are then modelled on the basis of this DSL (Domain Specific Language).

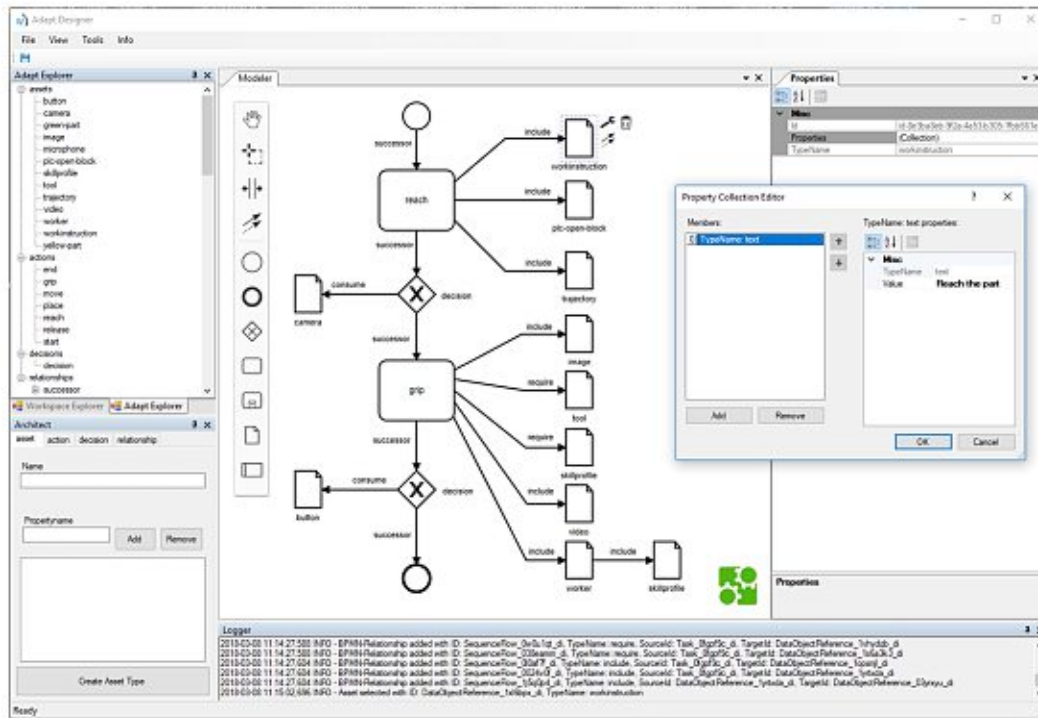


Figure 3.23: Interface of ADAPT Designer [143, p.563]

Nevertheless, looking at the use case, that the model should be used to create collaborative workflows with e.g. the cobot Panda. Depending on the robot, which is used in the scenario, and the visualization that will be used for the worker assistance system, this approach requires a high programming/adjustment effort, assuming that there is not already a code generator for exactly this robot and this type of visualization. The creation of the corresponding code generators then requires professional programming knowledge and thus causes high efforts and costs. In addition, good modelling skills are required at least for the creation of the Meta-Model. A correct and well elaborated Meta-Model is the basis for all workflows, which are created with this DSL. Errors/modifications in the Meta-Model that are detected late can cause numerous changes in existing models, which in turn can cause high effort and costs. It must be remembered that meta-modelling is not a trivial task and requires lots of knowledge and experience. The current implementation can be seen as offline programming. With the software prototype it is necessary to create first the workflows. These workflows are then converted, by the use of so-called code generators, into code to control the robot and visualizations, that are appropriate for the humans. In order to make the development as easy as possible for the end user, it would make sense to use instead of offline programming online programming, as the authors describe in the future work section [143, p.563]. Afterwards what the approach also does not investigate is the question whether this type of programming using workflow-based programming is perceived as intuitive and easy for the end users or not.

3.3 Cobot programming in practice

In this chapter a number of different solutions for programming cobots, which are already existing on the market, are presented. In almost all cases these are manufacturers of robots or collaborative robots, that offer their own programming environment/user interface for their products. The reason for this is very obvious. The various manufacturers of collaborative robots have to offer their customers a more or less simple way to program them. Each manufacturer creates his own solution. There are two reasons, why this is the case. On the one hand this is necessary, because at the moment there exists no software on the market, that can control all cobots of the different manufacturers. On the other hand this results in a certain company dependency for the customer, also known as vendor lock-in, which in turn is advantageous for the manufacturers.

3.3.1 Universal Robots

Universal Robots [146] is the market leader in the field of collaborative robots [147]. The product range comprises four cobots, which differ mainly in their size and the maximum possible payload. All robots are programmed via teach pendant as it can be seen in Figure 3.24 and 3.25. To program the cobot the user defines so-called waypoints. The created program is displayed in a tree structure as shown in Figure 3.24. The left section (see Figure 3.24) shows the different elements that can be inserted into the program tree, such as Move, Waypoint or Wait. By clicking "Edit pose" the interface shown in Figure 3.25 appears, which consists of three sections. To specify the exact position of the waypoint, the user can control the robot via touch using the arrows in the left section. The middle area shows a 3D representation of the cobot and in the right section the parameters can be edited in detail.

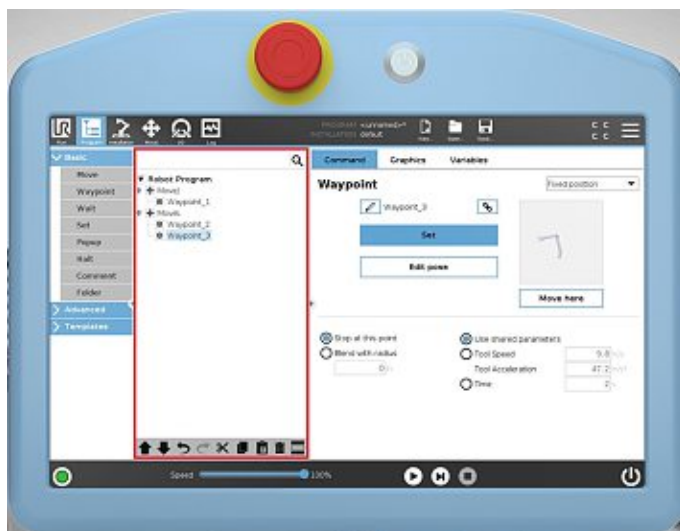


Figure 3.24: Programming interface of Universal Robots [148]



Figure 3.25: Programming interface of Universal Robots [148]

One aspect of the interface that stands out is the fact that the human being is not directly taken into account in the programming. The only way to map human tasks is by using the Wait command. As it can be seen in [57], [135] the usability of the Universal Robots user interface, based on the the System Usability Scale (SUS) [29], scores rather poorly with a value between 50 and 60%, which clearly shows that there is still room for improvement.

3.3.2 Kuka

Kuka [149] uses in the field of programming collaborative robots a very similar approach to Universal Robots. Currently the product range includes only one cobot, the Kuka iiwa. However, the company is working on a second collaborative robot, the Kuka iisy, which is not yet available on the market [150].

The Kuka iiwa is also programmed by using a teach pendant and creating waypoints (see Figure 3.26). Since it is very similar to the user interface of Universal Robots, the GUI will not be further explained.

Although the Kuka iisy is not yet available, there are some information already available on the web. According to the manufacturer, the programming via tablet should be particularly simple and intuitive [151]. In addition, there should be several operating levels, adapted to the knowledge of the user. If the user is an expert, he/she can program the robot with e.g. Java. People without any previous knowledge should be also able to program the cobot very easily with the intuitive graphical user interface [151]. As you can see in Figure 3.27, it seems that the tree structure will be replaced by a flow-based diagram. Furthermore, the GUI looks a bit more tidy and modern than the user interface of the Kuka iiwa.



Figure 3.26: Programming interface of Kuka iiwa [152]



Figure 3.27: Programming interface of Kuka iisy [153]

3.3.3 Fanuc

Fanuc is a pioneer in the field of automation and offers the widest product range with over 100 robots [154], seven of them are collaborative robots. The cobots are programmed graphically by the use of a teach pendant. The CR models are programmed in a similar way to the cobots of Universal Robots and Kuka by creating waypoints (see Figure 3.28 and 3.29). The GUI appears very clear and as it can be seen in Figure 3.29 a timeline is used instead of a tree structure.



Figure 3.28: Programming interface of Fanuc (CR models) [155]



Figure 3.29: Programming interface of Fanuc (CR models) [155]

The programming of the CRX models, which were released at the end of 2019, is little bit different and should be even easier and more intuitive (see Figure 3.30). The GUI follows an icon-based programming approach and consists of three areas: a 3D visualisation of the robot, a horizontal timeline and in the lower area are the icons for programming. The user can move these elements via drag and drop into the timeline and thus create the program.



Figure 3.30: GUI of Fanuc (CRX models) [156]

3.3.4 Fruitcore Robotics

Fruitcore Robotics [157] aims to combine the advantages of classic industrial robots (speed and precision) and collaborative ones (easy programming). The result is an industrial robot that can be programmed very easily via teach pendant. The GUI is similar to the solutions of the above mentioned manufacturers by means of 3D visualisation and the definition of waypoints. Furthermore, the interface is generally very clear, well structured and can be tested online for free [158].



Figure 3.31: Programming interface of Fruitcore Robotics [157]



Figure 3.32: GUI of Fruitcore Robotics [158]

3.3.5 Franka Emika

The manufacturer Franka Emika [6] currently only has one (collaborative) robot in its range, called Panda. For programming, the web-based application "Desk" is used. The difference to the teach pendant examples of the other manufacturers mentioned so far

is that the programming is not done by means of a teach pendant and the usage of a task/skill-based approach instead of defining waypoints.

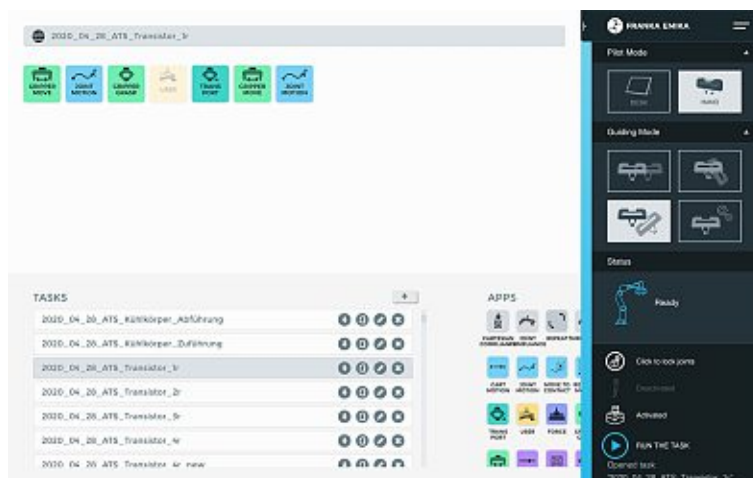


Figure 3.33: Programming interface (Desk) of Franka Emika (own Figure)

The user interface (see Figure 3.33) essentially consists of three areas. In the lower right section all "skills", called apps, are displayed, which can be moved by drag and drop into the work area at the top. Typical apps are "Move", "Grasp", "Transport Motion", but also more complex ones like "Typing", "Press Button" are available. Each app has different parameters for configuration. By combining the apps in the workspace a task can be created, which represents the overall goal like Pick&Place Object A. An overview of all programmed tasks is shown in the lower left area.

As shown in [57], [135] the usability is much higher compared to the teach pendant UIs of other manufacturers like Universal Robots and Fanuc.

3.3.6 Techman (TM) Robot

Techman Robot [159] offers various programming solutions for its collaborative robots, such as a vision system that uses a camera to recognise objects, a CAD-based modelling software, a ROS API and a flow-based programming environment (for simple programming). In the following we will take a brief look at the flow-based programming environment TMflow (see Figure 3.34). Programming can be done both online and offline and the GUI consists of only two areas (toolbar and workspace). The flow chart is created by combining the elements from the toolbar (e.g. Point, Wait for, If, Move) in the workspace. In addition, there is also the possibility of defining the robot position by using a 3D simulation. What is noticeable with the elements in the toolbar is that the programming is done by using basic robot functions like move, set, listen, wait for, etc. No skills such as pick, transport, press button are provided. The programming is therefore rather function and not skill oriented and follows the icon-based programming approach as described in Chapter 3.1.



Figure 3.34: Flow-based GUI of Techman Robot [160]

3.3.7 ABB

ABB [161] is also one of the big players in the field of industrial robots. Since 2015 the product range also includes a collaborative robot called YuMi. A special feature compared to the cobots of other manufacturers is the possibility to use the cobot with two arms (see Figure 3.35).



Figure 3.35: ABB's cobot YuMi available with one or two arms [162]



Figure 3.36: Programming YuMi online via teach pendant [163]

ABB uses the text-based programming language RAPID for programming their robots. As it can be seen in Figure 3.37, this method is not suitable for non-professionals. For this reason, a graphical programming method, based on Blockly, has been developed with the application "Wizard Easy Programming", which should also enable people without programming knowledge to program the cobot. Figure 3.38 shows the same program as in Figure 3.37, but programmed graphically with "Wizard Easy Programming". Programming can be done either offline using ABB's RobotStudio or online via teach pendant (see Figure 3.36). In the background the Blockly based YuMi programs are converted into RAPID code. The advantage of this is that e.g. an expert could program the collaborative robot using RAPID and the user can subsequently change/alter the same

3. STATE OF THE ART

program graphically using "Wizard Easy Programming". This makes the programming of the cobot flexible and suitable for both experts and laypersons.



Figure 3.37: Text-based programming with RAPID [164]

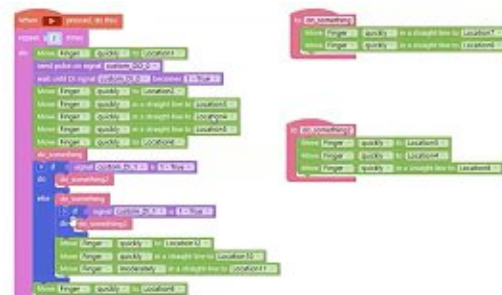


Figure 3.38: "Wizard Easy Programming" [164]

3.3.8 Drag&Bot

In comparison to the applications shown so far, Drag&Bot [165] supports a wide range of classic but also collaborative robots from different manufacturers such as Kuka, ABB, Fanuc and Universal Robots. Programming, which uses a task-based approach as a basis, can be done online as well as offline (see Figure 3.39). Offline the user has the possibility to use additional tools like a 3D simulation (see Figure 3.40). As can be seen in Figure 3.39, the user is provided with various function blocks (skills), which are combined into tasks via drag and drop. The clear advantages of Drag&Bot are the support of a multitude of robots of different manufacturers and the task based approach.

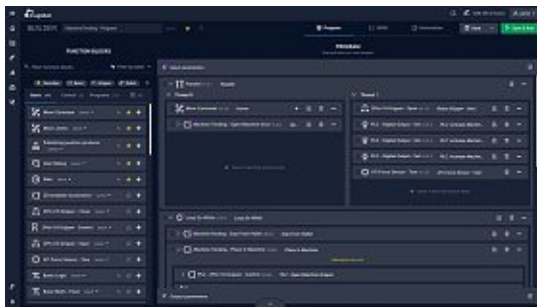


Figure 3.39: Programming interface of Drag&Bot [165]

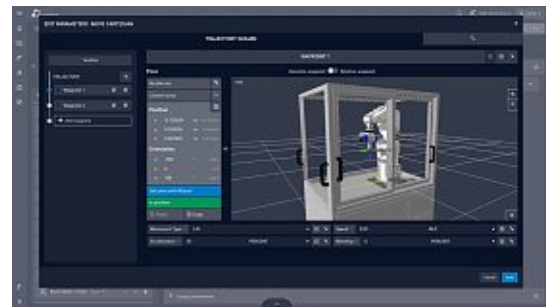


Figure 3.40: 3D Simulation with Drag&Bot [165]

3.3.9 Robot Operating System (ROS)

The Robot Operation System (ROS) [166] is an open-source framework, that provides hardware as well as software functionalities to create applications for robots. Originally

initiated by the Stanford University, an extension called ROS-Industrial [167] was subsequently developed to extend ROS with industry relevant hardware and applications. ROS and ROS-Industrial provide among others drivers for robots, code for various use cases and a huge online community. Furthermore the programming is independent from the programming language, which means that the functionalities can be implemented with different programming languages like C++, Python or Java. The big disadvantage (see Figure 3.41) is that the programming is text-based on the one hand and on the other hand the implementation requires a lot of knowledge. Therefore, this method is reserved for experts and not suitable for easy and intuitive programming of cobots.



Figure 3.41: Programming with ROS [168]

3.3.10 Conclusion

Summarized, the following three points can be derived from this brief overview of cobot programming solutions available on the market:

- The majority of the applications are manufacturer specific and use a teach pendant for programming.
- In all the solutions shown, the focus is on programming the robot. There are no, or only very limited, options for mapping human tasks (e.g. Wait (Universal Robots), User Input (Panda)).
- Since the various applications do not map the human tasks, the worker has no additional support during the execution.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Implementation

For the practical implementation of the prototype, User-centred Design (UCD) was used as methodology, as already described in chapter 1.3. This approach was chosen, because so the user can be integrated into the whole process and therefore a good usability can be achieved. There are various methods in the literature that implement the basis concept of UCD.

In the course of the work, the Human-centred Design Approach for Interactive Systems of the International Organization for Standardization, described in the ISO standard 9241-210 [14], was applied. In this approach, as it is usual with UCD, several iterations are run through. There were two iterations for the implementation of the BPMN prototype in the context of the thesis.

An iteration consists of 4 parts, whereby not all parts have to be run through in each iteration.

The 4 stages of an iteration of the Human-centred Design Approach [14] are:

- Understand and specify context
- Specify the user requirements
- Produce solution
- Evaluation

4.1 Iteration 1

4.1.1 Understand and specify context

In this part a systematic literature review based on the Guidelines by Kitchenham [15] was conducted to gain knowledge about the field and to identify the state of the art. A detailed description can be found in Chapter 3.

4.1.2 Specify the user requirements

In order to understand the mental model and the requirements of the users, a focus group of experts and users from the field of robotics was formed.

The great advantage of focus group interviews is that on the one hand the opinions of the individual participants can be collected and on the other hand new ideas can emerge in the course of the conversation [26, p.12]. It is important that the group size is not too large but also not too small. In the literature, a group size of 6-12 people is often recommended [26, p.12], [25, p. 10].

Formation of the focus group

So the first task at this point was to find 6-12 members for the focus group meetings. The main requirement here was experience in the use and/or programming of collaborative robots. Subsequently, an invitation (see Appendix A) was sent by e-mail to potential participants.

As potential participants the following three groups of stakeholders could be identified:

- Workers, who programme and use cobots in the industry.
- Researchers, who work in the field of Human-robot interaction (HRI) and provide the necessary knowledge for the development of new systems.
- Interested people, who use Makerspaces or Fablabs to gain access to collaborative robots.

Participants from all three groups were recruited for the focus group meetings.

First Focus Group Meeting

The aim of the first (online) meeting was to identify problems in programming collaborative robots and to work out requirements for an intuitive prototype for programming/modelling collaborative human-robot processes, which can also be used as worker assistance systems. The questions to be discussed during the meeting were provided to the participants in advance by email. Seven people from the three stakeholder groups mentioned above took part.

The following questions were discussed in this session:

- **Q1: What are current problems with programming collaborative robots?**
 - What works well/badly?
 - What is currently not possible?
 - What could/should be improved?
- **Q2: Keyword Human-machine Collaboration: How are human work steps currently considered/represented in programming?**
 - Are they taken into account?
- **Q3: Keyword Worker Assistance System Integration: How can the worker be better supported in the execution of his/her tasks?**
- **Q4: Keyword Intuitive Programming: What are your expectations/requirements for an intuitive user interface for programming (collaborative) robots?**

The meeting was recorded and transcribed. Below is a summary of the results for each question.

Q1: What are current problems with programming collaborative robots?

- **Inaccuracy of the robot:** This problem was confirmed by several participants. It can happen that a task that has been programmed, can no longer be executed correctly, due to minimal changes (e.g. minimal movement of the table or of components). This leads to errors during new runs, e.g. the robot cannot place the component in the workstation as programmed.
- **Nontransparent presentation of information:** A major problem is the lack of transparency of information. For example, the exact coordinates are not displayed in the interface of the cobot Panda, which makes it almost impossible to adjust faulty processes based on the above-mentioned error (inaccuracy of the robot). The consequence of this is that entire parts have to be newly programmed in the case of errors due to inaccuracy.
- **Lack of versioning / lack of reversible actions:** Another problem is the lack of an undo function. A versioning system similar to that used in software development would be desirable.

The following two points were highlighted as particularly good:

- **Force monitoring / safety:** The participants particularly praised the force monitoring, which makes working with the robot really good and safe.
- **Steep learning curve:** It was also pointed out that the learning curve for people without knowledge is very steep with the existing user interfaces. Within a very short time, people without experience can operate, control and programme cobots (at least the basic elements).

Q2: Keyword Human-machine Collaboration: How are human work steps currently considered/represented in programming?

- **Human work steps can only be mapped to a very limited extent:** Current programming environments of cobots (e.g. from Universal Robots or Franka Emika) take the work steps of humans only to a very limited extent into account. This is only possible with the following two methods:
 - **Timer:** By using a so-called timer function, the execution of the robot can be paused for a certain time, e.g. 40 seconds. This functionality can be used to indirectly integrate human work steps. The big problem is that the robot resumes work after this time, regardless of whether the worker has finished his step or not. This method is therefore only suitable for mapping human tasks to a very limited extent.
 - **Interaction:** With some cobots, such as the Panda, it is possible to define during programming that the robot only resumes the work when the user touches the robot arm. However, participants noted that this functionality does not really work reliably and that the robot does not always react immediately.
- **Cobot interface as a central element:** A major problem of user interfaces of cobots is that these user interfaces are always seen as the central element. There is no other layer that enables the modelling of entire processes and the integration of other elements such as sensors, other robots, human tasks, etc.

Q3: Keyword Worker Assistance System Integration: How can the worker be better supported in the execution of his tasks?

Currently, this is realised through the use of other systems (assistance systems). With the help of these, e.g. step-by-step instructions can be implemented. For a better support/integration of the worker, a different/additional representation would be needed. For example, a process modelling language like BPMN, with which the entire process could be mapped. This would make the robot "only" a part of the whole process and it could become more of a tool than the central element (see Q2). This representation of the

whole process and its execution would/could better support the worker in the execution. Furthermore, the participants affirmed that the possibility of storing and displaying additional information (instructions) at the process level (for human tasks) seems to be useful. However, care should be taken not to present this additional information (instructions) in an intrusive manner. Otherwise, the human being could get the feeling that the machine is telling him/her how to do the work. The human controls the robot and the process and not vice versa.

Q4: Keyword Intuitive Programming: What are your expectations/requirements for an intuitive user interface for programming (collaborative) robots?

- **Consider design guidelines:** First of all, certain design guidelines should be taken into account, e.g. consistency, so that the buttons are always in the same place, etc.
- **Different level/layer architecture:** A multi-layer architecture would make sense. This was affirmed by the majority of the participants. For example, a layer for programming using ROS, a layer for the existing interfaces of the manufacturers and a layer for process representation. In this way, programming can be separated.
- **Based on the layer architecture, further extensions would be useful and possible:** Based on this process representation, further features could be implemented. For example, it would be useful to see transparently over the entire process where problems (e.g. due to excessive speed) could occur. At this process level, it would also make sense to be able to define the speed of the robot for the entire process.

Based on the SLR (see Chapter 3) and the results of the focus group (especially the results from questions 2-4), the goal was to create a process modelling layer that extends the existing programming environment of cobots and can additionally be used as a worker assistance system.

The GUI should provide the following functions:

- Req 1: Creation of collaborative BPMN processes
- Req 2: Definition of additional parameters (name of the robot tasks, which should be executed)
- Req 3: Storage of additional information (work instructions as image and text)
- Req 4: Execution of the created processes (start/stop)

- Req 5: Representation of the current progress
- Req 6: Representation of the additional information (work instructions) during the execution
- Req 7: Detailed representation of robot tasks
- Req 8: User interaction (confirmation that the user has completed a specific task)

4.1.3 Produce solution

In this step, a first prototype was developed. It was implemented based on the process modelling language BPMN (see Chapter 2.3 and Chapter 5).

As can be seen in Figure 4.1, the BPMN prototype adds a third layer to the existing architecture of the cobot. With the help of this process modelling layer, created robot tasks of the manufacturer dependent user interface (layer 2) can be reused and entire processes can be mapped and executed. In the course of the work, the cobot Panda by Franka Emika [6] was used to implement the first prototype. The prototype currently only works with this cobot and would have to be extended accordingly for use with other collaborative robots.

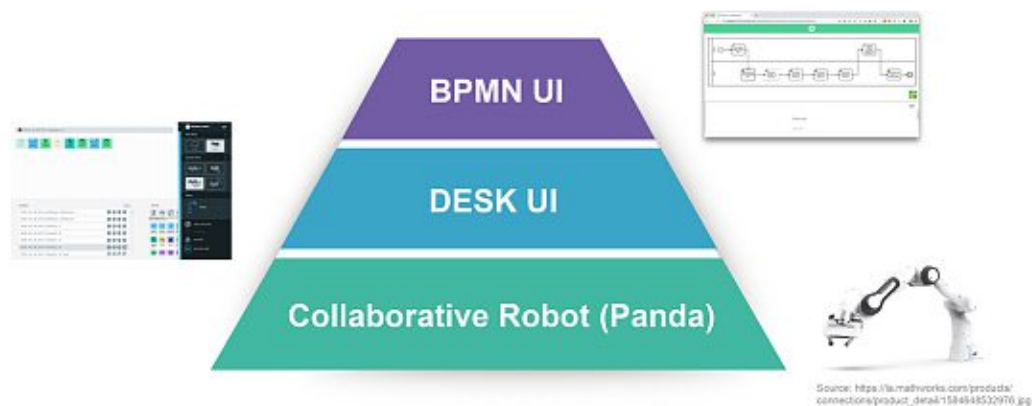


Figure 4.1: 3-layer architecture (own Figure)

Based on the requirements (see 4.1.2), this first prototype fulfilled the following six criteria (highlighted in green):

- **Req 1: Creation of collaborative BPMN processes**
- **Req 2: Definition of additional parameters (name of the robot tasks, which should be executed)**

- Req 3: Storage of additional information (work instructions as image and text)
- **Req 4: Execution of the created processes (start/stop)**
- **Req 5: Representation of the current progress**
- Req 6: Representation of the additional information (work instructions) during the execution
- **Req 7: Detailed representation of robot tasks**
- **Req 8: User interaction (confirmation that the user has completed a specific task)**

Figure 4.2 shows the web-based graphical user interface of the BPMN prototype.

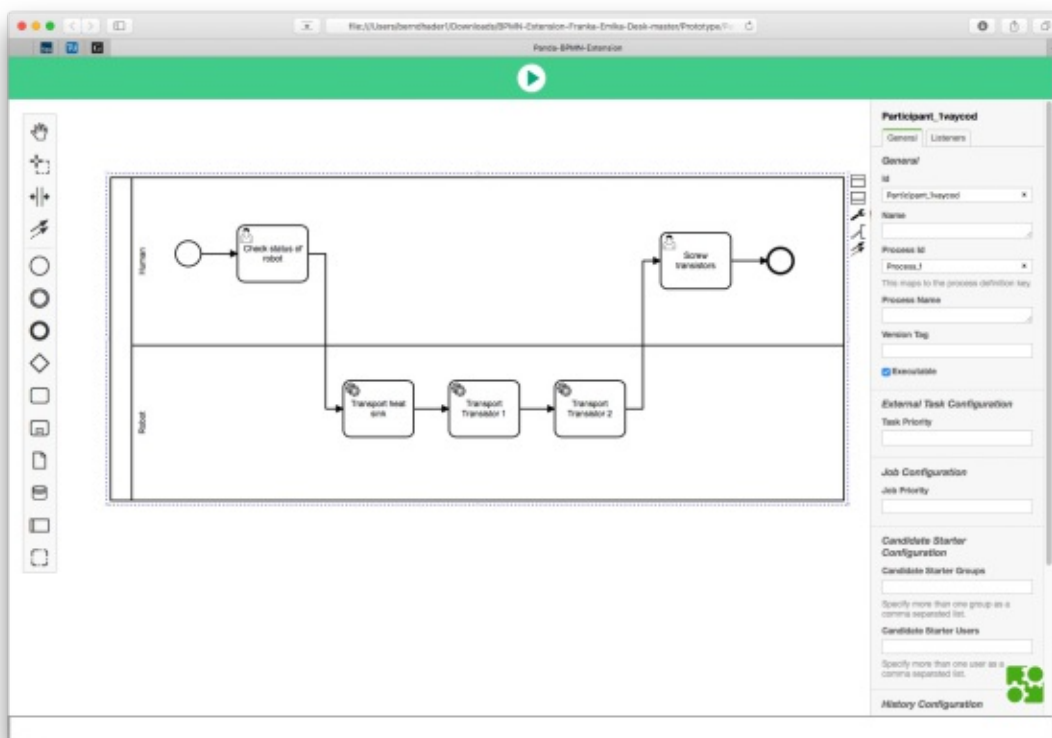


Figure 4.2: First draft of the GUI (own Figure)

The user interface essentially consists of only 4 areas:

- the workspace in the middle
- the toolbar on the left side
- a menu on the right side: Various parameters such as the name of the elements can be edited here.
- the green bar on top: By clicking on the Play button the created programme is executed.

As can be seen in Figure 4.2, a collaborative human-robot process has already been modelled, using a pool and two swimlanes, one lane for the human and one for the robot (Req. 1).

To model a human task, a so-called user task is used, which is represented by the person symbol. Finally, a name can/should be assigned and the user task is complete (see Figure 4.3).

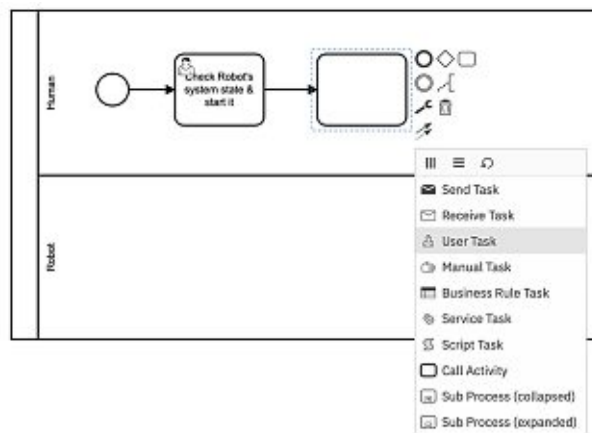


Figure 4.3: Creation of human tasks (own Figure)

To model a task for the robot, a so-called service task is used, represented by the gear wheel symbol. Furthermore, a name can/should be assigned (see Figure 4.4).

Finally, since it is a robot task, it must be defined what task the robot should execute. As already mentioned, the programmes of the Desk User Interface are used for this (created with the layer 2 user interface). For this assignment, select in the menu on the right (General tab) the entry Implementation "external" and enter "pandaTaskExecution" as the topic (see Figure 4.5). Furthermore, create in the tab "Input/Output" an input variable with the name "name" and the value "Name of the panda programme" must be

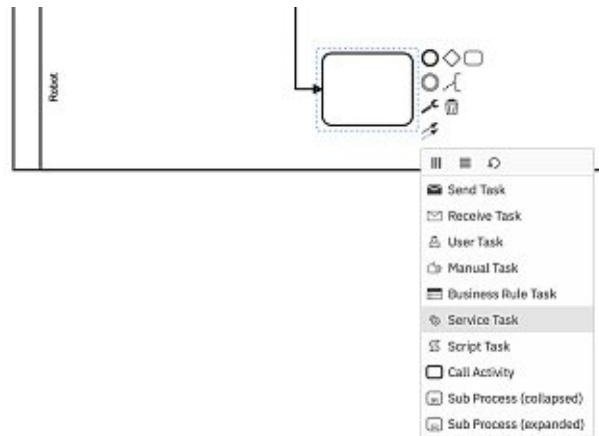


Figure 4.4: Creation of robot (service) tasks (own Figure)

assigned. In the example (see Figure 4.5), the task "transportTransistor1" was assigned (Req. 2).

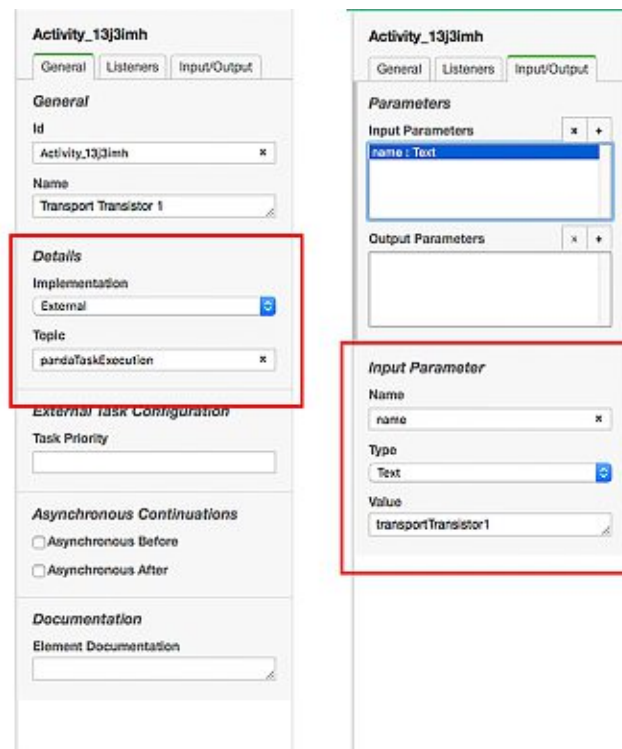


Figure 4.5: Creation of robot (service) tasks (own Figure)

4. IMPLEMENTATION

The created collaborative human-robot process can be executed by clicking on the play button (Req. 4). The menus disappear and the green animation shows the current status (see Figure 4.6 , Req. 5). If it is a user task, the user can confirm completion by clicking on the user task (Req. 8). If it is a robot task, the robot interface (DESK UI) is also displayed in the lower area of the GUI (see Figure 4.6, Req. 7). This provides the user with information about what action the robot is currently performing in detail.

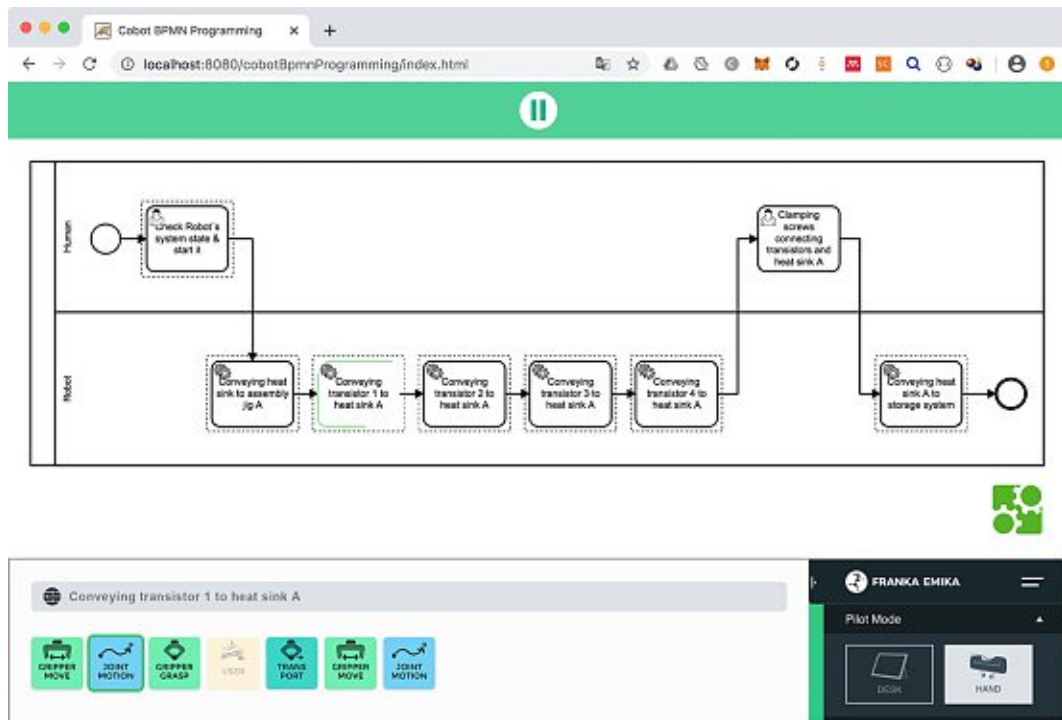


Figure 4.6: Execution of the collaborative process (own Figure)

4.1.4 Evaluation

This stage was carried out together with the phase "Specify the user requirements" of the second iteration and is therefore described in detail in Chapter 4.2.2.

4.2 Iteration 2

4.2.1 Understand and specify context

This stage was skipped in this iteration, because this step was already covered in detail in Iteration 1 (see Chapter 4.1.1).

4.2.2 Specify the user requirements

As already mentioned in Chapter 4.1.4, this stage was carried out together with the "Evaluation" phase of the first iteration. The first draft of the prototype was presented and evaluated in the course of the second focus group meeting. Furthermore, extension-/improvements were elaborated. This meeting was again held online and six people from the three identified stakeholder groups (see Chapter 4.1.2) participated. The meeting was recorded and transcribed.

The following points were elaborated as possible extensions/improvements:

- **Simplification of the user interface:** All menus and elements in the toolbar and in the menu on the right which are not needed should be removed. Otherwise, the user interface seems very complex and users could be overwhelmed very quickly.
- **Simplification of the robot task selection:** The selection of robot commands is very complicated in the first prototype and should be simplified as far as possible, e.g. a single input or search field.
- **Parameterisation of tasks:** A possible extension would be the parameterisation of robot tasks. In this way, instead of reteaching in the Desk UI, e.g. a move task with start and end coordinates could be defined in the BPMN UI.
- **Predefined standard processes:** Based on the previous point, an extension could be the integration of entire standard processes such as a Pick&Place task, where only the individual parameters need to be adjusted. This would allow inexperienced users to create complex processes very quickly.
- **Integration of a stopwatch as well as the display of reference times:** This would enable the employee to assess whether he/she is too fast or too slow in carrying out his/her tasks. Furthermore, it would be useful to have an estimated time until completion for the robot tasks, so that the worker knows when he/she has to take over the work.

4.2.3 Produce solution

Based on the results of the focus group meeting and the open requirements (see Chapter 4.1.3, Req. 3,6), the prototype was adapted as follows.

The worker assistance system (requirements 3,6) was implemented:

- **Req 3: Storage of additional information (work instructions as image and text)**
- **Req 6: Representation of the additional information (work instructions) during the execution**

As can be seen in Figure 4.7, a second menu (on the right) has been added, which is only available for human tasks. Here the user can enter additional information (text and/or images). During the execution of the collaborative process, the stored instructions, if available, appear in the lower area (see Figure 4.8).

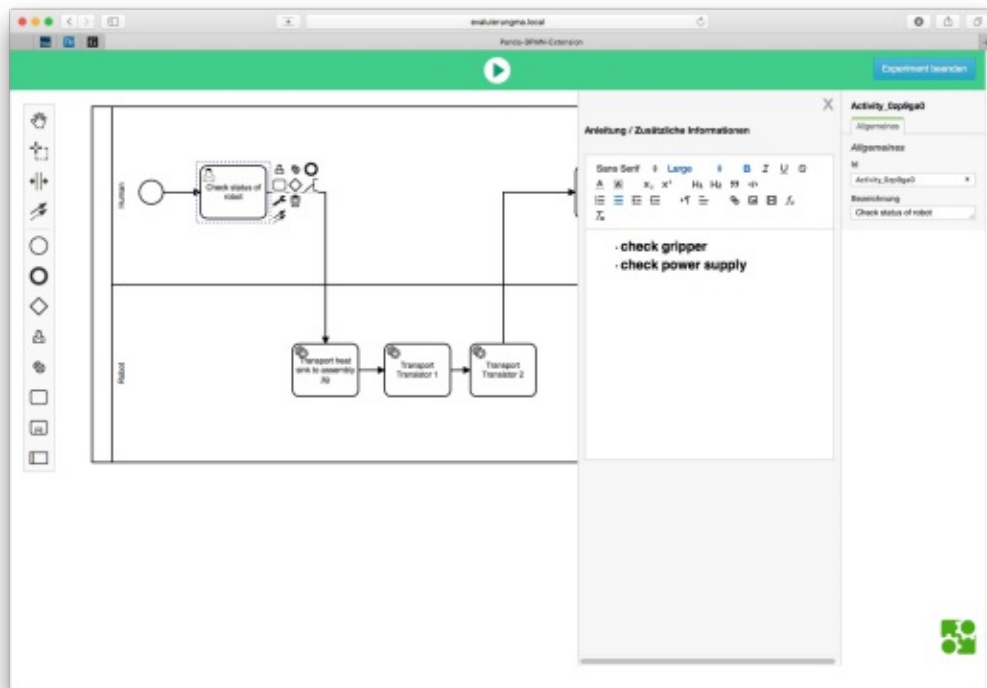


Figure 4.7: Storage of additional information (work instructions) (own Figure)

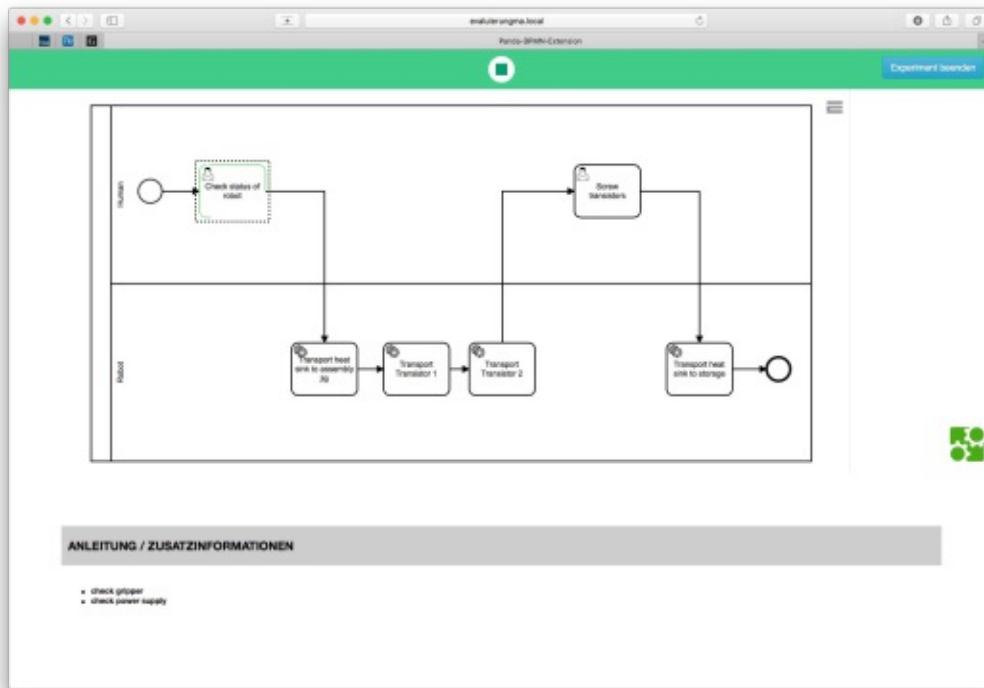


Figure 4.8: Representation of the work instructions during execution (own Figure)

The selection of the robot task has been simplified considerably so that the user only has to select the correct programme in the menu on the right using the drop-down menu (see Figure 4.9).

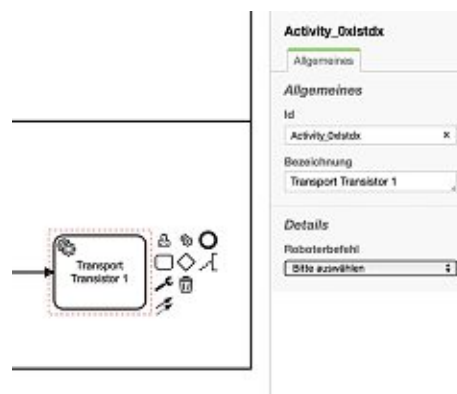


Figure 4.9: Simplification of the robot task selection

4. IMPLEMENTATION

Furthermore, the toolbar and the menu (directly next to the elements) as well as the menu on the right have been considerably simplified (see Figure 4.10), so that only the necessary elements are available. In the menu on the right, the user can only define two things: the name of the element and, in the case of a robot (service) task, the robot command to be executed. As you can see in the toolbar on the left, user and service tasks are now also available here.

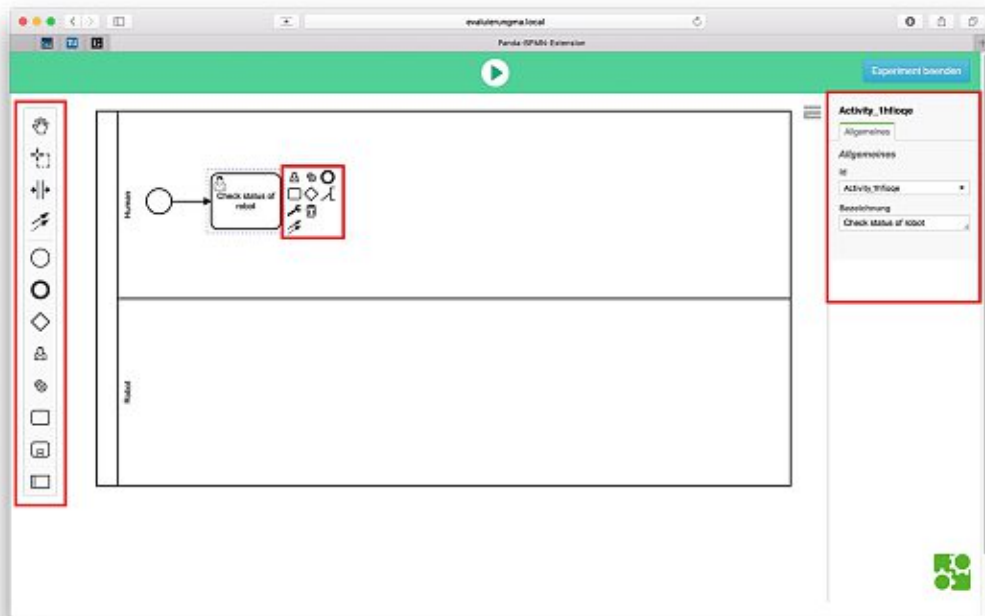


Figure 4.10: Simplification of the user interface (own Figure)

The points "Parameterisation of tasks", "Predefined standard processes" and "Integration of a stopwatch as well as the display of reference times" could not be implemented at this point due to the limited time frame of the thesis.

4.2.4 Evaluation

The final evaluation was conducted by means of an online user study. A detailed description of this phase can be found in Chapter 6.

Software Architecture

In this chapter follows an explanation of how the prototype was implemented.

As shown in Figure 5.1, the architecture consists of the following four main components:

- **Web-based User Interface**
- **Camunda BPMN Engine**
- **External Node JS Task Client**
- **Franka Emika Panda Cobot**

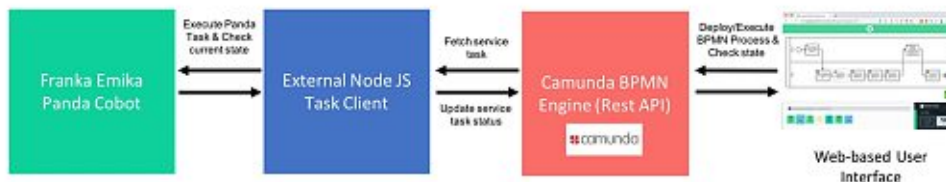


Figure 5.1: Software Architecture (own Figure)

We will now look at each of this components in detail, starting with the central core of the system - the BPMN Engine.

5.1 (Camunda) BPMN Engine

As already described in Chapter 2.3.4 the application was realized with the help of the process modelling language BPMN 2.0. An essential advantage of BPMN 2.0 is the

possibility to execute these models without transformation into another language. For this purpose, a so-called BPMN Engine is necessary. Various solutions, both free and paid, already exist on the market, such as Signavio [99] and Camunda [100]. Which engine is ultimately used, is irrelevant as long as it is BPMN 2.0 compliant. Finally, Camunda was used in the course of the work, since it is open source, free and very well documented. There is also a paid version (Enterprise Platform), but for the use as a pure BPMN engine in the background of the application the Community Platform is completely sufficient.

The main task of the Business Process Model and Notation Engine is to execute and control the created workflows [84, p.136].

For explanation:

At this point it is unimportant how the BPMN workflow is created. A detailed explanation how this was implemented in the course of the work follows later. Assume that we have created the following collaborative human-robot BPMN process (see Figure 5.2). This is a very simple assembly process. After the process has been started, the human first checks the robot, then the robot transports two transistors to the workstation. Finally, the worker screws the transistors.

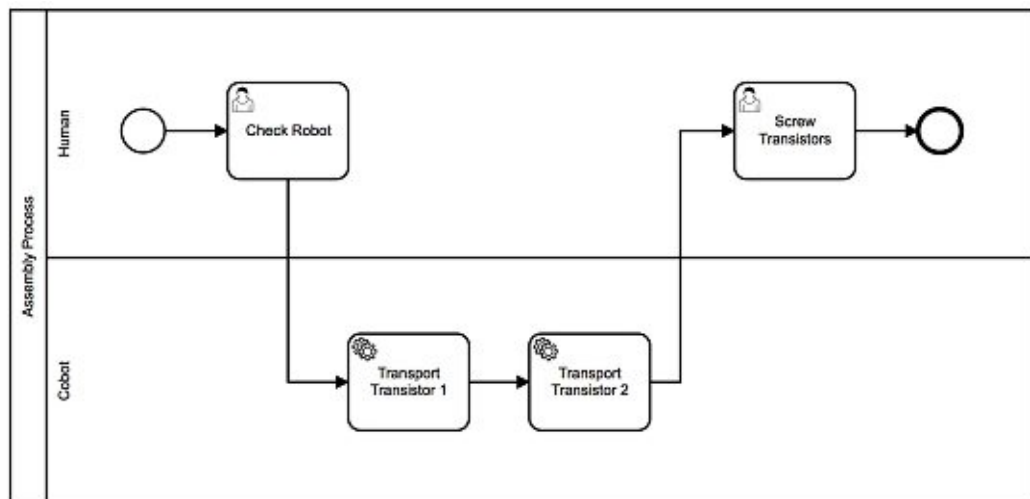


Figure 5.2: Example of a collaborative assembly process modeled with BPMN (own Figure)

The task of the BPMN Engine is to control and monitor the execution of the process. Furthermore, other systems must be able to communicate with the BPMN Engine so that these systems (for example the robot) know if it has to execute a task. As soon as the robot has executed the task, it has to tell the BPMN Engine that the task is finished.

Figure 5.3 shows a snapshot of the same process within the BPMN Engine. The blue circle at the task "Check Robot" indicates that this is the current task to be executed. As

soon as this task is finished, the blue circle jumps to the next task "Transport Transistor 1", as you can see in Figure 5.4.

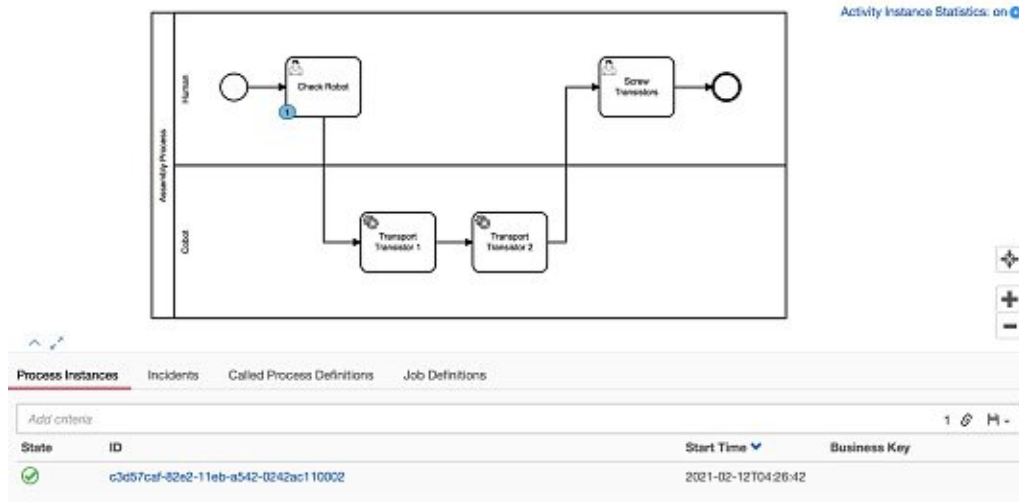


Figure 5.3: Running process in the BPMN engine (own Figure)

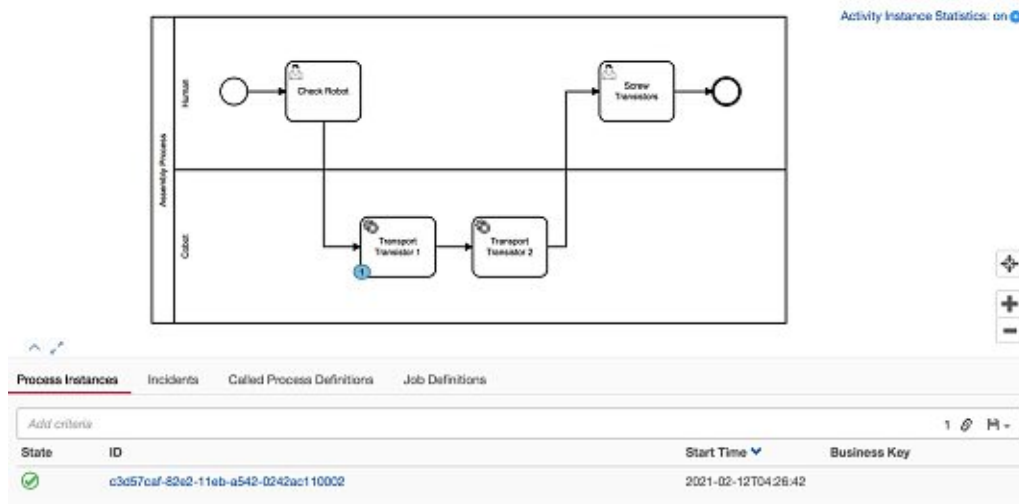


Figure 5.4: Running process in the BPMN engine (own Figure)

5.1.1 Interaction with the BPMN Process Engine

As already mentioned, the BPMN Engine is used in the background to control and execute the process. For interaction with the process engine an interface, a so-called API, is needed.

An Application Programming Interface, short API, is an interface of a software system, which enables the connection/communication of different systems or different components within the software [169].

This is necessary depending on the architecture, since the components are usually distributed. For example Google Maps can be used as IOS app, Android app or via browser. For various search queries as well as the route planning, the app or browser must communicate with the Google server, this is done by using an API.

Furthermore, it is often necessary that different software systems communicate with each other. As an example one can look at a booking platform like <https://www.booking.com>. If someone books a hotel using this site, the platform has to communicate with the IT system of the hotel during the booking process, so that the same room is not booked twice at the same time. Afterwards payment by credit card requires communication with a banking application to check if the credit card is valid and covered. [169]

Coming back to the BPMN Engine: The API enables the communication with other systems. This is necessary to transfer new process models, query open tasks and change their state.

The various suppliers like Signavio [99] or Camunda [100] offer different APIs to communicate with their process engines.

Camunda currently provides the following two Application Programming Interfaces:

- Java API [170]
- REST API [171]

Since it is a web-based prototype, which was programmed using Javascript, NodeJS, HTML and not Java, the REST API was used.

What is a REST(ful) API?

The Representational State Transfer (REST) is an architectural style in software development, based on six constraints (client-server, stateless, cache, uniform interface, layered system, code-on-demand), that defines the machine to machine communication of distributed systems [172, p.76]. A RESTful API, often just called REST API, is in further consequence an API which implements this architectural style.

For a better understanding we will look in the following section how such a RESTful API looks like. A variety of services that you use every day, such as Twitter, Facebook,

Google and co, use RESTful APIs. But first it must be clarified how communication in the World Wide Web is done. For this the http/https protocol is used.

The Hypertext Transfer Protocol, short HTTP, is the way how all browsers and web servers communicate with each other. For example, when you visit e.g. <https://www.iLoveCobots.at/hello>, your Browser requests the page "hello" via HTTP from iLoveCobots. Since not only data should be retrieved but also changed or deleted, the HTTP protocol has several methods that can be used for a request (e.g. GET, POST, PUT, DELETE) [173]:

- To retrieve data from the server the method GET is used, as in the example above (<https://www.iLoveCobots.at/hello>).
- A new resource (e.g. a new user account) can be created through the POST method.
- With PUT a resource (e.g. user data) can be changed.
- DELETE, as the name suggests, is used to delete a resource.

Example RESTful API

Assuming that user data should be read, created, modified and deleted on the website <https://www.iLoveCobots.at>, the REST API could be defined as follows:

- **GET /users**
The HTTP GET request (<https://www.iLoveCobots.at/users>) returns all users who are already registered.
- **POST /users**
The same request as above but with the method POST instead of GET, creates a new resource (in this specific case, a new user). Of course, in the API specification it must also be defined exactly which parameters must be passed in which form.
- **PUT /users/id**
The method PUT in combination with an ID (e.g. <https://www.iLoveCobots.at/users/7>) is used to change the data of one specific user (in this example the user with the ID 7).
- **DELETE /users/id**
The same request as before, but with the method DELETE instead of PUT deletes the user.

An example of a detailed API description (concrete how to delete a deployment in Camunda) can be found here [174]. It specifies in detail how the resource can be accessed, which parameters can/should be passed and which server responses are possible. How the system finally executes the task in the background is not relevant for the user of the API.

5.1.2 Camunda REST API [171]

Under [171] a complete description of the Camunda REST API for using the BPMN engine can be found.

In the following it is now explained in detail how

- a process can be deployed,
- a process can be started,
- the currently open tasks can be retrieved and
- the status of a specific task can be changed.

Deploy a Process [175]

URL:	/deployment/create
HTTP Method:	POST

Parameters

Name	Content Type	Description
deployment-name	text/plain	The name of the deployment.
deployment-source	text/plain	The source of the deployment.
deploy-changed-only	text/plain	A flag, that checks if the deployment already exists. If set to true, only changes are deployed.
*	application/octet-stream	The binary deployment data (bpmn diagram).

Result

The response is a JSON object with a variety of properties. For the prototype, only the "id" of the deployed process definition is of interest. It is important to note that a deployment has a process definition only if the BPMN diagram is executable. With this process definition id the process can be started and open tasks can be queried.

Example

For a better understanding an example of how a new deployment can be created using the presented REST API follows. In the example (see Figure 5.5) the API client Postman [176] was used.

Figure 5.5 shows the POST request /deployment/create in Postman. In this example, the Camunda REST API can be accessed via the URL "http://localhost:8080/engine-rest".

The parameters described above (deployment-name, deployment-source, deploy-changed-only and myProcess) were defined in the body. The name of the deployment (myProcess) can be chosen freely. As you can see in Figure 5.5, the BPMN diagram was attached as a file (assemblyProcess1.bpmn). By clicking on the button "Send" the request is finally sent. Figure 5.6 shows the response of the request. The status 200 OK (top right) indicates that the request was successfully executed. Furthermore the answer contains the JSON object, which provides among other parameters also the id of the "deployedProcessDefinitions".

KEY	VALUE	CONTENT TYPE
<input checked="" type="checkbox"/> deployment-name	AssemblyTask1	text/plain
<input checked="" type="checkbox"/> deployment-source	Panda BPMN Extension	text/plain
<input checked="" type="checkbox"/> deploy-changed-only	Text true	text/plain
<input checked="" type="checkbox"/> myProcess	assemblyProcess1.bpmn X	application/octet-stream
Key	Value	Auto

Figure 5.5: REST API Request "Process Deployment" (own Figure)

```

1  {
2    "links": [
3      {
4        "method": "GET",
5        "href": "http://localhost:8080/engine-rest/deployment/13f3be84-2814-11eb-8214-7ecc54920a56",
6        "rel": "self"
7      }
8    ],
9    "id": "13f3be84-2814-11eb-8214-7ecc54920a56",
10   "name": "AssemblyTask1",
11   "source": "Panda BPMN Extension",
12   "deploymentTime": "2020-11-16T15:00:26.413+0100",
13   "tenantId": null,
14   "deployedProcessDefinitions": {
15     "Process_lxr30lk:1:13f963d6-2814-11eb-8214-7ecc54920a56": {
16       "id": "Process_lxr30lk:1:13f963d6-2814-11eb-8214-7ecc54920a56",
17       "key": "Process_lxr30lk",
18       "category": "http://bpmn.io/schema/bpmn",
19       "description": null,
20       "name": null,
21       "version": 1,
22       "resource": "assemblyProcess1.bpmn",
23       "deploymentId": "13f3be84-2814-11eb-8214-7ecc54920a56",
24       "diagram": null,
25       "suspended": false,
26       "tenantId": null,
27       "versionTag": null,
28       "historyTimeToLive": null,
29       "startableInTasklist": true
30     }
31   },
32   "deployedCaseDefinitions": null,
33   "deployedDecisionDefinitions": null,
34   "deployedDecisionRequirementsDefinitions": null
35 }

```

Figure 5.6: REST API Response "Process Deployment" (own Figure)

Figure 5.7 shows the BPMN process, that was successfully deployed in the Camunda BPMN Engine, using the REST API.

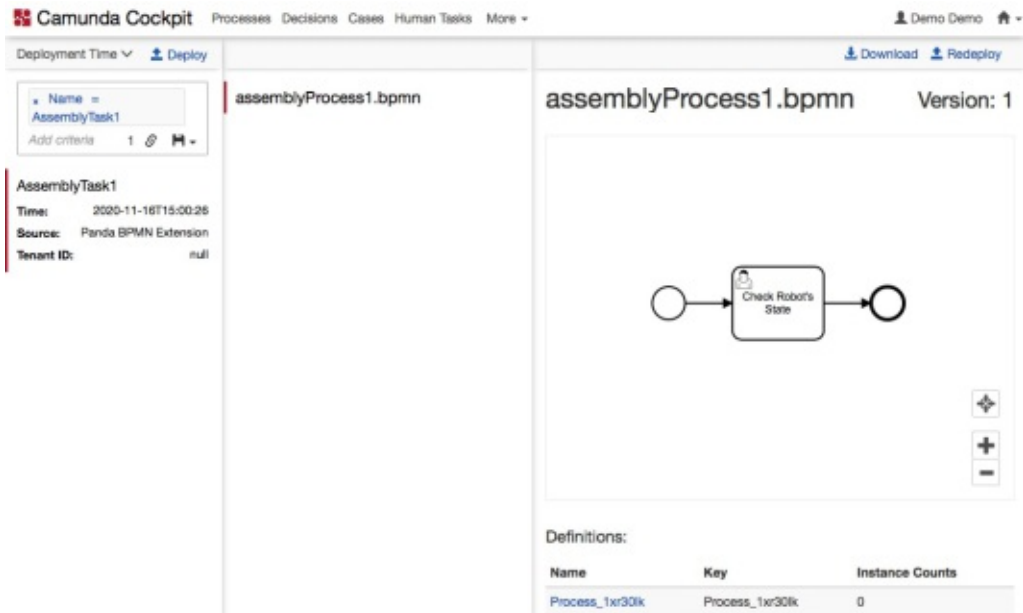


Figure 5.7: Overview Camunda Deployments (own Figure)

Start a Process 177

To start a new process instance of a deployment, the following REST endpoint can be used (there are several methods, in the following the method with the process definition id is described).

URL: `/process-definition/{id}/start`
HTTP Method: POST

Parameters

Name	Description
id	The id of the process definition.

Result

The response is a JSON object with a variety of properties (see Figure 5.9).

Example

Figures 5.8 and 5.9 show how the previously deployed process can be started using the REST API and Postman. It is important that the Content-Type is defined correctly (application/json) (see Figure 5.8), otherwise the request will return an error.

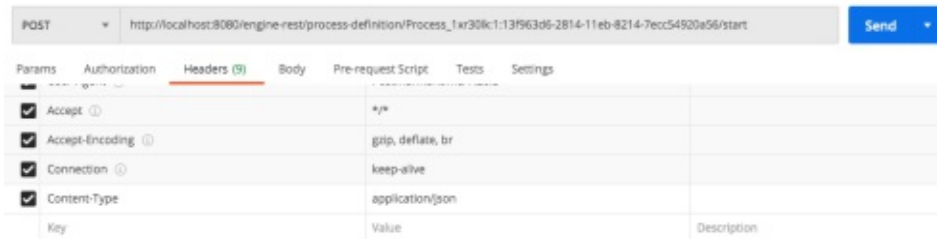


Figure 5.8: REST API Request "Start Process Instance" (own Figure)



Figure 5.9: REST API Response "Start Process Instance" (own Figure)

Figure 5.10 shows the BPMN process, that was successfully started in the Camunda BPMN Engine, using the REST API.



Figure 5.10: Successfully started BPMN process (own Figure)

Retrieve Open Tasks [\[178\]](#), [\[179\]](#)

The prototype uses service tasks for tasks, that are executed by the robot and user tasks for tasks, that are executed by humans. There is no way to get both (open user tasks as well as service tasks) with only one REST request. To query the currently open service/user tasks, it is necessary to use the following two endpoints.

Get Service/External Tasks [\[179\]](#)

URL: /external-task
HTTP Method: GET

Parameters

Name	Description
processDefinitionId	The id of the process definition.

Result

The response is an array of JSON objects.

Example

As it can be seen in Figure 5.12, the JSON array is empty. This means that there is currently no service/external task waiting to be executed. As our process model consists of only one user task (see Figure 5.10) this result was to be expected.

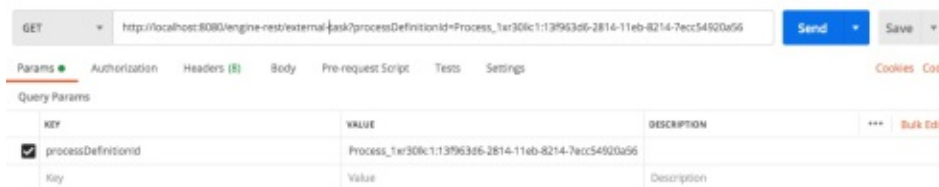


Figure 5.11: REST API Request "Get list of open service tasks" (own Figure)



Figure 5.12: REST API Response "Get list of open service tasks" (own Figure)

Get (User) Tasks [\[178\]](#)

With the following endpoint a list of all active (open) tasks (except external tasks) of a process instance can be retrieved.

URL:	/task
HTTP Method:	GET

Parameters

Name	Description
processDefinitionId	The id of the process definition.

Result

The response is an array of JSON objects.

Example

As it can be seen in Figure 5.14, there is currently exactly one user task waiting to be executed. This is the task "Check Robot's State" (see Figure 5.10).

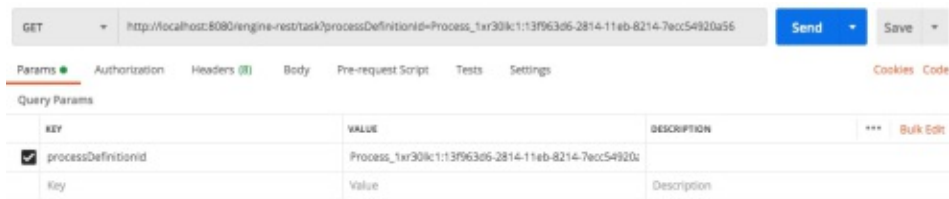


Figure 5.13: REST API Request "Get list of open (user) tasks" (own Figure)

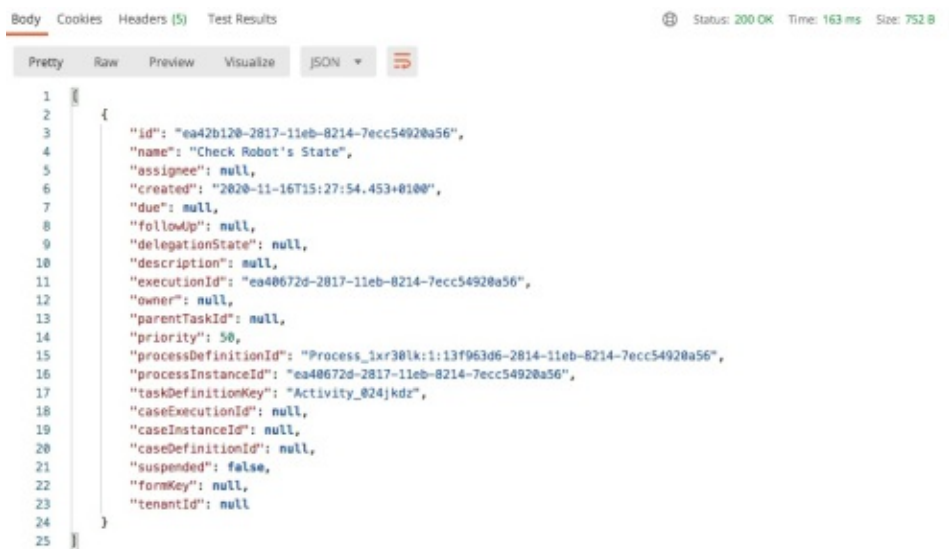


Figure 5.14: REST API Response "Get list of open (user) tasks" (own Figure)

Update the Status of a Task 180

To set the status of an open user task to completed, the endpoint looks like following. For external tasks (service tasks) this works exactly the same, only the endpoint is different (/external-task/id/complete).

URL: /task/{id}/complete
HTTP Method: POST

Parameters

Name	Description
id	The id of the task.

Result

The response has no content. If the HTTP status code 204 is returned, the request was successful (see Figure 5.16).

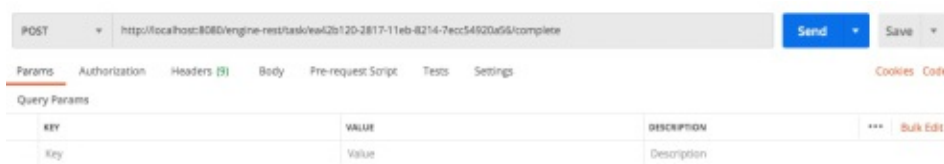


Figure 5.15: REST API Request "Set status of user task to completed" (own Figure)



Figure 5.16: REST API Response "Set status of user task to completed" (own Figure)

5.1.3 Conclusion

Summarized this means, that the (Camunda) BPMN Engine is the central core of the system and all systems (the robot as well as the user interface) have to communicate with it. In the case of Camunda there are two APIs for communication, whereby the prototype uses the REST API.

5.2 Franka Emika Panda Cobot

Franka Emika's Panda was used as collaborative robot in the course of the work. In contrast to other manufacturers, who mostly use teach pendants for programming, the Panda uses a web-based interface, whereby the programming is done directly in the browser.

The user interface (see Figure 5.17) basically consists of only three elements. On the bottom right are the so-called apps, which represent specific abilities of the cobot (like move, transport, grasp). Via drag and drop these apps can be combined to tasks in the upper area. An overview of all created tasks (programs) is displayed in the lower left area. Here the created programs can be edited and executed.

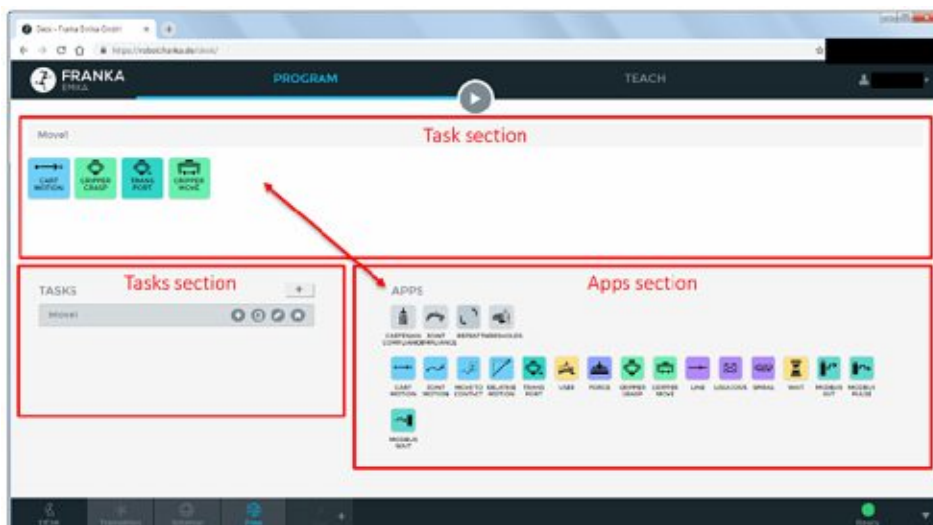


Figure 5.17: "Desk" User Interface of the Cobot Panda by Franka Emika [8]

The manufacturer offers at the moment only two options for the external control of the cobot [181]: `franka_ros`, which represents the ROS integration and `libfranka`, that provides a C++ library for low-level control of the robot.

A Java or RESTful API does not exist according to the official documentation.

However, based on existing projects [182] and an analysis of the Panda User Interface (Desk) it was found that a REST API exists. The lack of public documentation made it difficult to identify the required endpoints, but finally all necessary endpoints for controlling the robot could be identified.

The following section explains how to control the Panda (tested with system version 2.1.4) using the RESTful API.

5.2.1 Authentication

An authentication token is required to execute existing Panda tasks and query their status using the REST API. This token can be obtained by using the following endpoint and the desk login data.

URL:	<code>https://robot.franka.de/admin/api/login</code>
HTTP Method:	POST

Parameters

Name	Description
login	The username of the Desk credentials.
password	The encrypted Desk password.

Furthermore, the content-type must be set accordingly (Content-Type: application/json).

Encryption

As it can be seen in [182] and [183], the password is encrypted as follows:

- create password string: `password#username@franka`
- create hash of the string with sha256
- split hash into groups of two and convert hex values to integer
- create a string in the following form: `intValue1,intValue2,...,intValueX`
- convert string to binary data
- create a base64 string from the binary data

Result

The response is a string, which represents the authentication token. This token is only valid for a certain time, which is why the prototype always sends a login request before every action (e.g. start task).

5.2.2 Start a Task

As described in the Github repository [182], existing panda tasks can be executed using the following REST endpoint.

URL:	<code>https://robot.franka.de/desk/api/execution</code>
HTTP Method:	POST

Parameters

Name	Description
id	The name of the Panda task.

It is important to set the headers as follows:

Name	Value
Content-Type	application/x-www-form-urlencoded
Authorization	Value of authentication token (see section Authentication above)

Result

The response with an empty body and HTTP status code 200 indicates that the request was successful and the robot is executing the task.

5.2.3 Get Status of a Task

Whether a task is currently running or not can be queried with the following endpoint.

URL:	<code>https://robot.franka.de/desk/api/execution</code>
HTTP Method:	<code>GET</code>

Parameters

No parameters are needed, but it is important to add the authentication token in the header:

Name	Value
Authorization	Value of authentication token (see section Authentication above)

Result

The response is a JSON object (see Figure 5.18). The parameters "active" and "running" indicate that the task is being executed.

```
{
  "state": {
    "children": [
      {
        "children": [
          {
            "children": [],
            "active": false
          },
          {
            "children": [],
            "active": false
          },
          {
            "children": [],
            "active": true
          },
          {
            "children": [],
            "active": true
          }
        ],
        "result": null,
        "active": true,
        "id": "0_tele_1_aufnehmenleiterplatte",
        "exitPort": null,
        "error": null,
        "lastActivePath": {
          "indices": [0, 2],
          "id": "0_tele_1_aufnehmenleiterplatte",
          "running": true,
          "errorHandling": false,
          "tracking": true
        }
      }
    ]
  }
}
```

Figure 5.18: REST API Response "Start Panda Task"

5.3 External Node JS Task Client

As we will see in Chapter 5.4, all tasks performed by the robot are mapped using service tasks. This approach was chosen because BPMN by definition already specifies this.

"A Service Task is a Task that uses some sort of service, which could be a Web service or an automated application." [86, p.158]

Depending on the BPMN engine used, there are different ways to query service tasks to be executed. One possibility in Camunda is the RESTful API (see Chapter 5.1.2). However, Camunda offers another, more comfortable option, referred to as the external task pattern [184].

External Task Pattern

In the external task pattern so-called workers are defined, who are responsible for the execution of a specific task. This task could be, for example: the validation of address data. Now it is possible that there are several different external tasks like "Validate address", "Upload form", "Publish content on facebook" and so on. For each of these tasks, it is necessary to define a worker, who then executes exactly this type of task according to a concrete scheme. In order to assign service tasks to a worker, it is necessary to specify a so-called "Topic". In the following example, the service task is assigned to the worker who is in charge of the topic: "pandaTaskExecution".

```
<serviceTask id="Activity_0cqfbri"
  name="Transport Transistor 1"
  camunda:type="external"
  camunda:topic="pandaTaskExecution" />
```

In Camunda there are two possibilities how an external task client can be created, using Java or NodeJS [185]. In the course of the work the variant using NodeJS was chosen [186].

The basic structure of the Node JS task client, described in detail under [186], looks like this:

```
1 const { Client, logger } =
2   require("camunda-external-task-client-js");
3
4   // configure client
5   const config = { baseUrl: "http://localhost:8080/engine-rest",
6     use: logger };
7
8   // create a Client instance with custom configuration
9   const client = new Client(config);
```



```

10
11 // subscribe to the topic: 'pandaTaskExecution'
12 client.subscribe("pandaTaskExecution",
13 async function({ task, taskService }) {
14
15     // TODO: add logic to control the cobot Panda
16
17     // complete the task
18     await taskService.complete(task);
19
20 });

```

The focus here is on the subscription (see lines 12-20). The client receives all open external service tasks from the Camunda BPMN engine, which are assigned to the topic `pandaTaskExecution`. If the worker has finished his work, this is reported to the BPMN Engine via `taskService.complete(task)` (see line 18).

The business logic essentially comprises the following parts:

- **Login (request authentication token):** As already described in Chapter 5.2.1, an authentication token is required to start and control the execution of the cobot Panda via REST API.
- **Start a task:** Therefore, it is first necessary to obtain the name of the panda task, that should be executed. The parameter "name" is used for this.

```

<serviceTask id="Activity_0cqbri "
  name="Transport Transistor 1"
  camunda:type="external "
  camunda:topic="pandaTaskExecution " />

```

Then the RESTful API (see Chapter 5.2.2) is used to start the task (in the example the task "Transport Transistor 1" would be executed).

- **Check the status of the task execution:** Once the robot has been started, the next step is to determine whether the robot has finished the task or not. By using the REST API (see Chapter 5.2.3) it can be only determined if a task is running at the moment or not. However, it is not possible to obtain information when the robot will finish the respective task. For this purpose it was necessary to implement a state machine.

What is a Finite State Machine?

"It is a mathematical model of computation. It is an abstract machine that can be in one and only one state at a given time, has a special state

called the start state, and can change from one state to another via a transition activated because of some input. [187]

The Finite State Machine (FSM) is used to map the current status of the service task (robot task). A task received by the worker can have the following three states:

- **open:** This is the initial state.
- **running:** After the task has been started successfully via REST API (see Chapter 5.2.2), the status of the task is changed to "running".
- **completed:** As soon as the task has been completed by the robot, the status is changed to "completed".

In NodeJS there are different packages which provide state machines, concretely the following project [188] was used and the state machine was defined as follows:

```

1 var fsm = new StateMachine({
2   init: 'open',
3   transitions: [
4     { name: 'start', from: 'open', to: 'running' },
5     { name: 'end', from: 'running', to: 'completed' }
6   ]
7 });
```

For each task, which is executed by the robot, a new state machine object is created (see line 1). The task has initially the state 'open', as soon as the REST Request (Start a Task, see Chapter 5.2.2) has been executed successfully, the state changes from 'open' to 'running'.

Subsequently, the worker queries every 100ms via RESTful API (see Chapter 5.2.3) whether the collaborative robot Panda is still executing the task or has already finished it. As soon as the value "active: false" is transmitted, the robot has finished the task. Afterwards, the status is changed to 'completed' and the BPMN Engine is informed, that the worker has finished this task.

The code of the prototype can be found in the following github repository [183]. This repository also contains the External Node JS Task Client (see app.js).

The application can be started via command line running "node app.js". Thus the task client runs in the background and waits for external service tasks with the topic "pandaTaskExecution".

5.4 Web-based User Interface

The previous topics covered how BPMN processes can be executed and how the robot can be controlled by service tasks. The last component of the software architecture is the user interface, which should enable the creation of programs (BPMN models) and their execution. In addition, the prototype should provide an entire worker assistance system. Therefore, as already mentioned, it should be possible to store additional information (text and images) to provide additional assistance to the user during execution.

The graphical user interface was implemented as a web-based solution. This has the big advantage that no installation is necessary and the program can be used via browser. The open-source Github project bpmn.io [97] from Camunda was used as a basis.

5.4.1 What is bpmn.io?

The project bpmn.io [97] provides web-based tools (currently three), for modelling BPMN, DMN and CMMN diagrams. With the help of these three applications, users of Camunda can easily integrate the creation of BPMN, DMN and CMMN diagrams into their software. Since the tools are open source, they can be modified and extended as required.

bpmn-js [189]

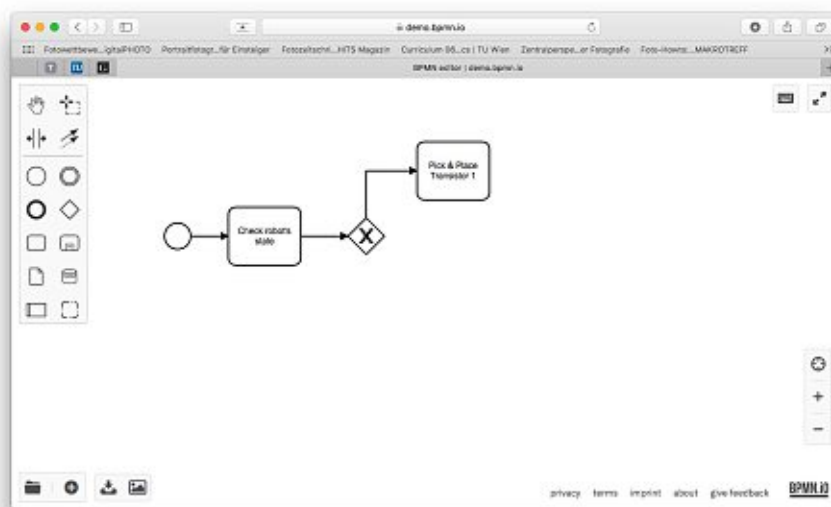


Figure 5.19: Web-based BPMN Modeler (bpmn-js) from bpmn.io (own Figure)

bpmn-js (see Figure 5.19) is one of the three projects of bpmn.io, which, as the name suggests, provides a toolkit for modelling BPMN diagrams. It is a Javascript project, written as node.js module. It can be integrated in node.js applications as well as in

classic web applications. For the latter one the node-js code must be converted into Javascript code by means of Browserify [190].

The project bpmn-js provides a variety of functions and extensions. First of all it was necessary to identify which functions are provided and which can be used for the creation of the worker assistance system.

The GUI of the digital worker assistance system should offer the user the following functions:

- Req 1: Creation of collaborative BPMN processes
- Req 2: Definition of additional parameters (name of the robot tasks, which should be executed)
- Req 3: Storage of additional information (work instructions as image and text)
- Req 4: Execution of the created processes (start/stop)
- Req 5: Representation of the current progress
- Req 6: Representation of the additional information (work instructions) during the execution
- Req 7: Detailed representation of robot tasks
- Req 8: User interaction (confirmation that the user has completed a specific task)

The following two requirements can be realised with bpmn-js:

- Req 1: Creation of collaborative BPMN processes
- Req 2: Definition of additional parameters (this can be achieved with the extension (bpmn-js-properties-panel [191])

On the following pages a brief explanation follows of how the individual requirements were implemented.

5.4.2 Req 1: Creation of collaborative BPMN processes

The web-based creation of BPMN processes using bpmn-js provides all features that are needed to fulfill Requirement 1. In this step it was only necessary to adjust the tool palette so that only those elements are available which are really needed. As an example the Task tool was removed and replaced by the two elements Service Task and User Task (see Figure 5.20 - on the left).

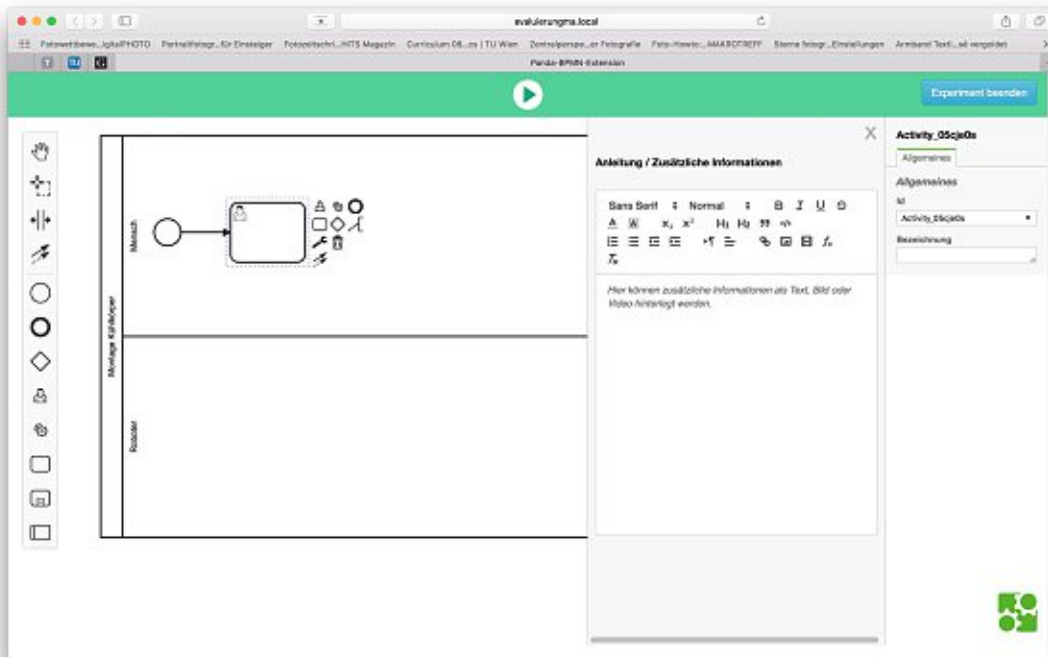


Figure 5.20: Final BPMN Prototype, create/update a user task (own Figure)

5.4.3 Req 2: Definition of additional parameters (name of the robot tasks, which should be executed)

The definition of additional parameters is not possible with the basic configuration of bpmn-js (see Figure 5.19). For this purpose the extension bpmn-js-properties-panel [191] was used. It is then possible to define additional variables via input/output parameters. This procedure is quite cumbersome, because it has to be repeated for each service/robot task. Therefore, the properties panel has been adapted so that the user can only change the following three parameters (see Figure 5.20 and 5.21 - right section):

- Id: The ID is automatically set when an item is created and can be edited in the properties panel.
- Name: The user defines a name, which is then also displayed directly in the BPMN process for each task.
- Robot Command (only for service tasks): The user must select one from a list of available robot tasks (see Figure 5.21). The list is retrieved directly from the robot via REST API.

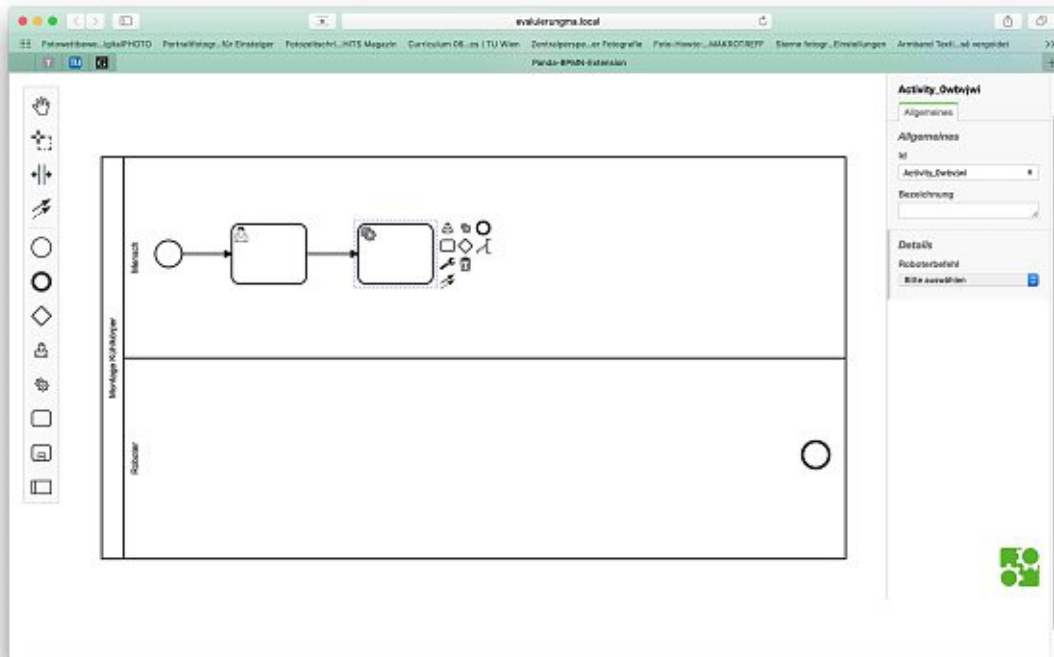


Figure 5.21: Final BPMN Prototype, edit service task (own Figure)

5.4.4 Req 3: Storage of additional information (work instructions as image and text)

So that additional work instructions can be stored for the respective user tasks, a further optional window was created using HTML, CSS and Javascript (see Figure 5.20 right section). The editor Quill [192] was used for this. Since the prototype does not use a database in the background, the inputs are stored in a Javascript array at runtime. To avoid this additional window being perceived as annoying, the user can close the window by clicking on "X". This window remains closed until the user opens it again by clicking on the symbol with the three lines.

5.4.5 Req 4: Execution of the created processes (start/stop)

A green bar (see Figure 5.20) has been added to the top, so that the created models can also be executed. By clicking on the "Play" button the created BPMN diagram is generated in a first step. Parameters like (isExecutable) as well as the robot commands are added to the BPMN process model, which is generated by bpmn-js. Afterwards the BPMN diagram is transferred via REST API to the process engine. This was also implemented by the use of HTML, CSS and Javascript.

5.4.6 Req 5: Representation of the current progress

To keep the focus on the process model during execution, all unnecessary areas are hidden, such as the toolbar and the properties panel (see Figure 5.22).

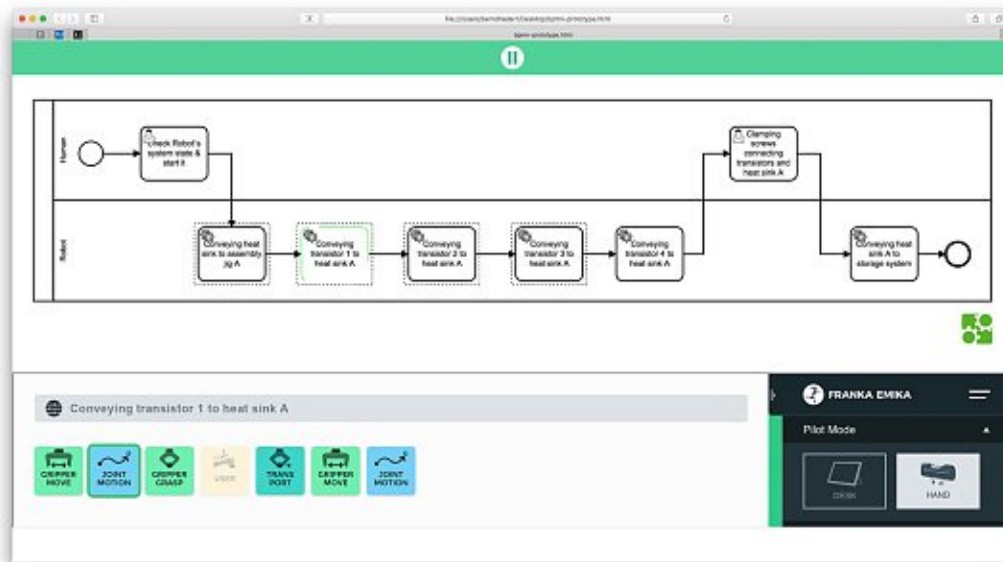


Figure 5.22: Process execution (own Figure)

Javascript is used to check every 100ms which tasks are currently "open" in the BPMN Engine. Based on this check, active tasks are animated with a green frame (implemented with CSS). With the help of this animation the viewers gaze is automatically directed to the current process progress.

As you can see in Figure 5.22, the play button, which has changed to a pause button, indicates that the BPMN process is currently running. Also all menus have disappeared. The green animated frame of the service task "Conveying transistor 1 to heat sink A" indicates that the robot is currently executing this task.

5.4.7 Req 6: Representation of additional information (work instructions) during execution

During the execution of the process, the user should have access to the information (work instructions) stored in Requirement 3. Therefore, the information is displayed in the lower part of the GUI during execution (see Figure 5.23).

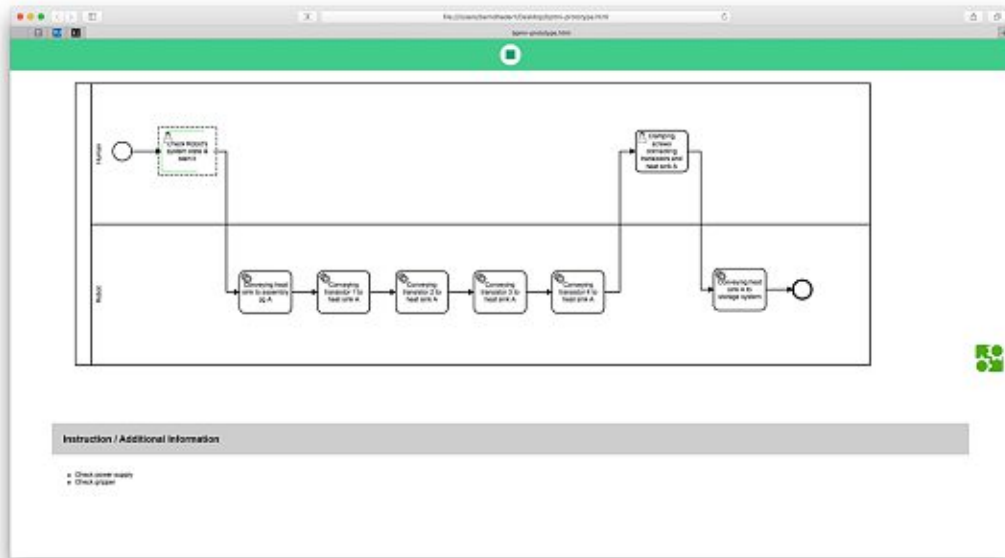


Figure 5.23: Representation of additional information during execution (own Figure)

5.4.8 Req 7: Detailed representation of robot tasks

Since a service task normally consists of several activities, the user should have a possibility to see in detail which action the panda is currently performing, e.g. service task "Conveying transistor 1 to heat sink A": the transport normally contains one or more move and pick and place commands. The service task gives little information about the task (move, graps, etc.) the robot is currently performing. For this purpose the Desk GUI was integrated into the prototype (see Figure 5.24 lower area). In this way, the user gets detailed information about the actions of the robot during the execution of a service task.

5.4.9 Req 8: User interaction (confirmation that the user has completed a specific task)

During execution, it is necessary for the user to be able to tell the system that he/she has completed a specific task. This has been implemented using Javascript, so that the user can complete the task by clicking on the green animated (currently open) task. In the background the user task is completed in the BPMN engine and the process continues.

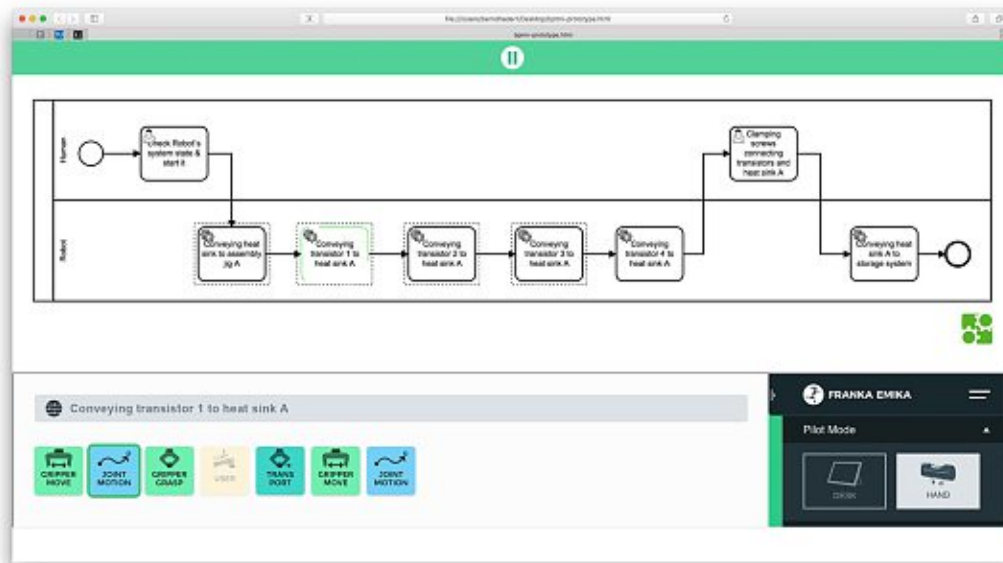


Figure 5.24: Integration of Desk UI (own Figure)

5.5 Code

The code as well as detailed instructions on how to install the individual components can be found on Github (<https://github.com/berndhader/BPMN-Extension-Franka-Emika-Desk>).



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Evaluation

In the course of the work, the goal was to create an intuitive prototype that should simplify the programming of collaborative robots (especially for people without programming experience). Whether the prototype (described in section 5) is an intuitive solution or not will be clarified in the following chapter.

6.1 Study Design

Originally, the study should be carried out in the pilot factory of the TU Wien. However, this was not possible due to the corona pandemic, so the experiment was conducted online. For this purpose the German website <https://ilovecobots.at> was created (see Figure 6.2). The switch to an online study had the following effects, which had to be taken into account:

- **No presence of the robot:** The participants did not have the opportunity to interact directly with the robot. In order to create a better understanding of what the robot does during its work steps, video sequences were created and shown during the execution of the tasks.
- **Change from face-to-face introduction to video tutorial:** In the pilot factory, there would have been a personal training of the participants before the experiment was conducted. This was replaced by a short video. The participants therefore had no opportunity to clarify questions/ambiguities. Through this one-way communication, an inadequate description of the task could thus have a direct impact on the evaluation of the system. To counteract this, special attention was paid to a clear description of the task. Furthermore, the study was already tested by pre-selected candidates during the development in order to eliminate possible ambiguities and errors in the description as well as the design of the website.

However, this switch to an online study had the great advantage of making it very easy to ensure that all participants had the same information and conditions. Each participant only had the information available on the website.

6.1.1 Hypothesis

The evaluation should essentially clarify the question of whether the BPMN prototype can be used by people (both with and without programming and BPMN experience) to quickly and easily program cobots in a simple and intuitive way.

Therefore, the following hypothesis was tested:

Hypothesis H1: The BPMN Prototype is intuitive and can be used without any special prior knowledge to create collaborative human-robot workflows.

6.1.2 Study Procedure

As already mentioned, the study was conducted online and the procedure was divided into the following sections:

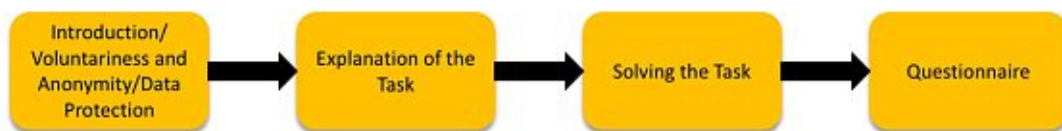


Figure 6.1: Study procedure (own Figure)

Introduction/Voluntariness and Anonymity/Data Protection

On the landing page of the website, participants were first given a brief introduction to the topic of collaborative robots. This was followed by detailed information on the data collected (topic: data protection regulation), the anonymous storage of data and the voluntary nature of participation (see Figure 6.3).



Figure 6.2: Landing page of <https://ilovecobots.at> (own Screenshot)



Figure 6.3: Introduction/Voluntariness Anonymity/Data Protection (own Screenshot)

To proceed to step 2 "Explanation of the Task" users had to actively confirm that participation was voluntary and that they accepted the privacy policy.

Explanation of the Task

In the second step, a short video explaining the prototype was provided to the participants. Furthermore, the user was also given a description of the task (assembly of a heat sink) to be solved.

Description of task: Assembly of a heat sink

1. **Human task:** Name: Check robot
2. **Robot task:** Name: Transport heat sink, robot command: TransportKuehlkoerper
3. **Robot task:** Name: Transport transistor 1, robot command: TransportTransistor1
4. **Robot task:** Name: Transport transistor 2, robot command: TransportTransistor2
5. **Human task:** Name: Screw on transistors
6. **Robot task:** Name: Transport heat sink, robot command: AbtransportKuehlkoerper



Figure 6.4: Video tutorial for the BPMN prototype (own Figure)

Afterwards, the participant had to actively confirm that he/she wanted to start the "Programming part". This was necessary so that the time needed to complete the task could be collected in the background (but more on this in the section "Further Evaluation Data").

Solving the Task (Programming part)

In step 3 of the study procedure, the prototype was used (see Figure 6.5). The participant's task here was to model the "Assembly Heat Sink" process.

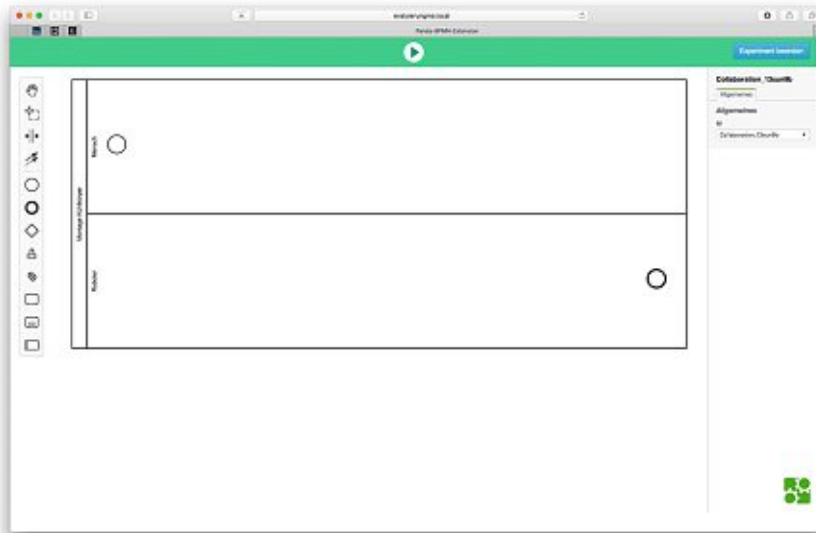


Figure 6.5: Study procedure part 3: Solving the task (own Figure)

The correct solution was as follows:

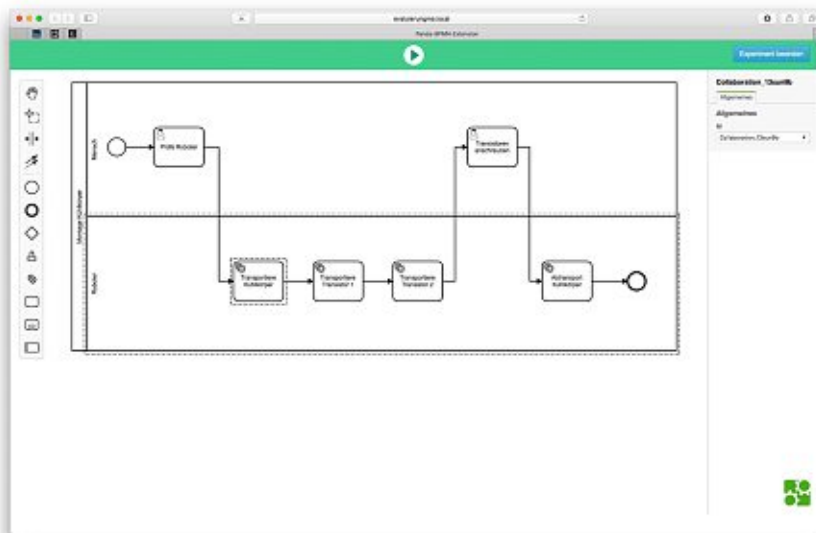


Figure 6.6: Correct solution "Assembly Heat Sink" (own Figure)

During the creation of the workflow, the users could also execute it using the play button. In the pilot factory, the robot would have executed the service tasks. In order to give the participants an understanding of this division of labour and the tasks of the robot online as well, short video sequences of the individual robot tasks were created in the pilot factory and shown during the execution of the process.

As it can be seen in Figure 6.7, the robot is currently carrying out the task "Transport Heat Sink". Accordingly, the corresponding video is played in the lower part of the website.

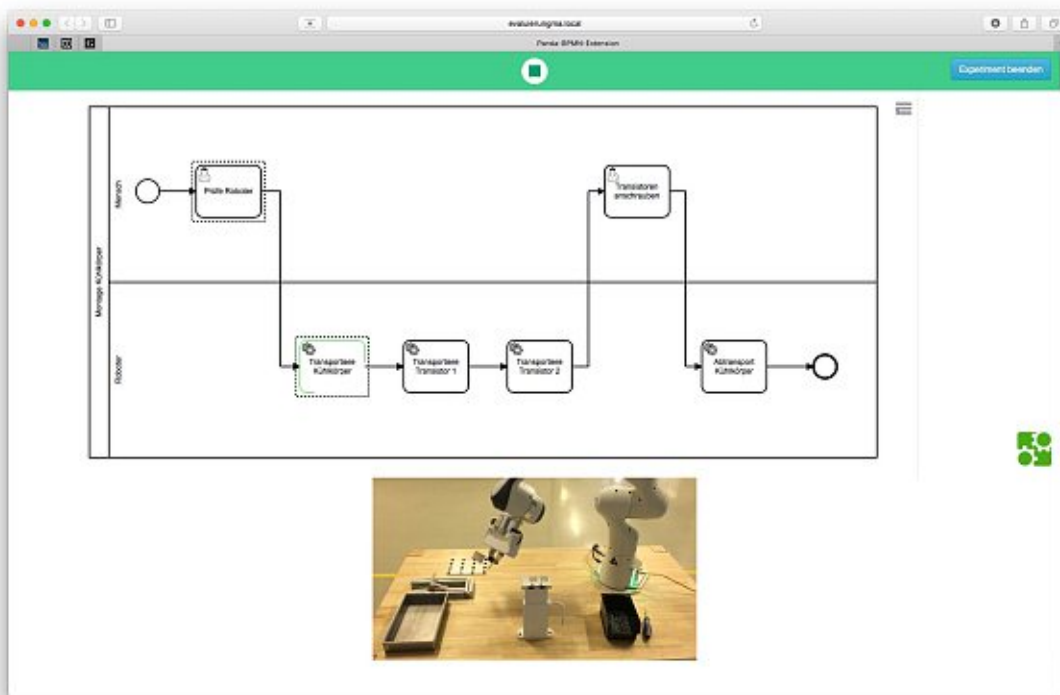


Figure 6.7: Visualisation of the robot tasks by means of video sequences (own Figure)

Questionnaire

The questionnaire (see Appendix B) consisted mainly of two parts, a demographic part and the evaluation of the prototype.



Figure 6.8: Online Questionnaire (Demographic data) (own Screenshot)



Figure 6.9: Online Questionnaire (Evaluation of the prototype) (own Screenshot)

Demographic Part

The following data were collected:

- Occupation
- School education
- Gender
- Age
- Technology affinity
- Experience with collaborative robots
- Programming knowledge
- Experience with process modelling
- Experience with BPMN

In particular, the data on prior experience with collaborative robots and process modelling languages, especially BPMN, should provide information on whether the prototype is perceived as simple and intuitive also for people without experience in these areas.

Evaluation of the Prototype

In order to evaluate whether the prototype has a good usability and is intuitive, the System Usability Scale and the NASA Task Load Index were used.

System Usability Scale: The System Usability Scale, SUS for short, by John Brooke is a widely used method for determining the usability of a system. It is a questionnaire comprising 10 questions with 5 point Likert scale. Based on the answers, the SUS score is calculated, which provides a value between 0 and 100, where 0 means that the system has a very poor usability and 100 a very good (excellent/perfect) usability. [29]

The great advantage of the System Usability Scale is that due to the very general questions, it is technology-independent and can therefore be used for a wide range of systems.

NASA Task Load Index: The NASA Task Load Index (Nasa-TLX) is a method for assessing the workload of a system or a task [30]. The NASA-TLX essentially consists of two parts [30]:

- Rating: The following six dimensions are assessed:
 - Mental Demand
 - Physical Demand
 - Temporal Demand
 - Performance
 - Effort
 - Frustration

A 20-item scale is used, with each item representing 5%, so that each dimension is assigned a value between 0 and 100%.

- Weighting: Here, the user has to define in pairs the importance of the dimensions.

Based on the weighting multiplied by the respective dimension, an overall workload score between 0 and 100 can be calculated [30]. Various studies have shown, that it is not clear what influence the weighting has [193]. For this reason and for the reason of simplicity, this second part (the weighting) is often omitted [193]. Therefore, the Raw TLX (RTLX), was also used in the work. Furthermore, the 20-item scale was replaced by a 5 point Likert scale.

Data was collected on how long the users can imagine working with the system and whether they would prefer to use the application via tablet.

The question "I felt very comfortable interacting with the system." served as a control question for the usability evaluation (SUS).

Finally, the following two open questions were added to collect problems and comments from the participants:

- Have you encountered any problems when using the system?
- Do you have any other comments?

6.1.3 Further Evaluation Data

In addition to the data collected through the questionnaire, the following data was collected:

- **Duration to solve the task:** This is the time it took users to solve the task. This was measured by actively clicking on the buttons "Start programming" and "Finish task".
- **Solution:** The created models were saved in the database and then manually checked for correctness.

6.1.4 Technical Implementation

The study was conducted entirely on the website <https://www.iLoveCobots.at>. PHP and the framework Laravel [194] were used as programming language. HTML, CSS and Javascript were used for the visualisation. A MySQL database was used to store the data. As mentioned above, the BPMN prototype was adapted so that videos of the robot were added to simulate the actions of the collaborative robot. The Camunda BPMN engine as well as the NodeJS Task Client, which were needed for the execution of the BPMN processes, were realised using Amazon Web Services. The questionnaire was also realised with the help of HTML, CSS, Javascript, PHP and MySQL and integrated directly into the website. This ensured that only people who used the prototype and tried to solve the task had access to the questionnaire.

6.2 Participants

6.2.1 Data Cleansing

In total, 64 entries (participants) were registered in the database, which actively opened the link to the prototype. Of these 64 entries, 51 participants completed the task as well as the questionnaire. If, for example, a participant starts the task, interrupts it in the meantime and then carries it out again on the same device with the same browser, this was detected as one participant.

This was implemented with the help of Sessions. A unique ID is assigned to the visitor of the website, which is stored on his laptop/PC. When the website is opened up again, this session ID (if available) is also transmitted to the web server and the web server can thus retrieve the user's stored information. This concept is also used, for example, to store products in the shopping basket of online shops. [195]

With the help of these sessions, two things were implemented. Firstly, this gave the user the opportunity to complete the task or questionnaire at a later time. In addition, this mechanism was intended to ensure in a simple way that a user did not participate in the study more than once. If the participant opens the page after submitting the questionnaire, the web server recognises this and the user is automatically redirected to the page "Thank you for your participation".

Of course, this mechanism cannot completely exclude multiple participation. For example, a user could participate more than once with a different browser, a different device or by means of a private session.

Since it is not possible to understand why these 13 visitors abandoned the study or "strayed" to the website by mistake. For example, entries could also have been created by a web crawler that searches the internet (e.g. search engines use such programs). Due to the lack of traceability, the data was cleaned and these 13 entries were removed.

6.2.2 Demographic Data

In sum **51 people** participated in the study, 21 of whom were female and 30 male (see Figure 6.10). As it can be seen in Figure 6.11 the majority of participants were employees (56.9%) and students (35.3%). The age distribution (see Figure 6.12) showed that the range covers almost all age groups, except for the group up to 19 years, where not a single participant was represented. Most participants (31 persons) were between 20 and 29 years old, followed by the age group 30-39 (10 persons).

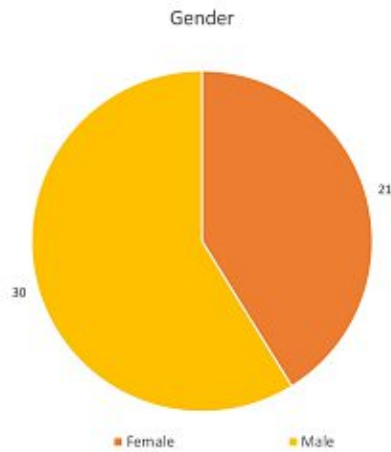


Figure 6.10: Gender (n=51) (own Figure)

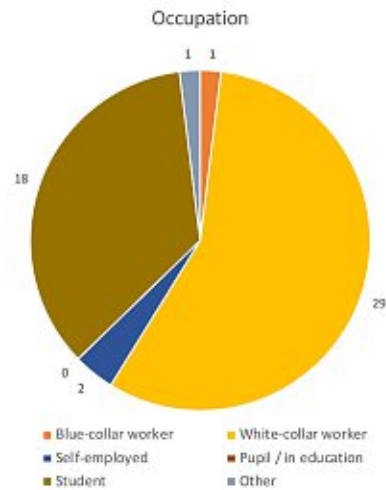


Figure 6.11: Occupation (n=51) (own Figure)

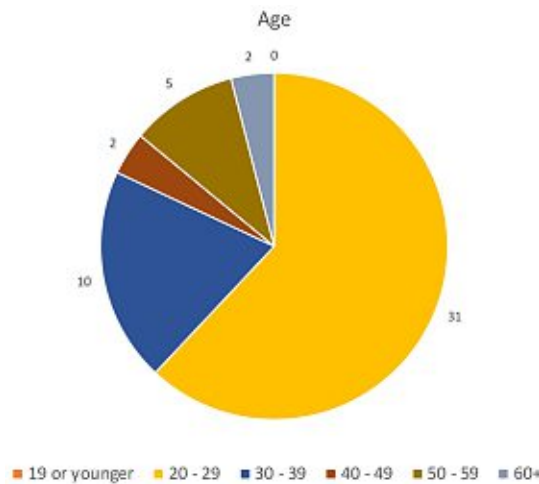


Figure 6.12: Age (n=51) (own Figure)

Experience

The evaluation of previous experience showed that 37 people (72.5%) have never worked with a collaborative robot before (see Figure 6.13). The majority of participants (70.6%) had no or only basic programming skills. As you can see in Figure 6.14, most participants (19) had basic coding skills, followed by 17 people who had no experience with programming at all.

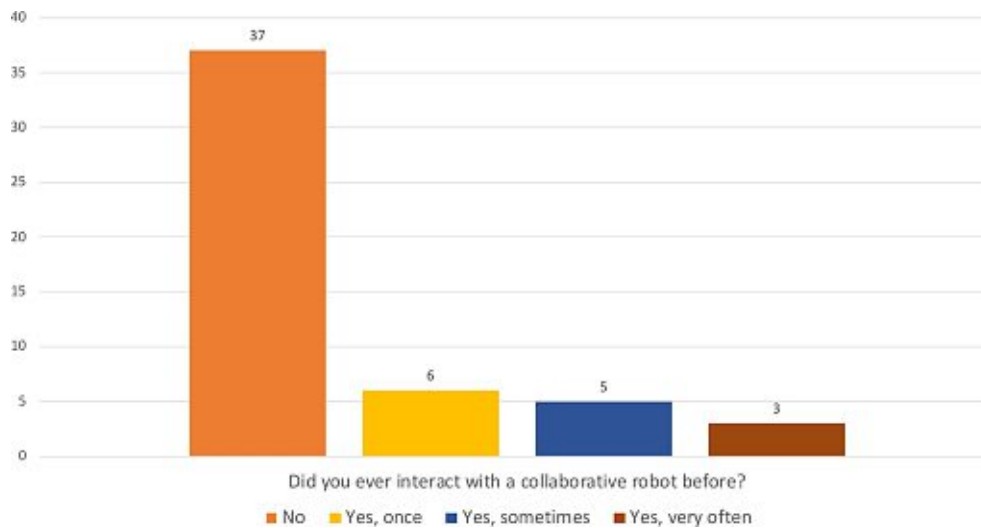


Figure 6.13: Experience with collaborative robots (n=51) (own Figure)

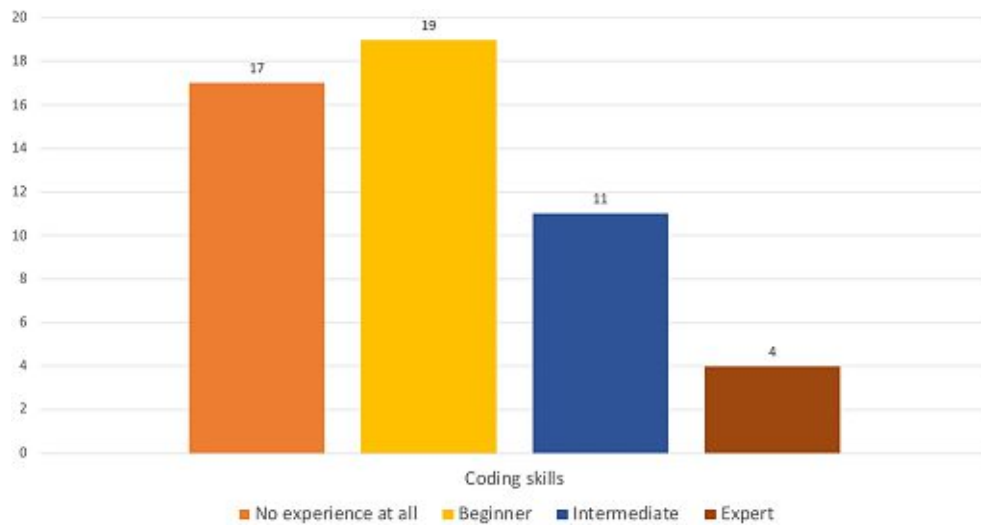


Figure 6.14: Programming experience (n=51) (own Figure)

58.8% (see Figure 6.15) already had experience with process modelling languages such as (UML, BPMN, etc.). However, it was interesting to note that about half of these people with process modelling experience have never worked with BPMN before. In relation to all participants, 64.7% (33 persons) had no previous experience or knowledge of BPMN (see Figure 6.16).

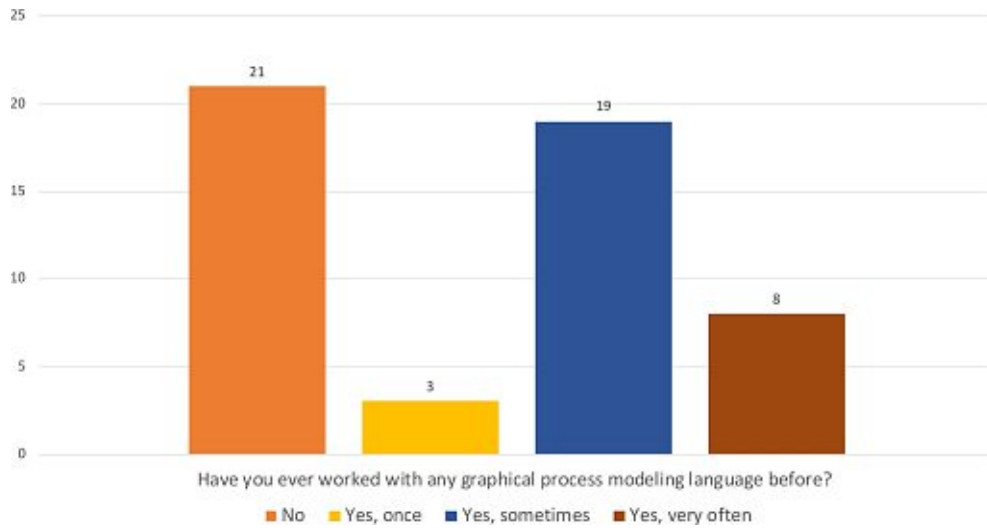


Figure 6.15: Experience with process modelling (n=51) (own Figure)

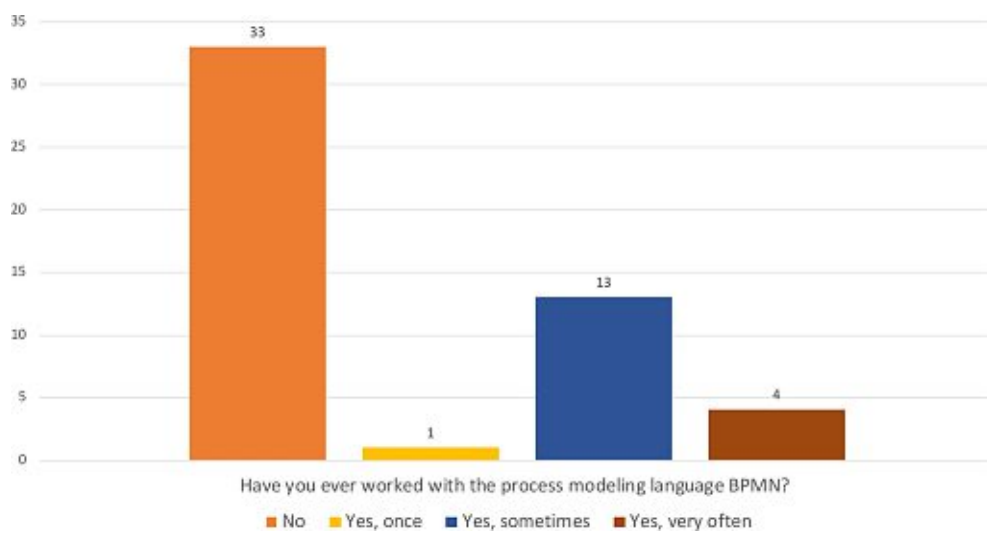


Figure 6.16: Experience with BPMN (n=51) (own Figure)

6.3 Results

6.3.1 Task solved correctly, Duration

As the diagram in Figure 6.17 shows, 43 participants (84.3%) were able to solve the task "Assembly Heat Sink" correctly. Particular attention was paid to determine the difficulties people had in solving the task. For this purpose, the incorrect solutions were analysed and the following types of errors were identified. Four people had problems modelling the programme flow. The most common cause were missing connections, especially to the start and end points of the process (circles). Furthermore, three persons had problems with the assignment of the robot tasks, these were either wrong (e.g. TransportTransistor1 instead of TransportTransistor2) or were forgotten. Finally, one participant had problems using the correct tasks (e.g. Human Task instead of Service Task). In summary, this means that the greatest difficulties were thus encountered in the creation of the programme flow, followed by the assignment of the robot tasks.

On average, the participants needed 7 minutes and 43 seconds for completing the task. As it can be seen in Figure 6.18, 75% of the participants were able to complete the task in a time between 3:07 and 8:39 minutes. For comparison, an expert needed 3:30 minutes for the same task. On average, users thus took twice as long. Assuming that the learning curve of the prototype is relatively steep (due to the very good usability (see section 6.3.2), it can be assumed that the time should be much better with repeated use of the prototype.

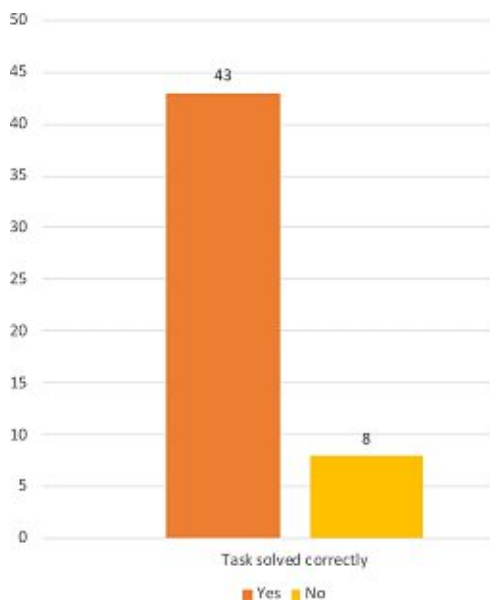


Figure 6.17: Correct Solution (n=51) (own Figure)

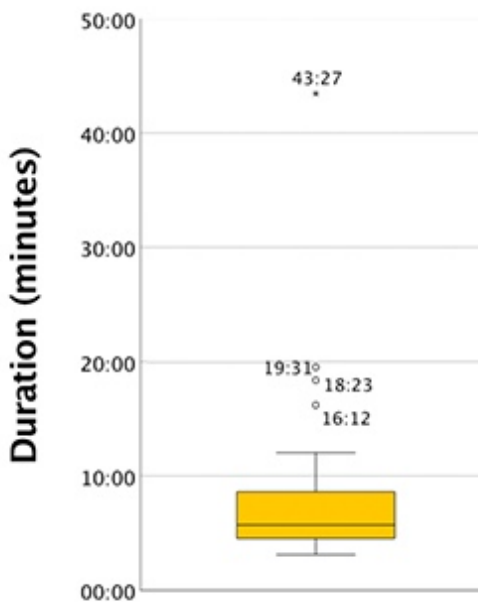


Figure 6.18: Duration for solving the task (own Figure)

6.3.2 Usability

As already mentioned above, the System Usability Scale (SUS) by John Brooke [29] was used to assess the usability. The system achieved a score of 86.

What does this value mean in concrete terms for assessing the usability? Bangor et al. [196, p.121] have created an adjective-based rating for the System Usability Scale. Based on this classification systems with a SUS-Score of 85 or higher are considered to have excellent usability.

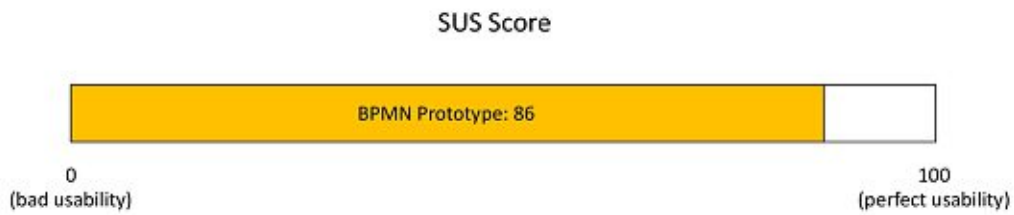


Figure 6.19: SUS-Score BPMN Prototype (own Figure)

The results of the Nasa-TLX (see Figure 6.20) show that Mental, Physical and Temporal Demand as well as Effort and Frustration were rated very low (5 point likert scale, where 1=very low and 5=very high). In contrast, the dimension "performance" was rated very high with an average value of 4.5.

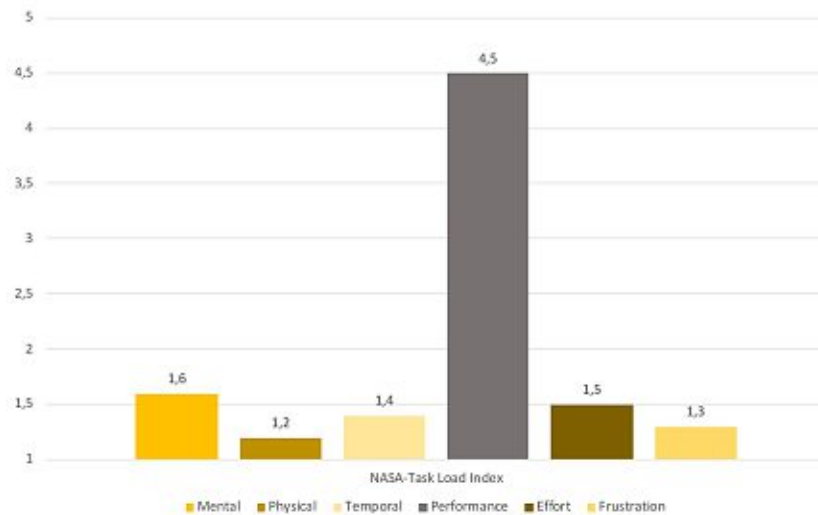


Figure 6.20: Nasa-TLX (own Figure)

As Naumann et al. [76] define, the goal of intuitive user interfaces is to reduce the user's cognitive effort so that they can concentrate on other tasks. Therefore, the result of the Nasa-TLX underlines that the created BPMN-Prototype is intuitive, as the participants were able to solve the task with very little effort (mental, physical etc.). Subsequently, the participants were very successful according to their self-assessment. The evaluation of the results (see section 6.3.1) shows that the users were indeed very successful.

In summary, this means that the BPMN prototype has an excellent usability, is easy to use even for people without experience and thus enables even laypersons to program collaborative human-robot processes. **Based on the data of the System Usability Scale (SUS) and the Nasa Task Load Index (Nasa-TLX), the hypothesis (H1: The BPMN Prototype is intuitive and can be used without any special prior knowledge to create collaborative human-robot workflows.) is supported.**

6.3.3 Further Results

The question of whether users would prefer to use the system via tablet was answered in the affirmative by 16% (see Figure 6.21) of the participants, while 48% declined this. The result was somewhat surprising, but very clearly confirms the preference for using the prototype via PC instead of a tablet.

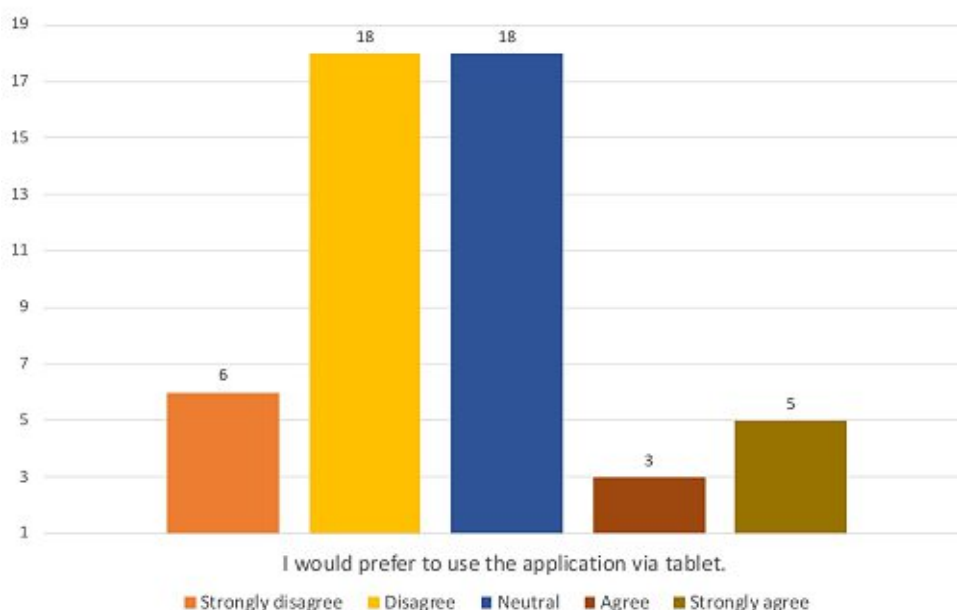


Figure 6.21: Tablet usage (n=50) (own Figure)

Furthermore, the evaluation of the results has shown that there is a significant positive correlation (Spearman-Rho, $r = 0.422$, $p = 0.02$) between the experience with BPMN

and the usability rating (SUS). This seems logical and means that people who have already worked with BPMN rated the usability of the prototype better. However, it is interesting to note that no significant correlation ($r = 0.110$, $p = 0.561$) could be identified between the experience with BPMN and the correct completion of the task. This shows that people without BPMN experience were as successful in solving the task as people with BPMN experience, which again underlines the simplicity and intuitiveness of the prototype.

6.3.4 Qualitative Data (Comments and Issues)

The evaluation of the two open questions showed that the majority of users did not notice any problems while using the system.

Some users had detected a bug in the system, where the name of the robot task was not assigned correctly in a specific case. Furthermore, two participants had space problems when using the system and found the additional menu (additional instructions for the human tasks) annoying. In addition, there were isolated problems with the execution of the process models. These points/errors should be closely examined and corrected in a further iteration.

The additional comments were very positive, such as (the answers were originally in German and have been translated for the following listing):

- Very good visualisation with the videos and great interaction of the user to visually imagine the process flow
- Very good implementation of robot programming
- This system runs solidly and I would like to work with it
- Easily understandable system

This data also confirms that the majority of users were able to solve the task without problems and that the system was perceived as very simple and understandable. The identified issues should be investigated and corrected in a further iteration.

Conclusion, Discussion & Outlook

7.1 Conclusion

The main problem of existing programming environments for collaborative robots is that they concentrate on the programming of the robot and disregard the process level and thus in particular the tasks of the human. As we have seen in Chapter 3 State of the Art, theoretical approaches exist that implement this mapping of the entire collaborative human-robot process. However, it is uncertain to what extent these approaches are suitable for the users (e.g. the workers). For this reason, the aim of the work was to create an intuitive graphical user interface (GUI) and subsequently a working prototype that could be used to evaluate whether this type of programming/modelling of collaborative human-robot processes is suitable for users, especially for those without any particular previous experience in the field of programming and robotics.

The two research questions that should be answered in the course of the work were:

- RQ1: What does an intuitive robot programming interface look like that can simultaneously serve as a worker assistance system?
- RQ2: What is an appropriate means to implement such a system?

In order to develop a prototype that meets the requirements of the future users and is therefore particularly intuitive/easy to use, a user-centred design method (UCD) was chosen, specifically the Human-centred Design Approach for Interactive Systems [14] of the International Organization of Standardization (ISO). By integrating the users into the whole process, in the form of a focus group consisting of experts and users, a web-based BPMN (Business Process Model and Notation) [86] prototype (see Figure 7.2)

was developed in two iterations. For this purpose, the existing two-layer architecture of the cobot Panda by Franka Emika [6] was extended by a third layer, the workflow layer (see Figure 7.1).

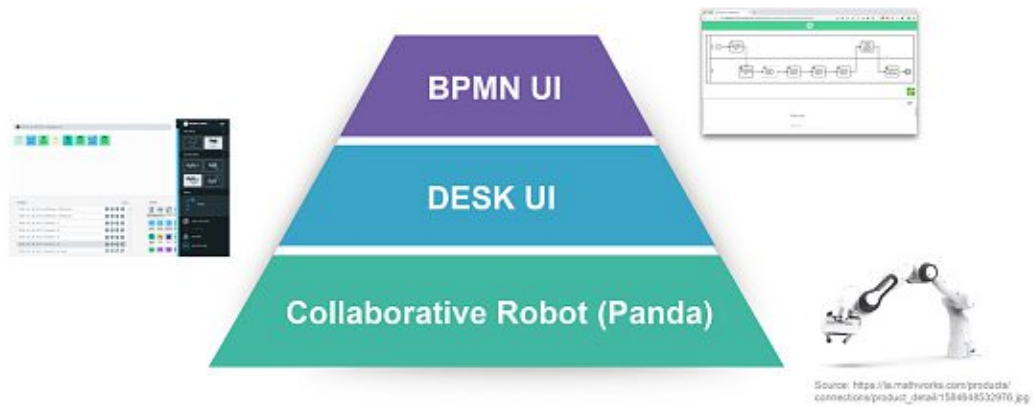


Figure 7.1: Extended 3-Layer Architecture (own Figure)

The final prototype (see Figure 7.2 & 7.3) enables:

- the modelling & execution of entire collaborative human-robot process
- the control of the robot
- the storage of additional information (for human tasks)
- the worker to be assisted during execution of the process.

The prototype was implemented by the use of a so-called BPMN engine, specifically Camunda [100]. Detailed information on the implementation can be found in Chapter 5 Software Architecture.

An essential part of the work was the evaluation of the prototype in terms of user-friendliness / usability / intuitiveness. Therefore, an online user study was conducted and the usability of the system was evaluated using the System Usability Scale (SUS) [29] and the NASA Task Load Index (Nasa-TLX) [30]. The results of the evaluation showed that the BPMN prototype has an excellent usability, is easy to use even for people without experience, and thus enables even laypersons to program collaborative human-robot processes.

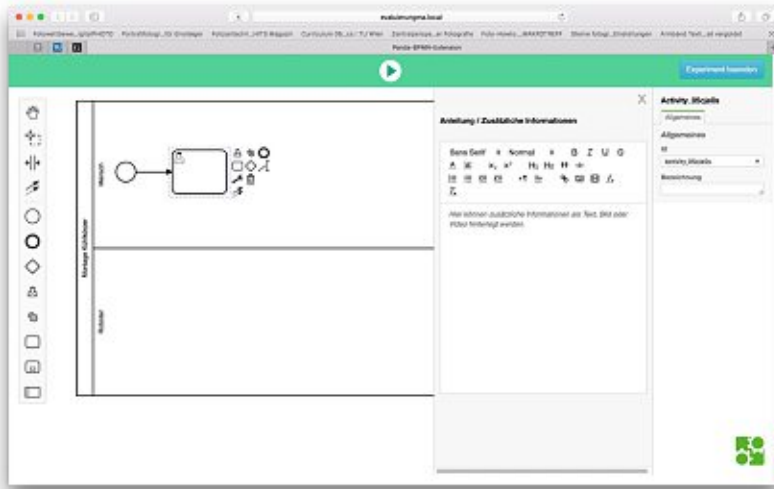


Figure 7.2: Final GUI of the BPMN prototype (own Figure)

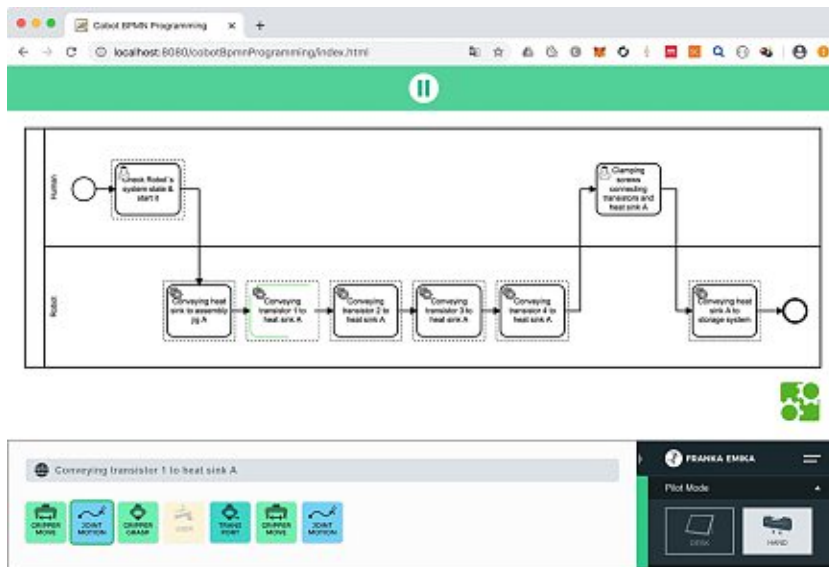


Figure 7.3: Execution of the collaborative process (own Figure)

7.2 Discussion & Outlook

Due to the corona pandemic, the user study was conducted online instead of directly on the robot. To what extent this affects the usability assessment should be tested in a further evaluation and compared with the results of the thesis. Furthermore, the participants of the study were mainly students and employees and not workers. An evaluation should also be carried out with this user group.

Within the scope of the thesis, it was not possible to implement all the points developed by the focus group.

Specifically, the following three features could not be implemented (see Chapter 4.2.3):

- **Parameterisation of tasks:** A possible extension would be the parameterisation of robot tasks. In this way, instead of reteaching in the Desk UI, e.g. a move task with start and end coordinates could be defined in the BPMN UI.
- **Predefined standard processes:** Based on the previous point, an extension could be the integration of entire standard processes such as a Pick&Place task, where only the individual parameters need to be adjusted. This would allow inexperienced users to create complex processes very quickly.
- **Integration of a stopwatch as well as the display of reference times:** This would enable the employee to assess whether he/she is too fast or too slow in carrying out his/her tasks. Furthermore, it would be useful to have an estimated time until completion for the robot tasks, so that the worker knows when he/she has to take over the work.

In a next step, these points could be implemented and evaluated on the basis of the existing prototype. In a further step, it could also be investigated to what extent parameterisation with BPMN is possible and whether the GUI (after the integration of this functionality) is still perceived as simple by the users.

Other possible ideas for future research in this direction that were collected during the course of the work concern the following two topics:

- **Security:** The topic of security was not considered in this work. However, with increasing networking (also in the industrial environment), this topic is becoming more and more important and should be addressed.
- **Dynamic Task Allocation:** The prototype created forms a good basis for the creation of simple collaborative human-robot processes. In a next step, it could be extended so that the division of labour between humans and robots can take place dynamically during runtime.

APPENDIX **A**

Focus Group Invitation



Einladung zur Fokusgruppe zum Thema

„Intuitive Programmierung von kollaborativen Mensch-Roboter-Prozessen“



Im Zuge meiner Masterarbeit zum Thema „Intuitive Programmierung von kollaborativen Mensch-Roboter-Prozessen“ werden Mitglieder für eine Fokusgruppe gesucht.

Das Ziel der Arbeit ist die Erstellung eines Prototyps, welcher die Programmierung von kollaborativen Robotern verbessern soll. Hierfür ist zunächst die Erarbeitung und Diskussion von aktuellen Problemen mit existierenden Lösungen ein Schwerpunkt. Schließlich soll ein Prototyp entwickelt werden welcher in zwei weiteren Sitzungen diskutiert und weiterentwickelt wird.

Gesucht werden Teilnehmer/innen, welche Erfahrung mit dem Einsatz oder der Programmierung von kollaborativen Robotern haben und an der Entwicklung von neuen intuitiven Lösungen interessiert sind.

Wann: 3 Meetings im Zeitraum Mai/Juni

Dauer: 1,5 - 2 Stunden pro Meeting

Wo: Online-Meeting

Ich würde mich über Ihre Teilnahme freuen und bedanke mich schon vorab für Ihre Unterstützung!

APPENDIX **B** 

Questionnaire

Fragebogen

Demografische Daten

1. Wie lässt sich Ihr derzeitiger Beruf einordnen?

- ArbeiterIn
- Angestellte(r)
- Selbstständig
- Schülerin / in Ausbildung
- StudentIn
- Sonstige
- keine Angabe

2. Was ist bislang Ihr höchster Bildungsabschluss?

- Hauptschule, Unterstufe AHS, NMS, Sonderschule
- Lehre
- Berufsbildende Mittlere Schule (BMS) wie Handelsschule, Fachschulen
- Matura
- Uni/FH
- keine Angabe

3. Bitte geben Sie Ihr Geschlecht an:

- Weiblich
- Männlich
- Divers

4. Bitte geben Sie Ihre Altersgruppe an:

- 19 oder jünger
- 20 - 29
- 30 - 39
- 40 - 49
- 50 - 59
- 60+

5. Wie schätzen Sie sich in Bezug auf Technik ein?

	Stimme überhaupt nicht zu	Stimme nicht zu	weder noch	Stimme zu	Stimme voll zu
Ich habe ein gutes Verhältnis zu Technik und Maschinen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Das Erlernen von neuen Technologien fällt mir leicht.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich weiß, wie man mit technischen Problemen umgeht.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich löse gerne technische Probleme, sehe dies als Herausforderung und habe Spaß daran.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich bin der Meinung, dass die meisten Technologien leicht zu erlernen sind.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
In Bezug auf neue Technologien bin ich Up-to-date.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

6. Haben Sie schon einmal mit einem kollaborativen Roboter gearbeitet?

- Ja, sehr oft
- Ja, einige Male
- Ja, einmal
- Nein

7. Bitte bewerten Sie Ihre Programmierkenntnisse anhand der folgenden Skala:

- Experte (Ich beherrsche zumindest eine Programmiersprache auf einem Expertenlevel)
- Fortgeschrittene Kenntnisse (Ich kann programmieren, bin aber kein Experte)
- Anfänger (Ich habe einen Programmierkurs besucht und/oder verfüge über Basiskenntnisse)
- Ich verfüge über keine Programmierkenntnisse

8. Haben Sie bereits mit einer grafischen Prozessmodellierungssprache wie zB UML, Aktivitätsdiagrammen, Erweiterte Prozessketten, BPMN oder Flussdiagrammen gearbeitet?

- Ja, sehr oft
- Ja, einige Male
- Ja, einmal
- Nein

Bewertung der Usability des getesteten Systems

1. Bitte bewerten Sie Ihre Benutzererfahrung mit dem System im Hinblick auf die folgenden Aussagen.

	Stimme überhaupt nicht zu	Stimme nicht zu	weder noch	Stimme zu	Stimme voll zu
Ich denke, dass ich das System gerne häufig benutzen würde.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich fand das System unnötig komplex.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich fand das System einfach zu benutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich denke, das System enthielt zu viele Inkonsistenzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich fand das System sehr umständlich zu nutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich fühlte mich bei der Benutzung des Systems sehr sicher.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

B. QUESTIONNAIRE

2. Bitte bewerten Sie Ihre Ansichten über das System im Hinblick auf die folgenden Aussagen.

	Stimme überhaupt nicht zu	Stimme nicht zu	weder noch	Stimme zu	Stimme voll zu
Die Nutzung des Systems ist für mich eine neue Erfahrung.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Nutzung des Systems ist nicht mit dem vergleichbar, was ich bisher gemacht habe.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Nutzung des Systems unterscheidet sich von anderen Erfahrungen, die ich gemacht habe.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Nutzung des Systems ist für mich eine neue Arbeitserfahrung.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Anwendung des Systems in der Praxis würde meinen eigenen Werten zuwiderlaufen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verwendung des Systems passt nicht zu meiner Sicht der Welt.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verwendung des Systems widerspricht dem, wofür Computer meiner Meinung nach eingesetzt werden sollten.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verwendung des Systems ist für eine Person mit meinen Wertvorstellungen bezüglich der Rolle von Computern nicht geeignet.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Nutzung des Systems widerspricht meinen Wertvorstellungen darüber, wie gearbeitet werden sollte.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verwendung eines solchen Systems würde meiner bevorzugten Arbeitsroutine entsprechen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Das System würde es mir ermöglichen, in der von mir bevorzugten Weise zu arbeiten.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Nutzung des Systems würde gut zu der Art und Weise passen, wie ich gerne arbeite.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Die Verwendung des Systems würde meiner bevorzugten Arbeitsmethode entsprechen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. Bitte bewerten Sie Ihre Erfahrungen bei der Lösung der Aufgabe mit dem System im Hinblick auf die folgenden Aussagen.

	Sehr (5)	(4)	Mittel (3)	(2)	Kaum (1)
Wie anspruchsvoll (mental) war die Aufgabe?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Wie anspruchsvoll (physisch) war die Aufgabe?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Wie eilig oder überstürzt war das Tempo der Aufgabe?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Wie erfolgreich waren Sie beim Lösen der Aufgabe?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Wie sehr mussten Sie sich anstrengen um das Ziel zu erreichen?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Wie unsicher, entmutigt, gereizt, gestresst und verärgert waren Sie?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Welche Probleme konnten Sie während der Benutzung des Systems feststellen ?

5. Wie viele Minuten könnten Sie sich vorstellen, mit dem System [...] zu arbeiten?

	unter 5 Min.	5 - 15 Min.	15 - 30 Min.	30 - 60 Min.	mehr als 60 Min.
... ohne Pause ...	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... pro Tag mit Pausen ...	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

6. Bitte bewerten Sie abschließend noch folgende zwei Aussagen.

	Stimme überhaupt nicht zu	Stimme nicht zu	weder noch	Stimme zu	Stimme voll zu
Ich habe mich bei der Interaktion mit dem System sehr wohl gefühlt.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich würde die Anwendung lieber mittels Tablet nutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

7. Haben Sie sonst noch Anmerkungen?




Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

1.1	Programming interface of cobot Panda by Franka Emika [8]	3
1.2	Collaborative Human-Robot Assembly Process with BPMN 2.0 (own Figure)	3
1.3	ISO standard 9241- 210, User-Centered	
	Design process for interactive systems [10, p.15]	5
2.1	Automated production using industrial robots [42]	15
2.2	Production line of the Audi A4 in Ingolstadt [43]	15
2.3	Annual installations of industrial robots 2009-2019 [45]	16
2.4	Annual installations of industrial robots by regions [46]	16
2.5	Annual installations of industrial robots by industries [46]	17
2.6	Typical cell of an industrial robot [38, p.42]	18
2.7	Design of a collaborative workplace [38, p.43]	18
2.8	Types of Human-Robot-Interaction by Bauer et al. [2, p.9]	19
2.9	Types of Human-Robot-Collaboration by Zaatari et al. [51]	20
2.10	Sales of classical industrial robots vs. cobots (2017-2019) [52]	22
2.11	Modular assistance system for manual assembly [62]	25
2.12	Worker Assistance System "Computer Aided Works" [63]	25
2.13	Interdisciplinarity of HCI [71]	29
2.14	The first commercial GUI, developed by Xerox [75, p.28]	30
2.15	BPMN example: bank credit quote creation process [93]	37
2.16	BPMN "Swimlanes" concept for collaborative human-robot processes (own Figure)	39
2.17	Activity [86, p.29]	39
2.18	Start/End event [86, p.31]	40
2.19	Parallel/Exclusive Gateway [86, p.34]	40
2.20	Sequence Flow [86, p.29]	40
3.1	Online Programming via teach-pendant [104]	44
3.2	Offline Programming with 3D simulation software [105]	44
3.3	Automatic programming with Programming by Demonstration [107]	45
3.4	Manual programming (graphical) with Lego Mindstorms [108], [109]	45
3.5	GUI of Universal Robots teaching pendant [111]	47
3.6	Example of a Morpha program (left), Kuka specific implementation according to the (withdrawn) ISO standard 15187 (right) [112]	48

3.7	GUI of RobotStudio [120]	49
3.8	GUI of CoppeliaSim [121]	49
3.9	Task-based GUI Razor [123]	49
3.10	Intuitive GUI for the cobot YuMi [129]	56
3.11	Intuitive skill-based graphical user interface [130]	57
3.12	Workflow-based programming with Choreoid [131, p.8]	58
3.13	Example workflow for the assembly of the main wing of a toy airplane [131, p.13]	59
3.14	User Interface of Blockly [133]	61
3.15	Extended Blockly App [56, p.96]	62
3.16	Injecting Blockly Code into Desk [56, p.97]	62
3.17	Cobot programming with Blockly [134]	63
3.18	Example of a centurio.work BPMN Process (adapted from [136])	65
3.19	Collaborative workflow modeled with HRTM [138]	67
3.20	Models, Meta-Models, Meta-Meta-Models [145]	69
3.21	Meta-Meta-Model ADAPT [143, p.561]	71
3.22	Example of a Meta-Model based on the ADAPT Meta-Meta-Model [143, p.561]	72
3.23	Interface of ADAPT Designer [143, p.563]	73
3.24	Programming interface of Universal Robots [148]	74
3.25	Programming interface of Universal Robots [148]	75
3.26	Programming interface of Kuka iiwa [152]	76
3.27	Programming interface of Kuka iisy [153]	76
3.28	Programming interface of Fanuc (CR models) [155]	76
3.29	Programming interface of Fanuc (CR models) [155]	76
3.30	GUI of Fanuc (CRX models) [156]	77
3.31	Programming interface of Fruitcore Robotics [157]	77
3.32	GUI of Fruitcore Robotics [158]	77
3.33	Programming interface (Desk) of Franka Emika (own Figure)	78
3.34	Flow-based GUI of Techman Robot [160]	79
3.35	ABB's cobot YuMi available with one or two arms [162]	79
3.36	Programming YuMi online via teach pendant [163]	79
3.37	Text-based programming with RAPID [164]	80
3.38	"Wizard Easy Programming" [164]	80
3.39	Programming interface of Drag&Bot [165]	80
3.40	3D Simulation with Drag&Bot [165]	80
3.41	Programming with ROS [168]	81
4.1	3-layer architecture (own Figure)	88
4.2	First draft of the GUI (own Figure)	89
4.3	Creation of human tasks (own Figure)	90
4.4	Creation of robot (service) tasks (own Figure)	91
4.5	Creation of robot (service) tasks (own Figure)	91

4.6	Execution of the collaborative process (own Figure)	92
4.7	Storage of additional information (work instructions) (own Figure)	94
4.8	Representation of the work instructions during execution (own Figure)	95
4.9	Simplification of the robot task selection	95
4.10	Simplification of the user interface (own Figure)	96
5.1	Software Architecture (own Figure)	97
5.2	Example of a collaborative assembly process modeled with BPMN (own Figure)	98
5.3	Running process in the BPMN engine (own Figure)	99
5.4	Running process in the BPMN engine (own Figure)	99
5.5	REST API Request "Process Deployment" (own Figure)	103
5.6	REST API Response "Process Deployment" (own Figure)	103
5.7	Overview Camunda Deployments (own Figure)	104
5.8	REST API Request "Start Process Instance" (own Figure)	105
5.9	REST API Response "Start Process Instance" (own Figure)	105
5.10	Successfully started BPMN process (own Figure)	105
5.11	REST API Request "Get list of open service tasks" (own Figure)	106
5.12	REST API Response "Get list of open service tasks" (own Figure)	106
5.13	REST API Request "Get list of open (user) tasks" (own Figure)	107
5.14	REST API Response "Get list of open (user) tasks" (own Figure)	107
5.15	REST API Request "Set status of user task to completed" (own Figure)	108
5.16	REST API Response "Set status of user task to completed" (own Figure)	108
5.17	"Desk" User Interface of the Cobot Panda by Franka Emika 	109
5.18	REST API Response "Start Panda Task"	111
5.19	Web-based BPMN Modeler (bpmn-js) from bpmn.io (own Figure)	115
5.20	Final BPMN Prototype, create/update a user task (own Figure)	117
5.21	Final BPMN Prototype, edit service task (own Figure)	118
5.22	Process execution (own Figure)	119
5.23	Representation of additional information during execution (own Figure)	120
5.24	Integration of Desk UI (own Figure)	121
6.1	Study procedure (own Figure)	124
6.2	Landing page of https://ilovecobots.at (own Screenshot)	124
6.3	Introduction/Voluntariness Anonymity/Data Protection (own Screenshot)	124
6.4	Video tutorial for the BPMN prototype (own Figure)	125
6.5	Study procedure part 3: Solving the task (own Figure)	126
6.6	Correct solution "Assembly Heat Sink" (own Figure)	126
6.7	Visualisation of the robot tasks by means of video sequences (own Figure)	127
6.8	Online Questionnaire (Demographic data) (own Screenshot)	128
6.9	Online Questionnaire (Evaluation of the prototype) (own Screenshot)	128
6.10	Gender (n=51) (own Figure)	132
6.11	Occupation (n=51) (own Figure)	132
6.12	Age (n=51) (own Figure)	132

6.13 Experience with collaborative robots (n=51) (own Figure)	133
6.14 Programming experience (n=51) (own Figure)	133
6.15 Experience with process modelling (n=51) (own Figure)	134
6.16 Experience with BPMN (n=51) (own Figure)	134
6.17 Correct Solution (n=51) (own Figure)	135
6.18 Duration for solving the task (own Figure)	135
6.19 SUS-Score BPMN Prototype (own Figure)	136
6.20 Nasa-TLX (own Figure)	136
6.21 Tablet usage (n=50) (own Figure)	137
7.1 Extended 3-Layer Architecture (own Figure)	140
7.2 Final GUI of the BPMN prototype (own Figure)	141
7.3 Execution of the collaborative process (own Figure)	141

Bibliography

- [1] Linda Onnasch, Xenia Maier, and Thomas Jürgensohn. “Mensch-Roboter-Interaktion - Eine Taxonomie für alle Anwendungsfälle”. In: *Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA) F 2369* (2016), pp. 1–12. DOI: [10.21934/baua:fokus20160630](https://doi.org/10.21934/baua:fokus20160630).
- [2] Wilhelm Bauer et al. *Leichtbauroboter in der manuellen Montage – einfach EINFACH anfangen. Erste Erfahrungen von Anwenderunternehmen*. 2016.
- [3] J. Edward Colgate, Witaya Wannasuphoprasit, and Michael A. Peshkin. “Cobots: robots for collaboration with human operators”. In: *American Society of Mechanical Engineers, Dynamic Systems and Control Division (Publication) DSC*. Vol. 58. ASME, 1996, pp. 433–439.
- [4] IFR International Federation of. *World Robotics Report: Global Sales of Robots Hit \$16.5B in 2018*. <https://www.roboticsbusinessreview.com/research/world-robotics-report-global-sales-of-robots-hit-16-5b-in-2018/> (visited on 02/08/2020).
- [5] TM Robotics. *The Global Robotics Report*. 2019. <http://news.tmrobotics.com/wp-content/uploads/2019/03/TMR006-Global-Robotics-Report-WP.pdf> (visited on 02/12/2020).
- [6] Franka Emika GmbH. *FRANKA EMIKA*. <https://franka.de/> (visited on 02/12/2020).
- [7] C. Schmidbauer et al. “Adaptive task sharing in human-robot interaction in assembly”. In: *IEEE International Conference on Industrial Engineering and Engineering Management 2020-Decem* (2020), pp. 546–550. ISSN: 2157362X. DOI: [10.1109/IEEM45057.2020.9309971](https://doi.org/10.1109/IEEM45057.2020.9309971).
- [8] *The-Desk-programming-interface-of-the-Franka-Emika-Panda-robot.png (850×467)*. https://www.researchgate.net/profile/Tudor%7B%5C_%7DIonescu/publication/331115083/figure/fig1/AS:726541268955139@1550232345954/

[The-Desk-programming-interface-of-the-Franka-Emika-Panda-robot.png](#) (visited on 06/12/2020).

- [9] International Organisation for Standardisation (ISO). *Ergonomics of human-system interaction - Part 11: Usability: Definitions and Concepts*. German ver. Beuth Verlag, Berlin, 2018, p. 2. ISBN: 924111:2018.
- [10] Michael Richter and Markus Flückiger. *User-Centred Engineering: Creating Products for Humans*. Springer Publishing Company, Incorporated, 2014.
- [11] Deborah J Mayhew. “The Usability Engineering Lifecycle”. In: *CHI '99 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 147–148. ISBN: 1581131585. DOI: [10.1145/632716.632805](https://doi.org/10.1145/632716.632805).
- [12] Alan Cooper, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design (Llibre electrònic de Google)*. 2007, pp. 929–934. ISBN: 1118079159.
- [13] Hugh Beyer and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. ISBN: 9780080503042.
- [14] International Organisation for Standardisation (ISO). *ISO 9241-210: Ergonomics of human-system interaction - Human-centred design for interactive systems*. Beuth Verlag, Berlin, 2010, p. 32. ISBN: 0-580-64009-4.
- [15] Kitchenham BA and Stuart Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.
- [16] *IEEE Xplore*. <https://ieeexplore.ieee.org/Xplore/home.jsp> (visited on 08/06/2020).
- [17] *ScienceDirect.com | Science, health and medical journals, full text articles and books*. <https://www.sciencedirect.com/> (visited on 08/06/2020).
- [18] *Home - Springer*. <https://link.springer.com/> (visited on 08/06/2020).
- [19] *Google Scholar*. <https://scholar.google.com/> (visited on 08/07/2020).
- [20] *CiteSeerX*. <https://citeseerx.ist.psu.edu/index> (visited on 08/07/2020).
- [21] *ACM Digital Library*. <https://dl.acm.org/> (visited on 08/07/2020).

- [22] *SAGE Journals: Your gateway to world-class research journals.* <https://journals.sagepub.com/> (visited on 08/06/2020).
- [23] Andreas Butz and Antonio Krüger. *Mensch-Maschine-Interaktion*. 2nd ed. De Gruyter Studium. Berlin: De Gruyter Oldenbourg, 2017. ISBN: 978-3-11-047636-1. DOI: [10.1524/9783110476378](https://doi.org/10.1524/9783110476378).
- [24] Joanneum Research. *Cobots Meets Makerspace.* <https://www.joanneum.at/en/robotics/reference-projects/comemak/> (visited on 02/12/2020).
- [25] Christine Henseling, Tobias Hahn, and Katrin Nolting. *Die Fokusgruppen-Methode als Instrument in der Umwelt- und Nachhaltigkeitsforschung*. 43. 2006. ISBN: 3929173433.
- [26] Marlen Schulz, Birgit Mack, and Ortwin Renn. *Fokusgruppen in der empirischen Sozialwissenschaft*. 2012. DOI: [10.1007/978-3-531-19397-7](https://doi.org/10.1007/978-3-531-19397-7).
- [27] Tiago Silva Da Silva et al. *User-centered design and agile methods: A systematic review*. 2011, pp. 77–86. ISBN: 9780769543703. DOI: [10.1109/AGILE.2011.24](https://doi.org/10.1109/AGILE.2011.24).
- [28] Lu Luo. *Software testing techniques: technology maturation and research strategy*. 2001.
- [29] John Brooke. *"SUS-A quick and dirty usability scale."* *Usability evaluation in industry*. 1996.
- [30] Sandra G. Hart. *NASA Task Load Index (TLX) v. 1.0 Paper and Pencil Package*. 1986.
- [31] Matthias Haun. *Handbuch Robotik*. 2013. ISBN: 9783642398575. DOI: [10.1007/978-3-642-39858-2](https://doi.org/10.1007/978-3-642-39858-2).
- [32] Karel Capek, Paul Selver, and Nigel Playfair. *R. U. R. (Rossum's Universal Robots)*. 1923.
- [33] *Robotik • Definition | Gabler Wirtschaftslexikon.* <https://wirtschaftslexikon.gabler.de/definition/robotik-54198> (visited on 09/15/2020).
- [34] D J Todd. *Fundamentals of Robot Technology*. 24th. USA: Halsted Press, 1986. ISBN: 0470203013.
- [35] *Duden | Roboter | Rechtschreibung, Bedeutung, Definition, Herkunft.* <https://www.duden.de/rechtschreibung/Roboter> (visited on 09/21/2020).

- [36] Matthias Haun. *Handbuch Robotik*. 2007. ISBN: 9788578110796. DOI: [10.1017/CBO9781107415324.004](https://doi.org/10.1017/CBO9781107415324.004). arXiv: [arXiv:1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- [37] *Unimate - The First Industrial Robot*. <https://www.robotics.org/joseph-engelberger/unimate.cfm> (visited on 09/23/2020).
- [38] International Organisation for Standardisation (ISO). “DIN EN ISO 10218-1”. In: April (2020).
- [39] *VDI 2860 - Definition Industrial Robot*. <https://learnchannel-tv.com/robot/definition-roboter/> (visited on 09/21/2020).
- [40] VDI. “VDI 2860:1990-05, Montage- und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole”. In: (1990).
- [41] *VDI 2860 - Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbol | VDI*. <https://www.vdi.de/richtlinien/details/vdi-2860-handhabungsfunktionen-handhabungseinrichtungen-begriffe-definitionen-symbol> (visited on 09/21/2020).
- [42] *Image Industrial Robots*. <https://www.produktion.de/assets/images/8/fertigung-5881c228.jpg> (visited on 09/23/2020).
- [43] *Image Audi A4 Production Line*. <https://automationspraxis.industrie.de/wp-content/uploads/I/n/Industrierobotik.jpg> (visited on 09/23/2020).
- [44] Andreas Pott and Thomas Dietz. *Industrielle Robotersysteme*. 2019. ISBN: 9783658253448. DOI: [10.1007/978-3-658-25345-5](https://doi.org/10.1007/978-3-658-25345-5).
- [45] International Federation of Robotics. *World Robotics 2020 Sales Flyer*. Tech. rep. IFR. https://ifr.org/img/office/World%7B%5C_%7DRobotics%7B%5C_%7D2020%7B%5C_%7DSales%7B%5C_%7DFlyer.pdf.
- [46] International Federation of Robotics. *Executive Summary World Robotics 2020 Industrial Robots*. https://ifr.org/img/worldrobotics/Executive%7B%5C_%7DSummary%7B%5C_%7DWR%7B%5C_%7D2020%7B%5C_%7DIndustrial%7B%5C_%7DRobots%7B%5C_%7D1.pdf (visited on 01/29/2021).
- [47] *Trends im Bereich Robotik und Automation Kollaborateure im Mittelstand*. <https://industrieanzeiger.industrie.de/allgemein/kollaborateure-im-mittelstand/> (visited on 09/28/2020).

- [48] Mikkel KNUDSEN and Jari KAIVO-OJA. “Collaborative Robots: Frontiers of Current Literature”. In: *Journal of Intelligent Systems: Theory and Applications* June (2020), pp. 13–20. DOI: [10.38016/jista.682479](https://doi.org/10.38016/jista.682479).
- [49] ALES VYSOCKY and PETR NOVAK. “HUMAN – ROBOT COLLABORATION IN INDUSTRY”. In: *MM Science Journal* (2016), pp. 903–906. <https://pdfs.semanticscholar.org/4005/d30d36fffe36ca9c15e3dad70d75363efd16.pdf>.
- [50] DIN Deutsches Institut für Normung e. V. “Roboter und Robotikgeräte – Kollaborierende Roboter (ISO/TS 15066:2016)”. In: *BEST BeuthStandardsCollection* April (2017), pp. 2017–2024. www.din.de.
- [51] Shirine El Zaatari et al. “Cobot programming for collaborative industrial tasks: An overview”. In: *Robotics and Autonomous Systems* 116 (2019), pp. 162–180. DOI: [10.1016/j.robot.2019.03.003](https://doi.org/10.1016/j.robot.2019.03.003).
- [52] *IFR presents World Robotics Report 2020 - International Federation of Robotics*. <https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe> (visited on 01/29/2021).
- [53] *Preisunterschiede bei Cobots – MRK-Blog.de*. <https://mrk-blog.de/preisunterschiede-bei-cobots/> (visited on 09/30/2020).
- [54] Fabian Ranz et al. “A Morphology of Human Robot Collaboration Systems for Industrial Assembly”. In: *Procedia CIRP* 72 (2018), pp. 99–104. ISSN: 22128271. DOI: [10.1016/j.procir.2018.03.011](https://doi.org/10.1016/j.procir.2018.03.011).
- [55] Baptiste Menges, Michaël Sarrey, and Patrick Henaff. “Integration of a collaborative robot in a hard steel industrial environment”. In: *IEEE International Conference on Automation Science and Engineering 2018-Augus* (2018), pp. 634–637. ISSN: 21618089. DOI: [10.1109/COASE.2018.8560583](https://doi.org/10.1109/COASE.2018.8560583).
- [56] Tudor B. Ionescu and Sebastian Schlund. “A participatory programming model for democratizing cobot technology in public and industrial fablabs”. In: *Procedia CIRP* 81 (2019), pp. 93–98. ISSN: 22128271. DOI: [10.1016/j.procir.2019.03.017](https://doi.org/10.1016/j.procir.2019.03.017).
- [57] Christina Schmidbauer, Titanilla Komenda, and Sebastian Schlund. “Teaching cobots in learning factories - User and usability-driven implications”. In: *Procedia Manufacturing* 45 (2020), pp. 398–404. ISSN: 23519789. DOI: [10.1016/j.promfg.2020.04.043](https://doi.org/10.1016/j.promfg.2020.04.043).

- [68] G Sinha, R Shahi, and M Shankar. *Human Computer Interaction*. Dec. 2010, pp. 1–4. DOI: [10.1109/ICETET.2010.85](https://doi.org/10.1109/ICETET.2010.85).
- [69] Alan Dix. “Human-Computer Interaction”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1327–1331. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9_192](https://doi.org/10.1007/978-0-387-39940-9_192).
- [70] “Human-Computer Interaction (HCI) and User Interfaces”. In: *Encyclopedia of Biometrics*. Ed. by Stan Z Li and Anil Jain. Boston, MA: Springer US, 2009, p. 713. ISBN: 978-0-387-73003-5. DOI: [10.1007/978-0-387-73003-5_445](https://doi.org/10.1007/978-0-387-73003-5_445).
- [71] Christopher Reid Becker. *Learn Human-Computer Interaction: Solve Human Problems and Focus on Rapid Prototyping and Validating Solutions Through User Testing*. Birmingham: Packt Publishing, Limited, 2020. ISBN: 1838820329.
- [72] International Organisation for Standardisation (ISO). *ISO 9241-210: Ergonomics of human-system interaction - Interaction principles*. September. Beuth Verlag, Berlin, 2019, p. 83.
- [73] B Shneiderman. “Direct Manipulation: A Step Beyond Programming Languages”. In: *Computer* 16.08 (1983), pp. 57–69. ISSN: 1558-0814 VO - 16. DOI: [10.1109/MC.1983.1654471](https://doi.org/10.1109/MC.1983.1654471). <http://doi.ieeecomputersociety.org/10.1109/MC.1983.1654471>.
- [74] Andreas Butz and Antonio Krüger. *Mensch-Maschine-Interaktion*. München: De Gruyter Oldenbourg, 2014. ISBN: 348671967X. [10.1524/9783486719673](https://doi.org/10.1524/9783486719673).
- [75] Jörn Hurtienne and L Blessing. “Metaphors as tools for intuitive interaction with technology”. In: *Metaphorik*. de June (2007), pp. 21–52.
- [76] Anja Naumann et al. “Intuitive use of user interfaces: Defining a vague concept”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 4562 LNAI. May 2014 (2007), pp. 128–136. ISSN: 16113349. DOI: [10.1007/978-3-540-73331-7_14](https://doi.org/10.1007/978-3-540-73331-7_14).
- [77] Jörn Hurtienne et al. “Intuitive Use of User Interfaces - Definition und Herausforderungen”. In: *i-com* 5 (Dec. 2006), pp. 41–68. DOI: [10.1524/icom.2006.5.3.38](https://doi.org/10.1524/icom.2006.5.3.38).
- [78] Wilbert O Galitz. *The Essential Guide to User Interface Design : An Introduction to GUI Design Principles and Techniques*. 3rd ed.. Chichester: Wiley, 2007. ISBN: 1280855339.

- [79] Jeff Johnson. *Designing With the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann, 2020. ISBN: 9780128182024.
- [80] Karen H. Frith. *User Experience Design*. Vol. 40. 1. 2019, pp. 65–66. ISBN: 9783642133626. DOI: [10.1097/01.nep.0000000000000451](https://doi.org/10.1097/01.nep.0000000000000451).
- [81] International Organisation for Standardisation (ISO). *Qualitätsmanagementsysteme – Grundlagen und Begriffe (ISO 9000:2015); Deutsche und Englische Fassung EN ISO 9000:2015*. November. Beuth Verlag, Berlin, 2015. ISBN: 7255520001201.
- [82] Peter Mertens, Joachim Griese, and Marco Meier. *Integrierte Informationsverarbeitung : 1. Operative Systeme in der Industrie*. 17., übera. Lehrbuch. Wiesbaden: Gabler, 2009. ISBN: 3834916455.
- [83] Thomas H Davenport. *Process innovation : reengineering work through information technology*. 12. [print. Boston, Mass: Harvard Business School Pr., 1997. ISBN: 0875843662.
- [84] Andreas Gadatsch. *Grundkurs Geschäftsprozess-Management*. 2017. ISBN: 9783658171780. DOI: [10.1007/978-3-658-17179-7](https://doi.org/10.1007/978-3-658-17179-7).
- [85] Andreas Gadatsch. *Gadatsch_Grundkurs GPM*. 2019, p. 204. ISBN: 9783658171780. DOI: [10.1007/978-3-658-17179-7](https://doi.org/10.1007/978-3-658-17179-7).
- [86] OMG. *Business Process Model and Notation (BPMN) Version 2.0*. 2011.
- [87] *Welcome To UML Web Site!* <http://uml.org/> (visited on 10/03/2020).
- [88] *BPMN Introduction and History*. <https://www.trisotech.com/blog/bpmn-introduction-and-history> (visited on 10/03/2020).
- [89] *Cover Pages: Business Process Modeling Language (BPML)*. <http://xml.coverpages.org/bpml.html> (visited on 01/27/2021).
- [90] Stephen White. “Introduction to BPMN”. In: (2004).
- [91] *About the Business Process Model and Notation Specification Version 2.0.2*. <https://www.omg.org/spec/BPMN> (visited on 10/03/2020).
- [92] *OMG | Object Management Group*. <https://www.omg.org/> (visited on 10/03/2020).

- [93] *BPMN-Basics: Pools und Lanes verstehen* / Signavio. <https://www.signavio.com/de/post/bpmn-pools-und-lanes-verstehen/> (visited on 10/03/2020).
- [94] Thomas Allweyer. *BPMN 2.0*. BoD, 2020. ISBN: 9783750435261.
- [95] Alexander Lübke and Sven Schnägelberger. “BPM Toolmarktmonitor 2015”. In: April (2015), pp. 1–9.
- [96] *BPMN Specification - Business Process Model and Notation*. <http://www.bpmn.org/> (visited on 10/07/2020).
- [97] *Web-based tooling for BPMN, DMN and CMMN* / bpmn.io. <https://bpmn.io/> (visited on 10/07/2020).
- [98] Brad Myers, Scott E. Hudson, and Randy Pausch. “Past, Present, and Future of User Interface Software Tools”. In: *ACM Transactions on Computer-Human Interaction* 7.1 (2000), pp. 3–28. ISSN: 15577325. DOI: [10.1145/344949.344959](https://doi.org/10.1145/344949.344959).
- [99] *Cover Pages: Business Process Modeling Language (BPML)*. <http://xml.coverpages.org/bpml.html> (visited on 10/03/2020).
- [100] *Workflow and Decision Automation Platform* / Camunda. <https://camunda.com/> (visited on 10/07/2020).
- [101] *BPMN 2.0 hat sich durchgesetzt - BPMN 2.0: Was sind die aktuellen Trends im BPM?* - TecChannel Workshop. <https://www.tecchannel.de/a/was-sind-die-aktuellen-trends-im-bpm,2041148,2> (visited on 10/07/2020).
- [102] Zengxi Pan et al. “Recent progress on programming methods for industrial robots”. In: *Robotics and Computer-Integrated Manufacturing* 28.2 (2012), pp. 87–94. ISSN: 07365845. DOI: [10.1016/j.rcim.2011.08.004](https://doi.org/10.1016/j.rcim.2011.08.004).
- [103] M H Ang, L Wei, and Lim Ser Yong. “An industrial application of control of dynamic behavior of robots-a walk-through programmed welding robot”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 3. Apr. 2000, 2352–2357 vol.3. DOI: [10.1109/ROBOT.2000.846378](https://doi.org/10.1109/ROBOT.2000.846378).
- [104] *Example Online-Programming*. https://www.robologic.eu/wp-content/uploads/2018/07/onlineprogrammierung%7B%5C_%7Dwellphoto%7B%5C_%7D83449602%7B%5C_%7Dadobestock.jpg (visited on 10/08/2020).

- [105] *Example Offline-Programming.* <https://www.steuerungstechnik.com/de/wp-content/uploads/sites/2/2016/09/Screenshot-RoboGuide.jpg> (visited on 10/08/2020).
- [106] Geoffrey Biggs and Bruce Macdonald. “A Survey of Robot Programming Systems”. In: *Proceedings of the Australasian conference on robotics and automation* January (2003), pp. 1–3. ISSN: 09574158.
- [107] M Tykal, Alberto Montebelli, and V Kyrki. “Incrementally assisted kinesthetic teaching for programming by demonstration”. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2016), pp. 205–212.
- [108] *LEGO® MINDSTORMS® – Über | Offizieller LEGO® Shop AT.* <https://www.lego.com/de-at/themes/mindstorms/about> (visited on 10/08/2020).
- [109] *Example Lego Mindstorms.* <https://i.pinimg.com/originals/61/8f/a4/618fa4607a1611c8dfedc6c70c4f4483.png> (visited on 10/08/2020).
- [110] Kerstin Fischer et al. “A comparison of types of robot control for programming by demonstration”. In: *ACM/IEEE International Conference on Human-Robot Interaction 2016-April* (2016), pp. 213–220. ISSN: 21672148. DOI: [10.1109/HRI.2016.7451754](https://doi.org/10.1109/HRI.2016.7451754).
- [111] *Universal Robots teaching pendant.* <https://www.therobotreport.com/wp-content/uploads/2017/08/Universal-Robots-Academy-teach-pendant-1.png> (visited on 10/13/2020).
- [112] Rainer Bischoff, Arif Kazi, and Markus Seyfarth. *The MORPHA style guide for icon-based programming.* Feb. 2002, pp. 482–487. ISBN: 0-7803-7545-9. DOI: [10.1109/ROMAN.2002.1045668](https://doi.org/10.1109/ROMAN.2002.1045668).
- [113] *Advanced man-machine interfaces for robot system applications | AMIRA Project | FP4 | CORDIS | European Commission.* https://cordis.europa.eu/project/id/FP4%7B%5C_%7D22646/de (visited on 10/14/2020).
- [114] *ISO - ISO 15187:2000 - Manipulating industrial robots — Graphical user interfaces for programming and operation of robots (GUI-R).* <https://www.iso.org/standard/26697.html> (visited on 10/14/2020).
- [115] Gregory Rossano et al. *Easy robot programming concepts: An industrial perspective.* Aug. 2013, pp. 1–6. DOI: [10.1109/ISR.2013.6695710](https://doi.org/10.1109/ISR.2013.6695710).

- [116] *CAD • Definition | Gabler Wirtschaftslexikon.* <https://wirtschaftslexikon.gabler.de/definition/cad-29408> (visited on 10/14/2020).
- [117] *Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics.* <https://www.coppeliarobotics.com/> (visited on 10/14/2020).
- [118] *RobotStudio - ABB Robotics.* <https://new.abb.com/products/robotics/de/robotstudio> (visited on 10/14/2020).
- [119] *Robot Programming Software for CAM Manufacturing | Autodesk.* <https://www.autodesk.com/solutions/robot-programming-software> (visited on 10/14/2020).
- [120] *ABB RobotStudio GUI.* https://img.directindustry.de/images%7B%5C_%7Ddi/photo-g/30265-13147232.jpg (visited on 10/14/2020).
- [121] *CoppeliaSim GUI.* <https://images3.programmersought.com/638/37/37e330708025871340a9221b68afd6e6.png> (visited on 10/14/2020).
- [122] C Schou et al. “Human-robot interface for instructing industrial tasks using kinesthetic teaching”. In: *IEEE ISR 2013*. 2013, pp. 1–6. DOI: [10.1109/ISR.2013.6695599](https://doi.org/10.1109/ISR.2013.6695599).
- [123] Franz Steinmetz, Annika Wollschl, and Roman Weitschat. “RAZER — A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization”. In: 3.3 (2018), pp. 1362–1369.
- [124] Rene Lindorfer, Roman Froschauer, and Georg Schwarz. “ADAPT - A decision-model-based Approach for Modeling Collaborative Assembly and Manufacturing Tasks”. In: *Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018* (2018), pp. 559–564. DOI: [10.1109/INDIN.2018.8472064](https://doi.org/10.1109/INDIN.2018.8472064).
- [125] Josef Wolfartsberger et al. “Industrial perspectives on assistive systems for manual assembly tasks”. In: *ACM International Conference Proceeding Series* (2018), pp. 289–291. DOI: [10.1145/3197768.3201552](https://doi.org/10.1145/3197768.3201552).
- [126] *Erste Sprachassistenten für die Industrie.* <https://www.theagilityeffect.com/de/article/in-der-industrie-bisher-kaum-verbreitet/> (visited on 09/10/2020).
- [127] *Siri - Apple (DE).* <https://www.apple.com/de/siri/> (visited on 09/10/2020).

- [128] *Amazon Alexa*. <https://alexa.amazon.de/> (visited on 09/10/2020).
- [129] T Pentikäinen and S Richard. “How to make collaborative robot programming easier: Workflow visualization on a tablet device”. In: June (2016). <http://www.diva-portal.org/smash/record.jsf?pid=diva2:955637>.
- [130] Maj Stenmark, Mathias Haage, and Elin Anna Topp. “Simplified Programming of Re-usable Skills on a Safe Industrial Robot: Prototype and Evaluation”. In: *ACM/IEEE International Conference on Human-Robot Interaction Part F1271* (2017), pp. 463–472. ISSN: 21672148. DOI: [10.1145/2909824.3020227](https://doi.org/10.1145/2909824.3020227).
- [131] Ryo Hanai et al. “Design of robot programming software for the systematic reuse of teaching data including environment model”. In: *ROBOMECH Journal* 5.1 (2018). ISSN: 21974225. DOI: [10.1186/s40648-018-0120-z](https://doi.org/10.1186/s40648-018-0120-z).
- [132] *Choreonoid Official Site — Choreonoid Official Site*. <https://choreonoid.org/en/> (visited on 09/10/2020).
- [133] *Blockly | Google Developers*. <https://developers.google.com/blockly> (visited on 09/02/2020).
- [134] Jurgen Blume. “IProgram: Intuitive programming of an industrial HRI cell”. In: *ACM/IEEE International Conference on Human-Robot Interaction* (2013), pp. 85–86. ISSN: 21672148. DOI: [10.1109/HRI.2013.6483513](https://doi.org/10.1109/HRI.2013.6483513).
- [135] Federica Ferraguti et al. “A Methodology for Comparative Analysis of Collaborative Robots for Industry 4.0”. In: *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019* (2019), pp. 1070–1075. DOI: [10.23919/DATE.2019.8714830](https://doi.org/10.23919/DATE.2019.8714830).
- [136] Juergen Mangler et al. “Centurio.work - Industry 4.0 Integration Assessment and Evolution”. In: *CEUR Workshop Proceedings* 2428.ii (2019), pp. 106–117. ISSN: 16130073.
- [137] DIN Deutsches Institut für Normung. *DIN SPEC 91345 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)*. April. 2016, pp. 1–40.
- [138] Dominik Schonberger, Rene Lindorfer, and Roman Froschauer. “Modeling Workflows for Industrial Robots Considering Human-Robot-Collaboration”. In: *Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018* (2018), pp. 400–405. DOI: [10.1109/INDIN.2018.8471999](https://doi.org/10.1109/INDIN.2018.8471999).

- [139] Harold Bright Maynard, G J Stegemerten, and John L Schwab. *Methods - time measurement*. MacGraw-Hill industrial organization and management series. McGraw-Hill, 1948. ISBN: 125835098X.
- [140] Andreas Syska. *Produktionsmanagement: Das A - Z wichtiger Methoden und Konzepte für die Produktion von heute*. Springer-11775 [Dig. Serial]. Wiesbaden: Betriebswirtschaftlicher Verlag Dr. Th. Gabler | GWV Fachverlage GmbH Wiesbaden, 2006. ISBN: 978-3-8349-0235-1. <http://dx.doi.org/10.1007/978-3-8349-9091-4>.
- [141] Marcos Roberto et al. “Methods time measurement on the optimization of a productive process: A case study”. In: *2017 4th International Conference on Control, Decision and Information Technologies, CoDIT 2017* 2017-Janua (2017), pp. 980–985. DOI: [10.1109/CoDIT.2017.8102726](https://doi.org/10.1109/CoDIT.2017.8102726).
- [142] Richard P Paul and Shimon Y Nof. “Work methods measurement—a comparison between robot and human task performance”. In: *International Journal of Production Research* 17.3 (1979), pp. 277–303. DOI: [10.1080/00207547908919615](https://doi.org/10.1080/00207547908919615).
- [143] Rene Lindorfer, Roman Froschauer, and Georg Schwarz. “ADAPT - A decision-model-based Approach for Modeling Collaborative Assembly and Manufacturing Tasks”. In: *Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018* (2018), pp. 559–564. DOI: [10.1109/INDIN.2018.8472064](https://doi.org/10.1109/INDIN.2018.8472064).
- [144] Marco Brambilla. *Model-driven software engineering in practice*. Ed. by Jordi [VerfasserIn] Cabot and Manuel [VerfasserIn] Wimmer. Second edi. Synthesis lectures on software engineering ISBN 9781627059886. [San Rafael, Calif.]: [San Rafael, Calif.] : Morgan & Claypool Publishers, 2017, 1 Online-Ressource (209 Seiten), Illustrationen. ISBN: 1627059881. <https://permalink.catalogplus.tuwien.at/AC15045195>.
- [145] *Judo*. <https://www.judo.codes/blog/mona-lisa-model-of-all-models> (visited on 09/02/2020).
- [146] Universal Robots A/S. *Universal Robots*. <https://www.universal-robots.com/de/> (visited on 02/12/2020).
- [147] *Universal Robots weiterhin Marktführer bei Cobots - ROBOTIK UND PRODUKTION*. <https://www.robotik-produktion.de/allgemein/universal-robots-weiterhin-marktfuehrer-bei-cobots/> (visited on 10/14/2020).

- [148] *Universal Robots Online-Schulung.* <https://academy.universal-robots.com/de/online-schulung/e-series-online-schulung/> (visited on 10/14/2020).
- [149] Kuka AG. *Kuka.* <https://www.kuka.com/> (visited on 02/12/2020).
- [150] *Stand der Mensch-Roboter-Kollaboration: MRK – noch in der Lernphase | Markt&Technik.* <https://www.elektroniknet.de/markt-technik/automation/mrk-noch-in-der-lernphase-179599.html> (visited on 10/15/2020).
- [151] *Kuka Iisy.* <https://www.blog.kuka.com/2019/01/31/lbr-iisy/> (visited on 10/15/2020).
- [152] *Kuka Programming Interface.* <https://www.erm-automatismes.com/il000206-fr-robot-agilus-smartpad.jpg> (visited on 09/10/2020).
- [153] *Kuka Iisy Programming Interface.* https://www.youtube.com/watch?time%7B%5C_%7Dcontinue=2%7B%5C%7Dv=61H2V8ZwxSI (visited on 09/10/2020).
- [154] *FANUC Industrieroboter für intelligente Automation.* <https://www.fanuc.eu/at/de/roboter> (visited on 10/15/2020).
- [155] *FANUC Programming Interface.* <https://www.youtube.com/watch?v=uWgYofd5EJ0> (visited on 09/10/2020).
- [156] *Fanuc CRX programming.* <https://www.youtube.com/watch?v=-Irgy-nKPNA> (visited on 10/15/2020).
- [157] *Industrieroboter HORST - Fruitcore Robotics.* <https://fruitcore-robotics.com/> (visited on 09/10/2020).
- [158] *horstCOSMOS by fruitcore robotics.* <https://horstcosmos.com/> (visited on 10/15/2020).
- [159] *Intelligent Cobots for a World of Applications | Techman Robot.* <https://www.tm-robot.com/en/> (visited on 10/16/2020).
- [160] Omron. *Software Manual TMflow Original Instruction Software Version: 1.68.* Tech. rep. https://assets.omron.eu/downloads/manual/en/v1/tm%7B%5C_%7Dflow%7B%5C_%7Dsoftware%7B%5C_%7Dmanual%7B%5C_%7Dinstallation%7B%5C_%7Dmanual%7B%5C_%7Den.pdf

- [161] *ABB Group. Leading digital technologies for industry.* <https://global.abb/group/en> (visited on 10/16/2020).
- [162] *YuMi - Kollaborativer Roboter von ABB - Überblick ABB-Industrieroboter.* <https://new.abb.com/products/robotics/de/industrieroboter/yumi> (visited on 10/16/2020).
- [163] *ABB - YuMi Programming Interface.* <https://www.cobotrends.com/block-based-programming-yumi-cobot-installations/> (visited on 09/10/2020).
- [164] *Webinar - ABB Wizard easy programming for single arm YuMi - YouTube.* <https://www.youtube.com/watch?v=OKlcUcLMHQM%7B%5C%7Dfeature=youtu.be> (visited on 10/16/2020).
- [165] *Drag&Bot Interface.* <https://www.dragandbot.com/de/produkt/core/> (visited on 09/10/2020).
- [166] *ROS.org | Powering the world's robots.* <https://www.ros.org/> (visited on 10/16/2020).
- [167] *ROS-Industrial.* <https://rosindustrial.org/> (visited on 10/17/2020).
- [168] *Robot Operationg System (ROS) Interface.* <https://www.theconstructsim.com/wp-content/uploads/2017/11/ROS-QA-Subscribe-custom-message-attributes-screen.png> (visited on 09/10/2020).
- [169] Brajesh De and Rajesh Doda. *API Management*. eng. 1st ed. Berkeley, CA: Apress L. P, 2017. ISBN: 1484213068. DOI: [10.1007/978-1-4842-1305-6](https://doi.org/10.1007/978-1-4842-1305-6).
- [170] *Process Engine API | docs.camunda.org.* <https://docs.camunda.org/manual/7.14/user-guide/process-engine/process-engine-api/> (visited on 11/11/2020).
- [171] *Rest Api Reference | docs.camunda.org.* <https://docs.camunda.org/manual/latest/reference/rest/> (visited on 11/11/2020).
- [172] Roy Thomas Fielding and Richard N Taylor. "Architectural Styles and the Design of Network-Based Software Architectures". PhD thesis. 2000. ISBN: 0599871180.
- [173] David Gourley and Brian Totty. *HTTP : the definitive guide*. First edit. Sebastopol, California: O'Reilly, 2002. ISBN: 1449379583.

- [174] *Delete Deployment* | docs.camunda.org. <https://docs.camunda.org/manual/latest/reference/rest/deployment/delete-deployment/> (visited on 11/12/2020).
- [175] *Post Deployment* | docs.camunda.org. <https://docs.camunda.org/manual/7.6/reference/rest/deployment/post-deployment/> (visited on 11/18/2020).
- [176] *API Client for REST, SOAP, & GraphQL Queries* | Postman. <https://www.postman.com/product/api-client/> (visited on 03/01/2021).
- [177] *Start Process Instance* | docs.camunda.org. <https://docs.camunda.org/manual/7.6/reference/rest/process-definition/post-start-process-instance/> (visited on 11/18/2020).
- [178] *Get Tasks* | docs.camunda.org. <https://docs.camunda.org/manual/7.8/reference/rest/task/get-query/> (visited on 11/18/2020).
- [179] *Get External Tasks* | docs.camunda.org. <https://docs.camunda.org/manual/7.6/reference/rest/external-task/get-query/> (visited on 11/18/2020).
- [180] *Complete Task* | docs.camunda.org. <https://docs.camunda.org/manual/7.6/reference/rest/task/post-complete/> (visited on 11/18/2020).
- [181] *Franka Control Interface Documentation — Franka Control Interface (FCI) documentation.* <https://frankaemika.github.io/docs/> (visited on 11/18/2020).
- [182] *DVK/DVK.py at master · ib101/DVK · GitHub.* <https://github.com/ib101/DVK/blob/master/Code/DVK.py> (visited on 11/18/2020).
- [183] *GitHub - berndhader/BPMN-Extension-Franka-Emika-Desk: A BPMN extension for Franka Emika Desk to create collaborative human-robot-processes.* <https://github.com/berndhader/BPMN-Extension-Franka-Emika-Desk> (visited on 11/25/2020).
- [184] *External Tasks* | docs.camunda.org. <https://docs.camunda.org/manual/7.7/user-guide/process-engine/external-tasks/> (visited on 11/25/2020).
- [185] *External Task Client* | docs.camunda.org. <https://docs.camunda.org/manual/7.14/user-guide/ext-client/> (visited on 11/25/2020).

- [186] *GitHub - camunda/camunda-external-task-client-js: Implement your BPMN Service Task in NodeJS.* <https://github.com/camunda/camunda-external-task-client-js> (visited on 11/25/2020).
- [187] Wang Jiacun. *Formal Methods in Computer Science*. New York: Chapman and Hall/CRC, 2019.
- [188] *javascript-state-machine - npm.* <https://www.npmjs.com/package/javascript-state-machine> (visited on 11/26/2020).
- [189] *bpmn-js: BPMN 2.0 rendering toolkit and web modeler / bpmn.io.* <https://bpmn.io/toolkit/bpmn-js/> (visited on 12/02/2020).
- [190] *Browserify.* <http://browserify.org/> (visited on 12/02/2020).
- [191] *GitHub - bpmn-io/bpmn-js-properties-panel: A properties panel for bpmn-js.* <https://github.com/bpmn-io/bpmn-js-properties-panel> (visited on 12/02/2020).
- [192] *Quill - Your powerful rich text editor.* <https://quilljs.com/> (visited on 12/02/2020).
- [193] Sandra G. Hart. "NASA-task load index (NASA-TLX); 20 years later". In: *Proceedings of the Human Factors and Ergonomics Society* (2006), pp. 904–908. ISSN: 10711813. DOI: [10.1177/154193120605000909](https://doi.org/10.1177/154193120605000909).
- [194] *Laravel - The PHP Framework For Web Artisans.* <https://laravel.com/> (visited on 01/19/2021).
- [195] *PHP Sessions.* https://www.w3schools.com/php/php%7B%5C_%7Dsessions.asp (visited on 01/19/2021).
- [196] Aaron Bangor et al. "Determining what individual SUS scores mean: adding an adjective rating scale". In: *Journal of usability studies* 4.3 (2009), pp. 114–123. ISSN: 1931-3357.