Institute of Software Engineering,
Institute for Control Engineering of Machine Tools and Manufacturing Units

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Generation of Reinforcement Learning Environments from Machine-Tool Descriptions

Viktor Krimstein

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. rer. nat. Stefan Wagner
Supervisors:	Dr. rer. nat. Justus Bogner, Dr.-Ing. Akos Csiszar
Commenced:	December 1st, 2020
Completed:	June 1st, 2021

Abstract

Due to the ever-increasing amount of available data, the technological advances for its processing, and in the context of Industry 4.0, research and industry are focusing on creating increasingly detailed digital twins. These aspire to transfer all the capabilities and attributes of their physical counterparts into the digital world. Digital Twins enable simulations of real production and manufacturing processes to be carried out, new approaches to be tested and, in turn, innovative conclusions to be drawn without having to take the risks that costly machines entail. In parallel, approaches from the fields of machine learning, artificial intelligence and reinforcement learning are finding continuously more applications in the manufacturing and robotics domains. Especially in the latter, OpenAI researchers achieved a breakthrough, namely the construction of a neural network that was trained to solve a Rubik's cube by a robotic hand using reinforcement learning. For the implementation, appropriate simulation environments were used, in which the agent responsible for controlling the robotic arm could train and learn for an enormous amount of times in the simulation.

However, the highly heterogeneous environment in the production environment makes it difficult to integrate reinforcement learning methodologies and create the necessary simulations. Researchers must spend a severe amount of their time implementing interfaces for specific machine-tool related components rather than working on the actual problem. It is exactly this issue that this thesis addresses.

The goal of this master thesis is the empirical development of a methodology for the automatic generation of reinforcement learning simulation environments for machine-tools. Within the scope of the thesis, different requirements shall be collected by interviewing domain experts as potential end users, generalized and transferred into a software concept. Furthermore, the possibility of deducing and abstracting state and action spaces for reinforcement learning environments and agents from a given machine-tool description is to be investigated within the scope of this work. In addition, the concept to be developed should be machine-tool- and platform-agnostic, as well as modular, so that subsequent research can be conducted upon the presented concept.

Kurzfassung

Sowohl durch die ständig zunehmende Anzahl an verfügbaren Daten, die technologischen Fortschritte für ihre Verarbeitung, als auch im Kontext der Industrie 4.0 fokussieren sich Forschung und Industrie auf die Erstellung immer detailgetreuerer digitaler Zwillinge. Diese bemühen sich darum, alle Fähigkeiten und Attribute ihrer physikalischen Gegenstücke in die digitale Welt zu übertragen. Sie ermöglichen es, Simulationen von realen Produktions- und Fertigungsprozessen durchzuführen, neue Ansätze zu erproben um so wiederum innovative Schlüsse ziehen zu können ohne dabei die Risiken eingehen zu müssen, welche kostspielige Maschinen mit sich bringen. Parallel hierzu finden Ansätze aus den Bereichen des maschinellen Lernens, künstlicher Intelligenz sowie dem Reinforcement Learning vermehrt Anwendung im Fertigungs- und Robotiksektor. Besonders im letzten Bereich gelang den Forschern von OpenAI ein Durchbruch, nämlich die Konstruktion eines neuronalen Netzwerks, welches mittels Reinforcement Learning darauf trainiert wurde eine Roboterhand einen Zauberwürfel zu lösen. Für die Umsetzung fanden entsprechende Simulationsumgebungen ihren Ansatz, in denen der Agent, welcher für die Steuerung des Roboters verantwortlich war, unendliche Male in der Simulation trainieren und Lernen konnte.

Jedoch erschwert die höchst heterogene Umgebung im Produktionsumfeld die Integration von Reinforcement und die Erstellung der notwendigen Simulationen. Forscher müssen einen Großteil ihrer Zeit damit zubringen, Schnittstellen für spezifische Komponenten zu implementieren, anstatt an dem tatsächlichen Problem zu arbeiten. Genau an dieser Problematik soll diese Thesis ansetzen.

Ziel dieser Masterarbeit ist die empirische Erarbeitung einer Methodik für die automatische Generierung von Reinforcement Learning Simulationsumgebungen für Produktionsmaschinen. Im Rahmen der Arbeit sollen unterschiedliche Anforderungen durch die Befragung von Domänenexperten als potenzielle Endanwender erhoben, generalisiert und in ein Softwarekonzept überführt werden. Ferner soll im Rahme dieser Arbeit die Möglichkeit untersucht werden, inwiefern sich Zustands- und Aktionsräume für Reinforcement Learning Umgebungen und Agenten aus einer gegebenen Maschinenbeschreibung deduzieren und abstrahieren lassen. Zusätzlich soll das zu erstellende Konzept soll maschinen- und plattformagnostisch, sowie modular gestaltet werden, damit anschließende Arbeiten auf den vorgestellten Konzepten aufbauen können.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AI Artificial Intelligence. 15, 22, 25, 27

API Application Programming Interface. 19

CNC Computer Numerical Control. 16

CPS Cyber-Physical Systems. 7, 15, 17

DT Digital Twin. 7, 15, 17, 18, 25, 27, 35

ICT Information and Communication Technologies. 15

IoT Internet of Things. 15

MDP Markov Decision Process. 22

ML Machine Learning. 15

NC Numerical Control. 18

OPC Open Platform Communications. 18

OPC UA Open Platform Communications Unified Architecture. 18

PLC Programmable Logic Controller. 18

RL Reinforcement Learning. 15, 16, 17, 22, 23, 24, 25, 27, 29, 31, 33, 35, 38, 39, 47, 48

RQ Research Question. 16

UML Unified Modeling Language. 18

1 Introduction

The industrial world is facing a digital transformation that started in Germany, in 2013, with the *Industrie 4.0* initiative [NB18; PCB+19]. This transformation, known as the fourth industrial revolution, is based on the use of Cyber-Physical Systems (CPS) and Information and Communication Technologies (ICT) in manufacturing systems. Significant advances in Machine Learning (ML), Artificial Intelligence (AI) and the Internet of Things (IoT) are also contributing [MLC20]. Based on these circumstances, current research focuses on the creation and optimization of Digital Twin (DT) [WSJ17]. Due to the increasing interconnection of CPS, continuously growing data and information stocks enable the increasingly accurate modeling of physical production systems [Woh19].

DTs enable the simulation and calculation of results from production processes as well as experimentation with different optimization approaches without having to access cost-intensive real machines [Can16]. In order to use the DTs efficiently, they are equipped with the same or very similar interfaces as real machines. The goal is to be able to generate a simulation of real production and manufacturing processes that is as accurate as possible in order to analyze and verify new production methods in a time- and cost-optimized manner.

In parallel, the use of ML and AI in the production environment is also increasing. The aspiration here is the possible optimization of existing processes and increase efficiency [CEV17]. Further, the generation of new insights and verification of innovative methods poses an additional research goal [JCKV18]. Reinforcement Learning (RL) algorithms can be used to train agents to simulate specific production processes and develop appropriate manufacturing strategies [LZYW20].

1.1 Motivation

Currently, the incompatibility of AI methods with those of the production and manufacturing domains poses significant challenges for research [NNXR08]. The creation of simulation environments and DTs requires a high degree of domain- and problem-specific knowledge. Furthermore, the large number of process and machine description models, some of which are incompatible, makes the generalization and modularization of simulation techniques difficult [ESLR19]. For example, if an agent trained by RL is to simulate the control of a sorting machine in order to evaluate the selected process strategy, it requires a specific description of the action and state spaces of the machine-tool used, as well as a set of input and output signals annotated with the corresponding domain context. The description and implementation of such simulation environments present researchers with an enormous effort. Depending on the type of the machine-tool used, the communication and description format as well as the target task, researchers have to describe and simulate these details in a highly heterogeneous domain.

1.2 Research Questions and Goals

Targeting the above-mentioned issues, the goal of this thesis is the empirical analysis and development of a methodology for the automatic generation of RL simulation environments for machine-tools. The goal is to simplify the implementation of simulation environments in a manner which is generalizable through various frameworks and communication protocol types. Following an empirical research approach, this work provides a concept for a software solution for automatic generation of RL environments and the according

agents for a given machine-tool description. Use-Cases and functional details of the resulting software architecture are derived from semi-structured interviews lead with domain experts from both, research and industry. The additional goal is to homogenize the methodologies with which the simulation environments are created, while at the same time leaving enough space for researchers to resort to the tools that are most appropriate for their work.

Based on the presented introduction and motivation, this thesis strives to answer the following Research Questions (RQ):

RQ1: What general inconveniences in the process of generating a RL environment and agent with the corresponding simulation have to be overcome to enhance researcher productivity and the overall process quality?

RQ2: Given a potential software solutions for automatic RL environment and agent generation. What Use-Cases and technical functionalities doe potential users expect from such a software?

RQ3: Given a potential software solutions for automatic RL environment and agent generation. What would be the application of such a solution in the daily research work of the users?

1.3 Outline

The remainder of this work is divided into the following Chapters:

Chapter 2 - Foundations: Here, necessary foundational knowledge for RL simulation environments and machine-tool description and communication types is provided.

Chapter 3 - Related Work: Related research papers for RL environment generation and applications to CNCs are highlighted and differences to the presented thesis are discussed.

Chapter 4 - Research Approach: This chapter presents the empirical research approach and strategies used in this work.

Chapter 5 - Evaluation and Results: Research results are presented and discussed in this chapter alongside the key aspects leading to the creation of the concept.

Chapter 6 - Concept: Based on the results of the empirical research the concept, for a methodology of RL environment generator is presented.

Chapter 7 - Conclusion and Outlook: Finally, this chapter provides a summary of the work and gives conclusive remarks, along with an outlook to future work.

2 Foundations

This chapter establishes the foundations necessary to follow the remainder of this work. The basic concept of Digital Twins are introduced. Later a general understanding of the components of a machine-tool is provided alongside the communication- and capability-description methods. Eventually, an introduction to RL is given together with specific concepts required for the generation of simulation environments.

2.1 Digital Twin

A Digital twin can be defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision-making [RSK20]. DTs implement a broader concept that refers to a virtual representation of manufacturing elements such as personnel, products, assets and process definitions, a living model that continuously updates and changes as the physical counterpart changes to represent status, working conditions, product geometries and resource states in a synchronous manner [LLK+20]. Further, DTs provide a more sophisticated method for testing and experimenting with manufacturing processes in a virtual environment excluding risks of misconfiguration on a physical machine-tool. Figure 2.1 puts the DTs and their corresponding physical devices in context with the global term of CPS.

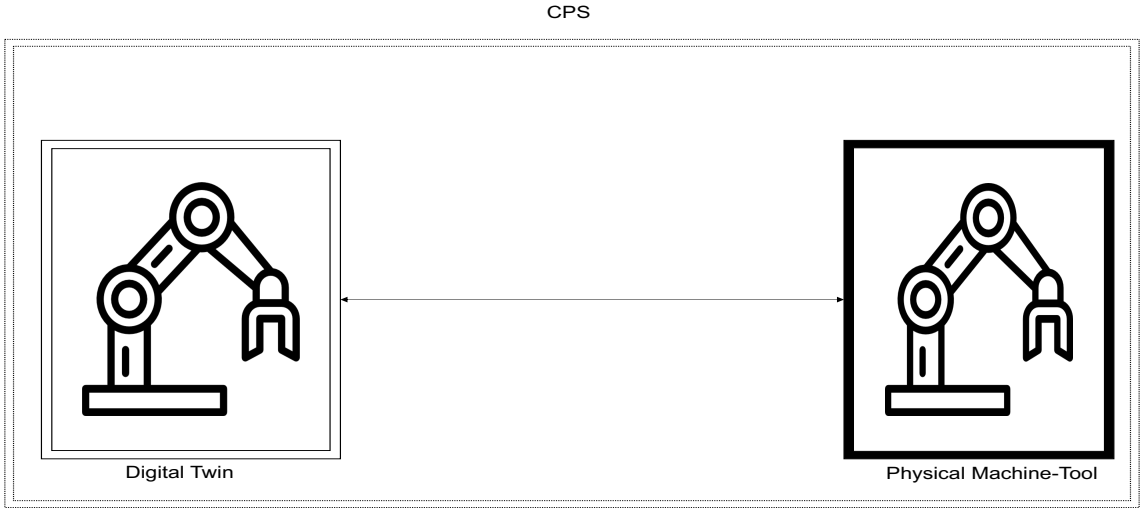

Figure 2.1: The relationship between a DT and a CPS (adapted from [BL15]).

The DT represents a high-fidelity representation of the physical device. It allows not only to model and simulate processes possibly executed on the physical device, but mirrors nearly in real-time the signals of its physical counterpart. Specifically, this means that a CPS is characterized by both properties - a physical machine and its Digital Twin.

Kritzinger et al. strive to differentiate recent applications of DTs based on the degree of data in digital and physical objects [KKT+18]. In their work, the authors separate two terms: the *Digital Model* and a *Digital Twin* [KKT+18]. A Digital Model implements bi-directional manual data flow, while the DT on the other

hand enables bi-directional data flow automatically. Digital Shadow only feeds one-way automatic data flow from the physical object into the digital object, while from digital object to physical object manually. Based on this categorical method, the majority of publications investigated by the authors were classified as Digital Shadow and Digital Model [KKT+18]. Studies where the Digital Twin classification could be applied correctly, are rare [XSK+21]. This shows the inconclusive specification of the term of DTs. Nevertheless, by speaking about DTs in this thesis, the term is defined by the definition of Kritzinger et al.

2.2 Machine-Tool Description and Communication

For the context of this work the machine-tool representation is abstracted to the following parts: the physical machine-tool itself (e.g. a machine for milling metal), the Programmable Logic Controller (PLC) which controls the motions of the physical tool parts and electrical signals of the machine-tool and the Numerical Control (NC) which is responsible for numerical calculation and preparation of the paths to be followed by the machine-tool. Figure 2.2 provides a highly abstracted Unified Modeling Language (UML) [Obj17] component diagram representation of the machine-tool.

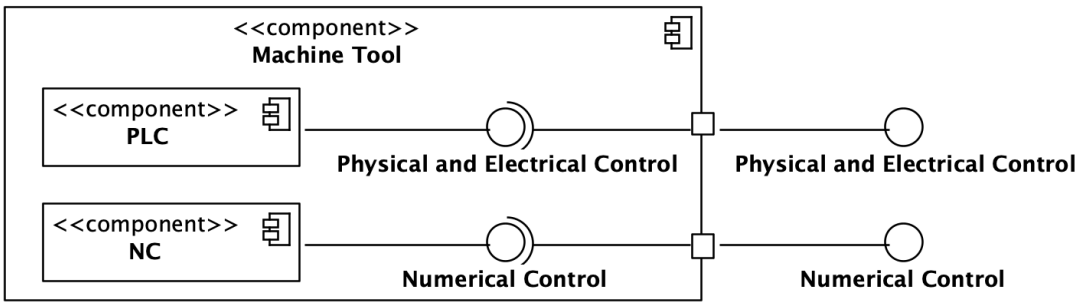

Figure 2.2: UML component diagram of the two main components of a machine-tool.

The machine-tool component provides two external interfaces to access the PLC and NC functionalities. The PLC of the machine-tool is responsible for the physical and electrical control of the manufacturing and machine tool processes. It can execute control commands like data gathering from the machine tool state, start- and stop of the production and transfer data to the NC. Processing of production parts data, planing of the milling routes for example and executing the part production is handled by the NC. Production data is loaded from and to the NC via the PLC and the current state is transferred to the PLC in real-time ensuring a correct monitoring of every production process.

In the following all physical interactions are encapsulated within the machine-tool component, since one is not interested how exactly the control realizes a specific manufacturing process (e.g. setting the rotation speed of a drill) but rather, that it is possible by sending an explicit signal to the control of the machine-tool.

2.2.1 Open Platform Communications Unified Architecture

The Open Platform Communications Unified Architecture (OPC UA) is the new version of the well-known Open Platform Communications (OPC) architecture [Had06] originally designed by the OPC Foundation to connect industrial devices to control and supervision applications [HS14]. The focus of OPC is on getting access to large amounts of real-time data while ensuring performance constraints without disrupting the normal operation of the devices [CJOC10]. The original OPC specifications, based on Microsoft's

Component Object Model/Distributed Component Object Model (COM/DCOM), are becoming obsolete and are gradually being replaced by new interoperability standards, including web-services what led the OPC Foundation to publish a new architecture, called OPC UA [Had06].

Note: The following two figures, shown in Figure 2.3 and Figure 2.4 were cited with the kind permission of the OPC Foundation to cite this images by referencing the origin OPC UA Specification Part referenced in [OPC17a].

OPC UA Server Model: OPC UA specifies the exchange of real-time information of production plant data between control devices or Information Technology (IT) systems from different manufacturers [VOX+05]. The communication is established by an inverted client-server system, where the client triggers actions on the server for automation control, and the server executes the commands on, or retrieves data from the underlying machine [IJ13].

Figure 2.3 shows the OPC UA Server Model according to its specification in [OPC17a]. OPC UA servers include an information model that allows users to organize data and their semantics in a structured manner.

This semantic *AddressSpace* is constructed of standalone or interconnected Nodes mapped to real CPS object representatives as shown in Figure 2.3. Furthermore, nodes can be divided into functionality and view nodes, each sort implementing different functionalities and manners of user interaction. Further, each node can be monitored and subscribed by parties of interest using the OPC UA Server Application Programming Interface (API) as presented at the bottom of Figure 2.3. The information model constitutes the *AddressSpaces* of OPC UA servers. It is a fullmesh network of nodes with their properties and relations.

In general, users create the information model for their OPC UA servers manually at implementation time or implement vendor-specific automatisms [HS14].

By providing specific monitorable elements and an asynchronous communication, OPC UA can be used to interact only with specific functionalities of the machine-tool while underlying concepts are hidden from the clients. It should be noted at this point, that the specification does not enforce a specific semantic mapping between nodes and their functionalities. Vendors are free to implement the *AddressSpace* in their own fashion, leaving it up to the domain experts and users to map the nodes to their semantic equivalents.

OPC UA Client Model: The OPC UA Client architecture models the Client endpoint of client/server interactions. Figure 2.4 illustrates the major elements of a typical Client and how they relate to each other. As presented in Figure 2.4, the OPC UA Client is constructed by two layers – the Client Application and the OPC UA Client API. The Client Application encapsulates the producer–consumer service functionality by accessing the underlying OPC UA Client-API following the asynchronous system designs described in [TV07]. Further, the Client Application is the code that implements the function of the Client. It uses the Client API to send and receive OPC UA Service requests and responses to the Server. The Services defined for OPC UA are described in Clause 6.4 of [OPC17b]. Note that the Client API is an internal interface that isolates the Client application code from an OPC UA Communication Stack.

The OPC UA Communication Stack converts Client API calls into messages and sends them through the underlying communications entity to the Server at the request of the Client application. The OPC UA Communication Stack also receives response and *NotificationMessages* from the underlying communications entity and delivers them to the Client application through the Client API.

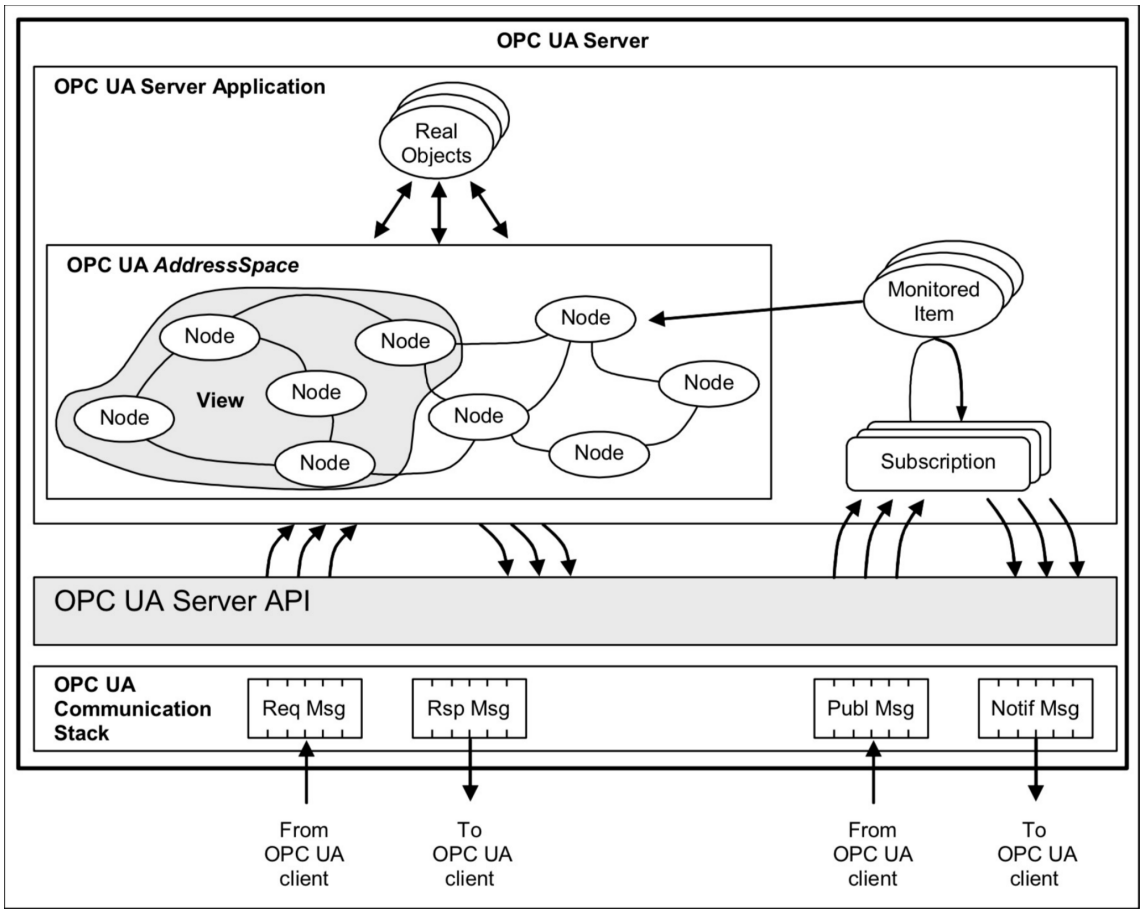

Figure 2.3: The OPC UA server architecture (adapted from [OPC17a]).

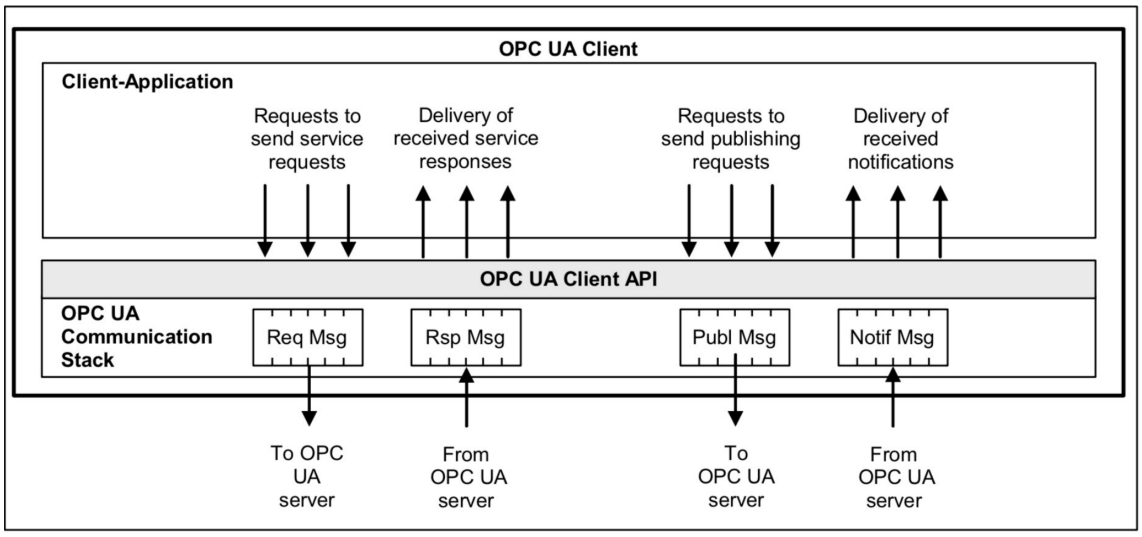

Figure 2.4: The OPC UA client architecture (adapted from [OPC17a]).

2.2.2 EtherCAT

EtherCAT is an ISO standardized Industrial Ethernet technology specialized in real-time communication using field bus systems [RSD10]. The main difference between the standard Ethernet protocol and EtherCAT is, that EtherCAT allows the processing of Data Frames *on the fly*. This differentiation reduces the costs of latency in usual Ethernet communication. The main target of EtherCAT is to ensure a standard communication profile in the industrial context. Emphasize is put on the flexibility of the protocol and an efficient communication with different top-layer protocols (like OPC UA) rather than a semantic and modular description of the machine-tool capabilities [WZLL21]. Figure 2.5 shows a schematic overview of the EtherCAT Master implementation.

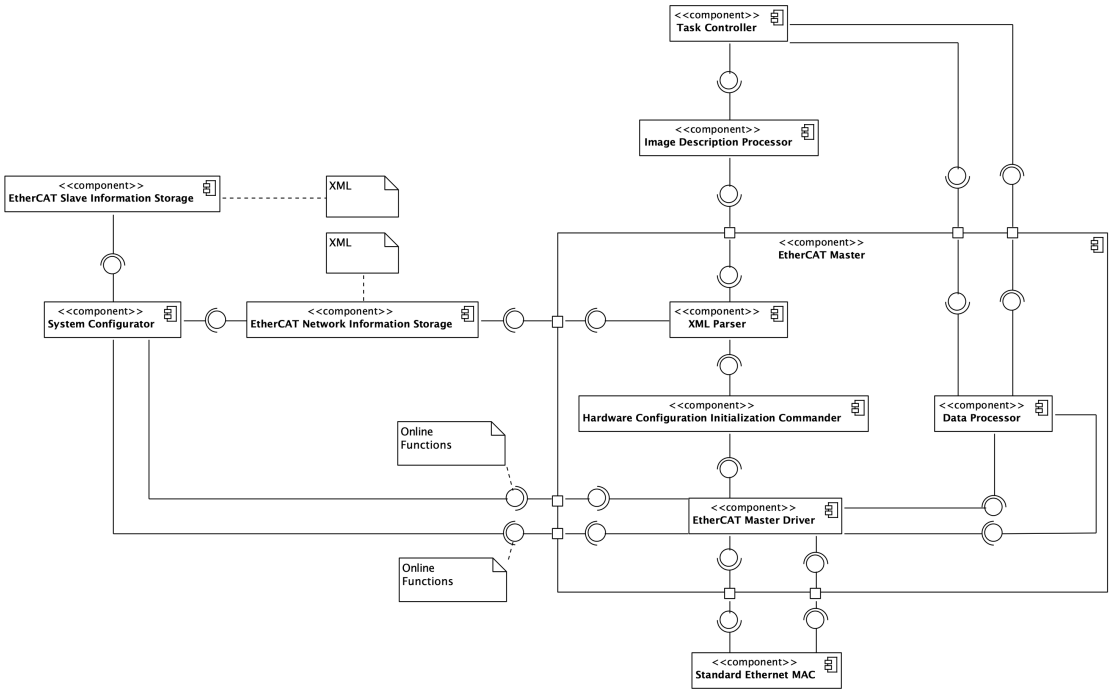

Figure 2.5: Schematic UML component diagram showing the EtherCAT Master implementation.

The scope of the available master implementation and the supported functions varies depending on the target application. Optional functionalities are supported or intentionally omitted to optimize hardware and software resources. One of the differences between EtherCAT and OPC UA is that EtherCAT is rather a pure communication protocol than OPC UA. EtherCAT does not decide about the modeling and functionality, how data sent is represented.

2.3 Reinforcement Learning

The term *Reinforcement Learning* describes a class of methods where one or multiple agents are placed in an environment for interactive task solving. The research field is influenced by AI, especially robotics, and classical control theory. In the following, this thesis follows the formalism from the textbooks by Sutton et al. [SB18], Russell et al. [RN02] and Goodfellow et al. [GBCB16]. First, we introduce the terminology of Reinforcement Learning. Figure 2.6 describes the general idea of RL.

Formalism An reinforcement learning agents' actions in a given environment, can be formalized by a Markov Decision Process (MDP) [RN02]. An MDP is a 5-tuple, $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \sigma_0 \rangle$, where

- $\mathcal{S}$ is the set of all valid states,
- $\mathcal{A}$ is the set of all valid actions,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, with $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$,
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition probability function, with $\mathcal{P}(s' \mid s, a)$ being the probability of transitioning into state s' if the agent starts in state s and takes action a,
- σ_0 is the starting state distribution.

The name MDP refers to the fact that the system obeys the Markov property, which can be described as the property that transitions only depend on the most recent state and action, and no prior historical data influences the decision.

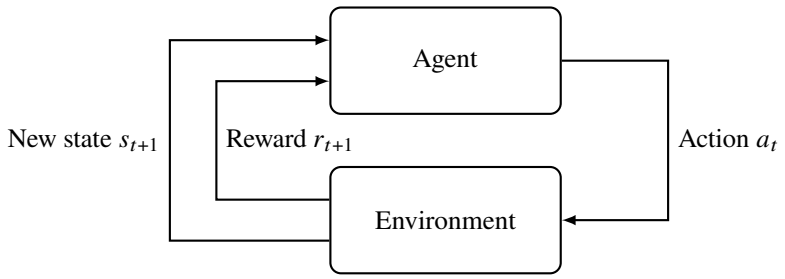

Figure 2.6: Generic Reinforcement Learning Agent-Environment-loop (adapted from [SB18]).

Environment and Agent The environment describes the space that a learning agent is placed in to solve tasks. While the environment can be either simulated or real, the agent is usually following a strategy that computationally defines how to explore and act in the environment. The goal of an agent in an environment is to learn to solve the specified task in the environment. During the learning phase, it follows a search strategy to explore and learn a strategy to exploit the experienced knowledge in order to solve the task.

In the context of simulating machine-tools, the agent performs actions by sending in- and output signals to the control unit of the machine-tool. By sending the control signals, the control unit lets the machine perform a certain task (e.g. moving a robotic arm forward). Hence, the agent performed an action in the environment, which is the machine-tool itself.

States and Observations A state $s \in \mathcal{S}$ describes an observable part of the environment that the agent is in. That description might not necessarily be complete, as the environment might only be partially observable. An observation o is a partial description of a state, which may omit information. For example, a robot tasked with navigating in a room might only have access to a camera image representing the room and the robots position. Terminal states denote a goal state, after which the environment usually experiences some kind of reset to an initial state. A terminal state also marks the end of an episode. The State space $\mathcal{S}$ describes the space of all possible states.

Actions When in a state $s \in \mathcal{S}$, an agent can perform an action $a \in \mathcal{A}$ from an action space $\mathcal{A}$. This results in a new state $s' \in \mathcal{S}$ and a reward signal. From then on, the agent is again tasked with performing an action, given the state s'. This loop is also illustrated in Figure 2.6. An agents interaction in an episode with the environment gives rise to a sequence

$$\tau = ((s_0, a_0), (s_1, a_1), \ldots)$$

which is also referred to as a trajectory. For example, a robot tasked with navigating from point A to B might get a small negative reward of -1 for every part of the trajectory it takes towards point B, but a huge positive reward of 100 for actually stepping into B, and thus reaching a terminal state that finishes the trajectory. The rewards therefore encourage the robot to reach B with as few, deliberately actions as possible, taking the direct route, as to minimize the potential negative reward that accumulates from taking detours.

Policies A policy π is a rule used by an agent to decide what actions to take. It can be either deterministic, in which case it is usually denoted by μ:

$$a_t = \mu(s_t),$$

or stochastic, in which case it is usually denoted by π:

$$a_t \sim \pi(\cdot|s_t),$$

Especially in the robotics (and therefore the manufacturing) domain, RL deals with parametrized policies. [AAC+19]. These policies are described by computable functions that depend on a set of parameters which are adjusted to change the behavior via some optimization algorithm. Such policies can be formalized as

$$a_t = \mu_\theta(s_t)$$
$$a_t \sim \pi_\theta(\cdot|s_t)$$

where θ is a set which encapsulates all necessary parameters.

Reward and Return The reward function $\mathcal{R}$ is critically important in RL. It depends on the current state of the world, the action just taken, and the next state of the world:

$$r_t = \mathcal{R}\left(s_t, a_t, s_{t+1}\right)$$

Put simple, the reward is the gratification for the agent which he gets when transitioning from his current state to the next state by taking a specific action. The agent literally makes a *step*. The goal of the agent is to maximize some notion of cumulative reward over a trajectory. One kind of return is the finite-horizon undiscounted return, which is just the sum of rewards obtained in a fixed window of steps:

$$\mathcal{R}(\tau) = \sum_{t=0}^{T} r_t.$$

Another kind of return is the infinite-horizon discounted return, which is the sum of all rewards ever obtained by the agent, but discounted by how far off in the future they're obtained. This formulation of reward includes a discount factor $\gamma \in (0, 1)$ which is usually set to $\frac{1}{2}$ in practice:

$$\mathcal{R}(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

The discount factor is added to enhance the chances, that an infinite-horizon sum of rewards may converge to a finite value. Convergence is the key factor in both reward function formulations.

Reinforcement Learning Problem Definition The main goal of every RL algorithm is to find an optimal policy to maximize the expected return if the agent acts according to it in the environment. The expected return and therefore the utility $\mathcal{U}$ of the chosen policy can be expressed as:

$$\begin{aligned}\mathcal{U}(\pi) &= \int_{\tau} \mathcal{P}(\tau \mid \pi)\, \mathcal{R}(\tau) \\ &= \mathbb{E}_{\tau \sim \pi}\left[\mathcal{R}(\tau)\right]\end{aligned}$$

The central optimization problem in RL can then be expressed by

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathcal{U}(\pi),$$

with π^* being the optimal policy.

2.4 Reinforcement Learning Environment Frameworks

In the near past the application of RL algorithms has led to ground-breaking results. Whether Google Deepminds AlphaZero [SHS+18] which has, as the first system ever, beaten the world-champion in Go (the Chinese version of chess) or OpenAIs robotic hand which was able to solve single-handedly a Rubik's Cube [AAC+19]. In the manufacturing domain, research was conducted on the creation of a RL agent, which can deduce the correct control policy of a manufacturing plant by acting in its DT (the Hardware-in-the-Loop simulation) [JCKV18].

All approaches have in common, that researchers have to design and implement the simulation and visualization, the abstraction layer for the RL environment and the RL agent. Depending on the posed optimization problem, this can be a cumbersome task [KCV18]. Although it is still required to write a certain amount of *boilerplate* source code, several Open Source projects provide standard approach for implementing RL agents and environments as well as training and gathering results of the algorithms and policies.

OpenAI Gym The OpenAI Gym framework is toolkit for RL research including the common interfaces, environment and simulation implementations alongside with benchmarks and reference implementations of basic algorithms [BCP+16]. OpenAI Gym includes several implementations of RL environments including the domains of Atari Console Games, classic control (like the CartPole problem) and MuJoCo [TET12]. The framework is implemented in Python and provides interfaces for the most common AI frameworks, including TensorFlow and PyTorch.

Google Dopamine Google's Dopamine is an Open Source framework for researches to prototype and experiment with different RL algorithms [CMG+18]. Although Dopamine is not as popular in the developer community[1], it is under continuous development and provides just as OpenAi's Gym environments and integration for simulation and research of agents in environments, like Atari Console Games and MuJuCo simulations.

[1] On the GitHub Platform around 9.400 developers starred the project while on the other hand over 20.000 starred the OpenAI Gym project.

3 Related Work

Chapter 2 was concerned about providing necessary background information and a general overview required for this work. Since the research and goals in the manufacturing and AI domains diverge it is challenging to find commonalities. Therefore, the following presents an overview of research topics in the intersection of both domains and how this work contributes to the research fields.

3.1 Digital Twin Generation

Digital Twins have been proven as an excellent tool to improve systems design, to improve systems exploitation efficiency, as well as a maintenance support tool [CLQS19]. Research is highly driven to create even more precise and configurable simulations starting from the earliest stages of the system development [YJ18]. However, highly specialized domain expertise is required at every step of the creation of simulations the integration into a physical plant [Kre14]. Therefore, one research focus is the automatic generation of Digital Twin simulations from given data [MSKV18]. The main focus of the conducted research is the generation of physical twins and their corresponding capabilities in terms of control and functionality. It terms of the application of AI or RL, researchers have to implement and integrate the agents themselves and create simulation environments by their own.

Tools like MatLab[1] provide mechanisms for simulation and training of RL agents and the creation of machine-tool simulation, but the integration and methodology is neither standardized nor straight-forward. It takes a certain amount of *boilerplate code* to generate a digital twin by its physical properties and even more to integrate all components together including many manual tasks which slow down the actual research [MOW+21; XSK+21].

As highlighted by Xia et al. in their paper, the design of the RL environment was particularly focused on the integration to a previously created DT [XSK+21]. Results were shown for a Proof-of-Concept of a specific tasks rather than a general approach of integrating the RL algorithm design process in a reusable manner.

3.2 Reinforcement Learning Environment Generation

Since the design of RL environments and agents is very task specific, it poses a very challenging task to find papers, which aim to generate such environments automatically. Although Ha et al. presented in their paper an approach to automate the reconfiguration of a robotic arm, their results and research questions are not comparable to those of this thesis. [HKY18].

[1] See the official MathWorks Website: https://de.mathworks.com/products/reinforcement-learning.html?s_tid=hp_brand_rl

3.3 Summary

This thesis tackles the challenge to provide a concept for generation of reinforcement learning environments and corresponding agents from given machine-tool descriptions and domain expert input. It may seem similar to the research field of Digital Twin generation, but this paper focuses on the Reinforcement Learning and generalization aspects, rather than on specific physical and electrical properties of systems. Thus, this thesis makes a symbiotic contribution to the current research.

4 Research Approach

After Chapter 2 and Chapter 3 have prepared the basics and a connection to related literature has been established, the following chapter deals with the research strategy with which the three initially mentioned research questions in Chapter 1 are to be answered.

4.1 Research Methodology

The research methodology for this thesis is organized into the following phases: (i) Literature Review and Decision of the programming language and RL framework to be used for the creation of the concept, (ii) the experience-based evaluation where semi-structured interviews with domain experts were conducted to provide a qualitative analysis and (iii) the conclusion and deduction of Use-Cases and requirements. Table 4.1 describes the implied reasons and included tasks for each phase.

Table 4.1: Distribution of research phases.

Research Phase	Task	Reason for Selection
Preparation	Literature Review Analysis of existing open source RL frameworks	gaining domain insights and foundational knowledge of existing approaches, minimal implementation to understand possible inconveniences of domain experts
Experience-based Evaluation	semi-structured interviews with domain experts	Qualitative Analysis, gaining insights from experts, external input from the end-users to design a concept which will fit their needs
Result Aggregation	aggregation of interview results identification of possible Use-Cases and requirements	Analysis of similarities between participants, combination of ideas to determine a clear understanding of the concept

The research of this thesis is focused around the interviews with the domain experts. Through the analysis and definition of Use-Cases directly from potential users the goal is to design a software architecture concept which can fit exactly to the user's requirements. Further, the results of the literature review in the first research phase have already been presented in Chapter 2 and Chapter 3. For the qualitative analysis of the frameworks and tools used for the concept creation, research was conducted about current implementations of RL algorithms and environments.

Research Strategy Both the creation of RL environments and agents, as well as possible ways of integration with machine-tools, have been researched thoroughly. It should be noted that it showed as a challenging task to find literature on general approaches and common best practices for simulation creation. Instead, the literature presents specific solutions and their implementations leaving it up to the recipients to draw parallels and generalize the methodologies to their own needs. Several research strategies exist and were used for this analysis. An exploratory form of research strategy is relevant and hence exercised as shown in Figure 4.1.

In the presented research process, several related topics were explored which provided valuable insights into the domain and the issues motivating further exploration. The motivation and a formal hypothesis for future research work was deduced.

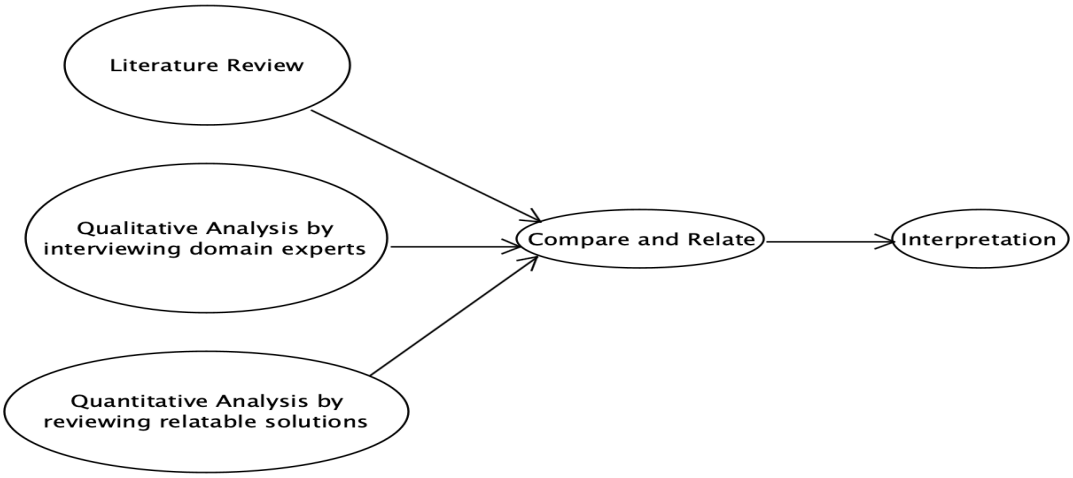

Figure 4.1: Schematic representation of an exploratory research strategy.

Research Design This research is designed in a flexible since not all the parameters and the required domain knowledge were available in advance. During the course of the research parameters and dependencies could be more and more clarified and through the literature research specified. On the other hand, a fixed research design would require a predefined set of elements before data collection can be initiated. Figure 4.2 shows the continuous interdependent workflow.

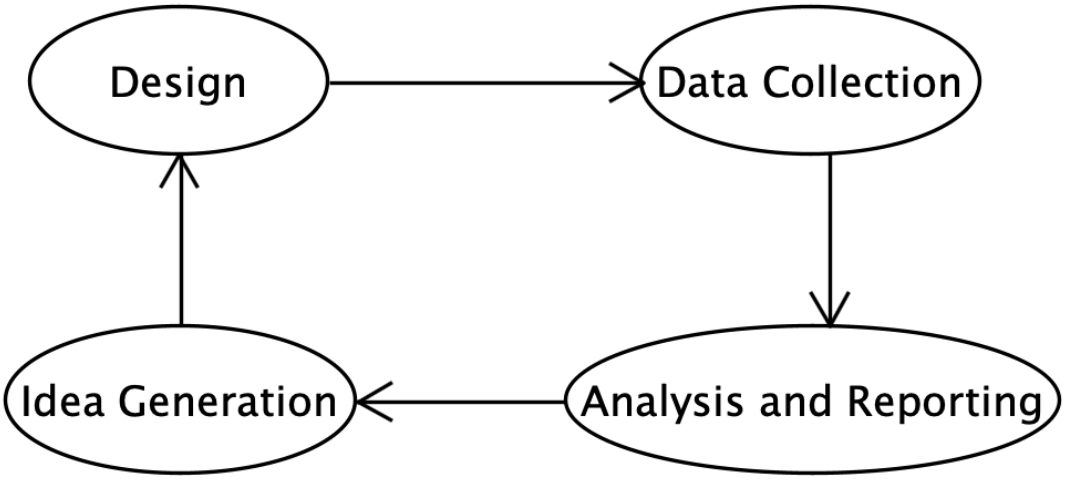

Figure 4.2: The research design of flexible type (adapted from [RTVG21]).

Through continuous data and information aggregation new insights could be unveiled and lead to more specific research design. Using this iterative approach, gathered knowledge from literature and interviews could be integrated in the concept creation process.

Research Category This case study is a deductive approach where we start with the collection of hypotheses from existing theories, and then we test these hypotheses against our observations.

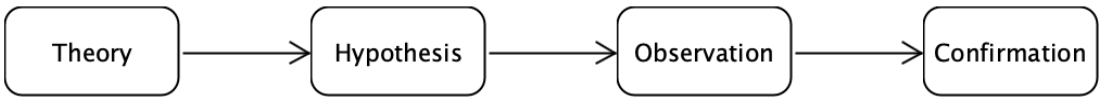

Figure 4.3: Schematic overview of a deductive research approach (adapted from [FP19]).

4.2 Interview Design

The semi-structured interviews in a 45-minute time frame with the domain experts. Format of the interviews were video- and audio calls. First, the participants were asked to provide brief information about themselves like their professional and demographic background, total years of professional employment and the total number of years in the domain of manufacturing and RL research.

RQ1: Targeting the first research question of this thesis mentioned in Chapter 1 the participants were asked about the general tools, programming languages and frameworks they use on a regular basis. The participants were asked open formulated question, so they have the according freedom to speak in general terms about the technologies they use and potential inconveniences during the implementation of software required for their research. The information provided included not only the specific tools they use but also information about the machine-tools used at the application stage of their research and the methodologies the use. Quantitative data was collected by formulating questions about the amount of time it usually takes to create a simulation, the required steps and their complexity. The participants were asked, how to perceive the efficiency of the current methodology and how satisfied they are on a scale of the interval $[0, 5]$ in full integer steps.

RQ2: As a follow-up, the participants were confronted with the idea, that given a software tool to automatically generate the simulation environments, what kind of requirements and Use-Cases such a software should implement and for which exact use. This developed into a productive dialog were specific ideas and potential requirements for a reference concept could be gathered, and the answers were directly put in relation with current research and related work. Further, questions on possible extension of such a software were asked to ensure, that required interfaces and dependencies were not left out during the concept creation.

RQ3: Finally, the participants were asked about hypothetical uses of such a software in terms of applying it to real physical machine-tools. Again a monologue arose and led to concrete ideas of applications. Additionally, questions about the perception and trust to such a software were asked to include possible required interactions in the generation process. Quantitative data was gathered by asking the participants to rank the priorities of the requirements and Use-Cases drawn during the interview.

4.3 Summary

During the course of the research for this thesis, quantitative results were obtained by literature review and specific questions in the style of a questionnaire during the conducted semi-structured interviews. These interviews provided the qualitative results for the research including insights to domain knowledge, common inconveniences and challenges in the regular work and research life of the participants. Following the flexible design and an explorative research strategy new data and insights from different source could be integrated into the concept resulting from the aggregated results. Table 4.2 summarizes the research approach of this thesis.

Table 4.2: Summary of the research approach.

Research Strategy	Explorative
Research Design	Flexible
Research Category	Inductive Research
Methods	Metric measures Experts interviews Survey
Type of Collected Data	Quantitative Qualitative
Question Types	Open and Closed
Type of Experts Interview	Semi-Structured
Arrangement Type of Experts Interview	Remote audio and video calls

5 Evaluation and Results

This chapter provides the aggregated quantitative and qualitative findings after the research was conducted as described in Chapter 4. Since it is a challenging task to find fitting participants and even more candidates, which conduct research in the field of RL and manufacturing, participants were selected from academia and industrial corporations equally. It should also be noted here that each participant's research deals with a different, specific subfield, depending on the interests of the chair or the particular employer.

5.1 Quantitative Data

In the following the quantitative data gathered by explicit formulated questions during the interviews is presented. Both the aggregated results as diagrams and the initial tabular data is provided along with partial discussion on the conclusions made from the data.

Professional Experience Figure 5.1 and the corresponding Table 5.1 show, that the participants have in average 4 years of professional experience in general and around 2 years of experience in the field of RL in the context of robotics and manufacturing. This is due to the problem of finding suitable study participants, which was mentioned at the beginning of the study. Nevertheless, the selected participants can be seen as valuable sources of domain expertise and knowledge.

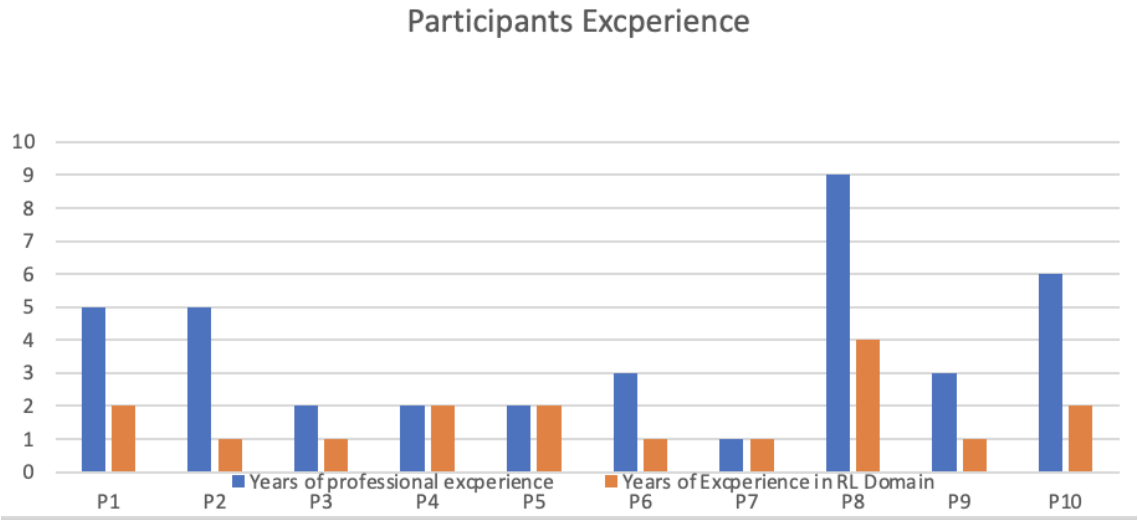

Figure 5.1: Pie chart of the distribution of participants professional experience and the experience specifically in the field of RL.

Table 5.1: Distribution of the participants professional experience and their respective experience in the field of RL.

Participant ID	Years of professional excperience	Years of Excperience in RL Domain
P1	5	2
P2	5	1
P3	2	1
P4	2	2
P5	2	2
P6	3	1
P7	1	1
P8	9	4
P9	3	1
P10	6	2

Professional Background Figure 5.2 and the corresponding Table 5.2 show that the majority of the interviewed participants works in the industry with 60% while 40% work as researchers in academia. This circumstance also reflects the current maturity of research and its adaptation in the industrial environment.

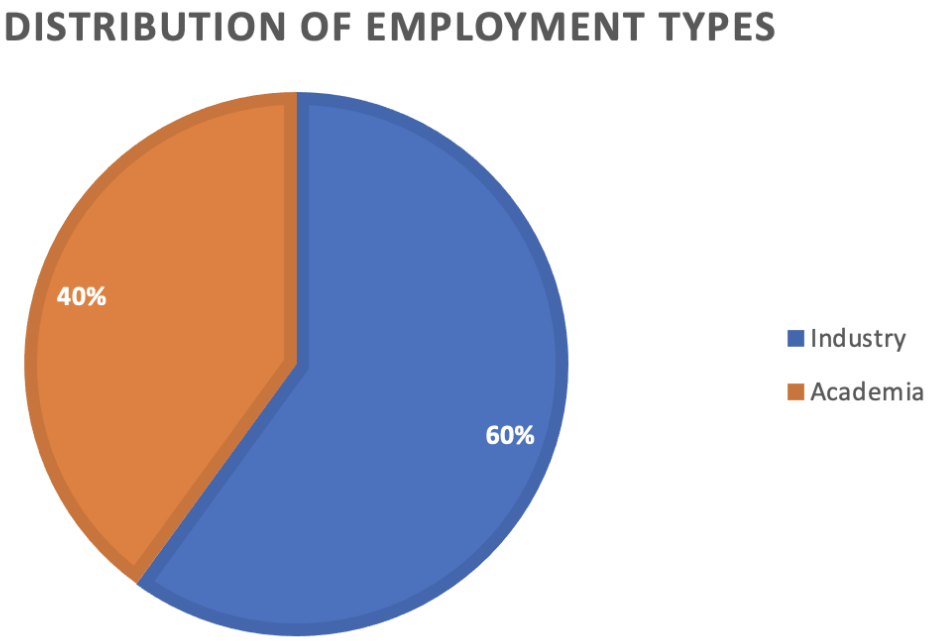

Figure 5.2: Pie chart of the distribution of participants employment types.

Table 5.2: Mapping of the participants to their employment types in academia or the industry.

Participant ID	Employment Type
P1	Academia
P2	Academia
P3	Industry
P4	Academia
P5	Industry
P6	Industry
P7	Academia
P8	Industry
P9	Industry
P10	Industry

Implementation Frequency During the interview, participants were asked to estimate how often they have to implement simulation environments from scratch, or add to existing environments, in their daily work. Figure 5.3 and the corresponding Table 5.3 present the results. *Strongly Disagree* here refers to the fact that the participant is almost never involved in the implementation of simulation environments. *Strongly Agree* refers to a regular involvement in the development process.

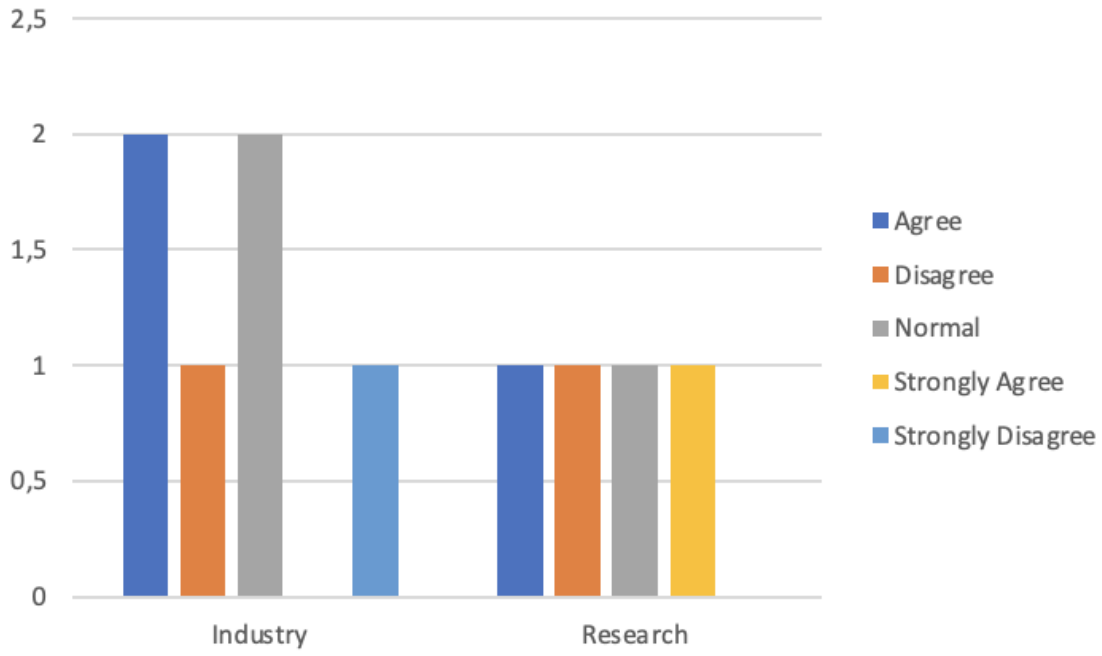

Figure 5.3: The distribution of the frequency of creation or enhancement of simulation environments in relation to the employment field.

From the provided results we can initially deduce that at least 40% of the participants have no regular or rare interaction with the implementation of simulation environments in general. The specification to concrete RL simulation environments was explicitly omitted since a participant may not be involved in the RL part of a research project but responsible for the machine-tool and DT related implementation and simulation tasks.

Table 5.3: The distribution of the frequency of creation or enhancement of simulation environments in relation to the employment field.

Participant ID	Employment Type	Regularity of Implementation
P1	Research	Strongly Agree
P2	Research	Agree
P3	Industry	Normal
P4	Research	Disagree
P5	Industry	Disagree
P6	Industry	Agree
P7	Research	Normal
P8	Industry	Agree
P9	Industry	Strongly Disagree
P10	Industry	Normal

Used Tools For gathering more information of the tools and concepts used by the participants in their regular work, they were asked to name 5 tools or programming languages they use. The following data is the aggregation distinct tools which are publicly available for industry and research equally.

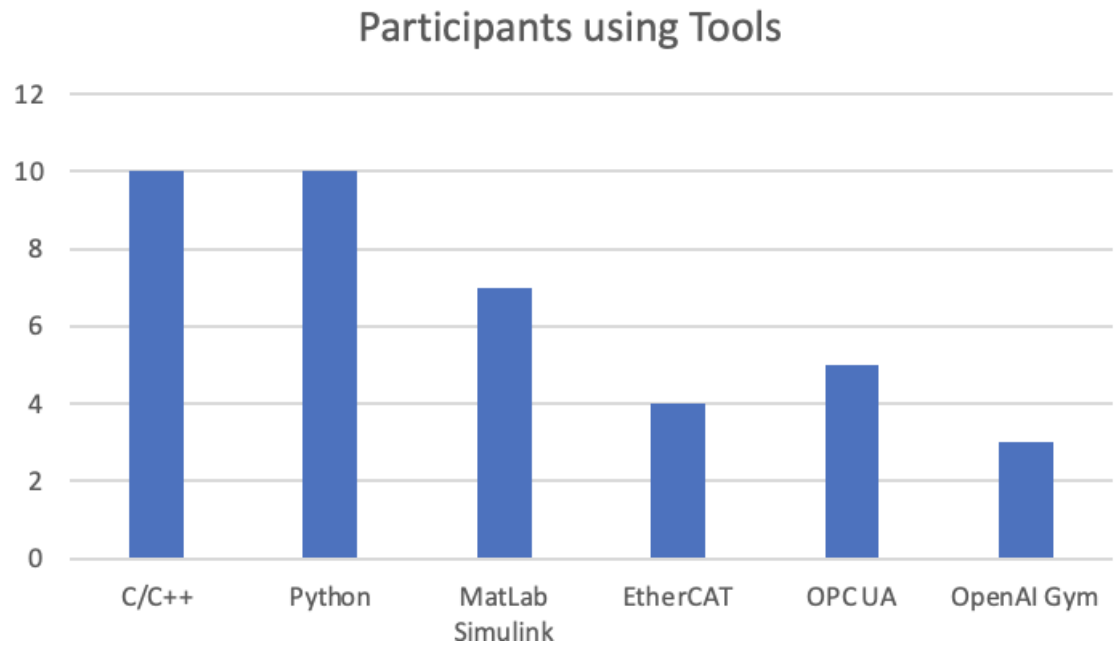

Figure 5.4: Representation of the six most frequently used tools by the participants.

Table 5.4: Representation of the six most frequently used tools by the participants.

Tools and Programming Languages	Participants using Tools
C/C++	10
Python	10
MatLab Simulink	7
EtherCAT	4
OPC UA	5
OpenAI Gym	3

Current Satisfaction Rate As a final quantitative question, participants were asked to describe their satisfaction with the current development processes and the effort involved. It turned out that all but one participant are either satisfied with the current development methods or at least neutral towards them.

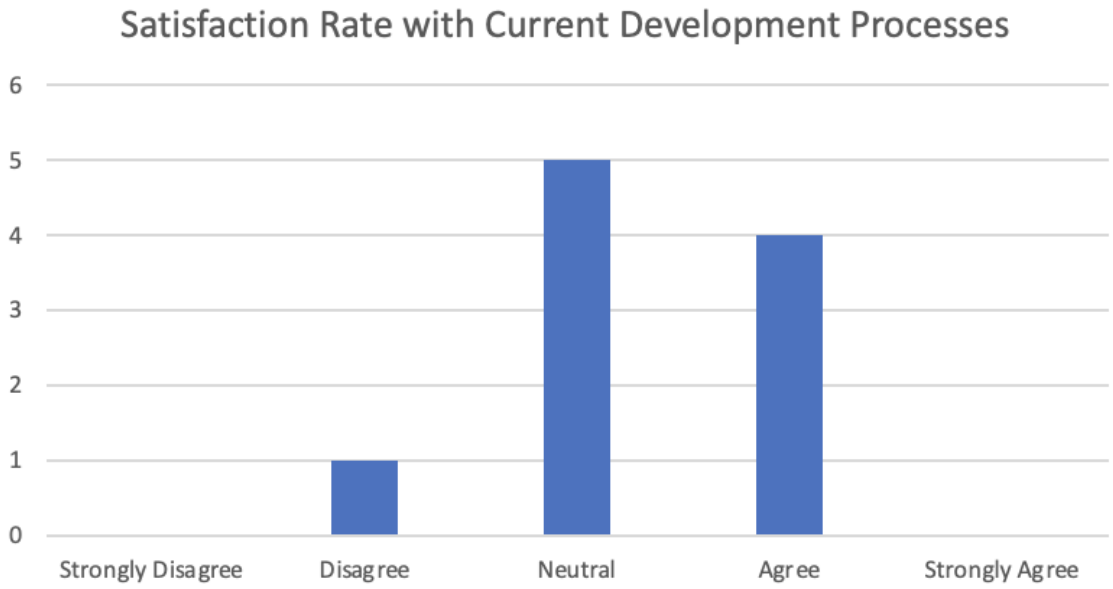

Figure 5.5: Satisfaction rating of the participants with the current development and research processes.

Table 5.5: Satisfaction rating of the participants with the current development and research processes.

Satisfaction	Count
Strongly Disagree	0
Disagree	1
Neutral	5
Agree	4
Strongly Agree	0

5.2 Qualitative Data

Ten semi-structured interviews were conducted with experts to gain insight into your daily work and research routine and how your currently used work methodologies and tools can be improved, and recurring tasks can be automated. For each interview the time frame was set around 45 minutes. The following section presents the results of the interviews in aggregated form.

After starting the dialogue by collecting quantitative data such as career field and the amount of years in professional employment, participants were first asked in general terms about the tools they used to create and use simulations of machine tools in their research. Participants were advised that they were free to list programming languages, communication technologies, simulation-related, and other software-related tools. Later, the participants were then asked to focus on recurring processes in their work and especially in the implementation and to list the work packages. After setting the framework for further dialog, participants were now asked to imagine having a software available that automatically handles recurring tasks such as setting up projects and generating boilerplate code. It was asked what functionalities such software should have in order to significantly improve the satisfaction and efficiency of the participants' work. Potential use cases were further elaborated and subsequently prioritized. In particular, technical details were addressed in this dialog, for example, at which points the participant attaches importance to being able to intervene in and adapt the generation processes. The interview was concluded by an open dialogue about potential application scenarios of such software. In particular, questions of feasibility, integration with real machine tools, reusability and portability were addressed.

5.2.1 Results

In the following, the results and similarities between the answers of the participants will be presented. All questions were posed open with the intent to stimulate a dialogue with the participant.

Would you be so kind to describe the tools which you use for creation of simulations in the manufacturing domain? The majority of participants separated the answer into the following aspects: (i) machine-tool control and communication profiles, namely EtherCAT and OPC UA as well as MQTT, field bus systems like Profinet and Sercos, (ii) programming languages used for implementation of the simulations, namely C/C++, MatLab Simulink and Python, and (iii) the different types and machine-tool PLCs they use.

Besides basic software development tools like integrated development environments and source code management systems, all participants strongly emphasized that the decision of the used tools depends on the type of the machine-tool and its control and communication capabilities. OPC UA and EtherCAT could be identified as common denominators in terms of communication protocols. All participants mentioned, that they rely on modularization and abstraction of interfaces to integrate and test different approaches based on existing projects. Further, all participants named the OpenAI Gym framework when asked for the framework they use for RL purposes and Python as the corresponding programming language.

The majority of participants does not create simulation environments from scratch but rather enhances and complements existing simulations and models. One participant's research work can't even be simulated, and the participant must perform development and testing tasks directly on the control of the machine-tool.

If you think about repetitive tasks in your day-to-day work, how do they look like and which other time-consuming tasks could you think of? For this question, all participants agreed that the most time- and effort consuming task is the implementation of wrapper and boilerplate code to integrate different concepts into their work. Depending on the complexity of the system, the participant's estimates on the duration of the tasks reached from days to months of work. Further, the majority of participants from enterprises agreed upon the lack of standardization of interfaces and methodologies used. Two industrial participants mentioned, that the design, implementation and experimenting with new RL algorithms and

neural networks takes only a third of the time they need, to integrate all stakeholder systems and implement the required boilerplate code. All participants mentioned the recurring task of documenting training results of the RL agents and the testing of different algorithmic approaches against each other.

Given a potential software solutions for automatic RL environment and agent generation. What Use-Cases and technical functionalities would you expect of the software? The dialogue took the most time of the interviews. The deductions made from this list will be discussed in Section 5.2.2 Each participant provided interesting ideas but after being asked for the feasibility of their suggestions, the following list defined the common denominator:

- automatic generation of base projects and the required boilerplate code,
- it should be possible to decide between the communication protocol to use and to configure the connectivity information in advance
- generate standardized projects without *hacky* fixes
- allow the deployment of an agent directly to the machine-tool
- possibility to test multiple agent implementations on the same environment
- possibility to instantiate a pre-configured environment
- logging of the agents actions in nearly real-time
- the ability to test different reward functions without changing the whole implementation

5.2.2 Discussion on Findings

The following discusses the key deductions and interpretations from the interviews. In this process, Experts opinions are analyzed thoroughly and Interpretations are made manually under software engineering aspects and the preceding literature research. The findings are discussed one-by one.

Automatic Base Project Generation Following the expert's opinions, the boilerplate code for the simulation environments and agents should be generated in a standardized manner. Users of a potential software should be able to select the programming language level and required libraries and dependencies, so the software can generate bot, environment and agent implementations. Either by setting CLI parameters or providing a YAML file, the experts suggested the usage of persistable configurations, so they can be shared between research partners.

Abstraction of Environment and Client Implementation Especially the academic employees emphasized the importance of the separation between the actual simulation environment and the implementation of a learning agent. Following the expert's opinion, this would improve the separation of work packages in a collaborative fashion as well as the maintainability and clearness of the implementations.

Generic Interface for the Reward Function Implementation One important research topic is the design of suitable reward functions for the agents to learn in a given environment. Experts suggested the separation of the reward function from the underlying agents implementation and providing it through a standardized interface. At the phase of the interviews it was left unclear, how this can be achieved.

Configurable Machine-Tool Connectivity Since heterogeneity of communication and machine-tool description standards is a known issue in the manufacturing domain, the experts suggested designing a protocol-agnostic solution. Potential users of the software should be able to select from reference implementations (e.g. OPC UA) and only provide the necessary connectivity details (e.g. IPv4 addresses, usernames and passwords) and the software should set up the connectivity based on the provided information.

Abstraction of the underlying Machine-Tool Capabilities Experts highlighted that the underlying semantic of a machine-tool description still requires a certain amount of domain expertise from an end-user. Therefore, the suggestion was to hide away the capabilities of the machine-tool and provide only possible parameters of the machine-tool to the user. Users could then select and adjust the required parameters themselves, but having at least a filtered view of available parameters would already be an enhancement.

Modularity and Extensibility of the Software Research is fast, and innovation cycles are short. Experts suggested designing a potential software as modular and loosely coupled as possible so new ideas and technologies can be integrated and tested fast. Further, heterogeneous environment of tools and methodologies used in all phases of the development requires a way to extend the existing implementations.

Manual Selection of In- and Output Parameters In the course of the interviews, the experts' desire to be able to intervene in individual aspects of the generation process crystallized. In particular, this wish was clearly expressed in the selection of in- and output parameters for simulation.

Instantiation of Environments and Agents For further applications of a potential software, the experts asked for a way to instantiate multiple agents and environments on different machines to research new approaches in parallel. Further, testing of different agent reward functions and different agents on the same environments could be a huge improvement for the time needed to aggregate training results and decide further development and research directions.

5.2.3 Derived Use-Cases

Figure 5.6 and Figure 5.7 provide a brief schematic overview of the core Use-Cases derived from the interviews. The following chapter will describe the underlying concepts and functionalities in detail and provide a reference architecture.

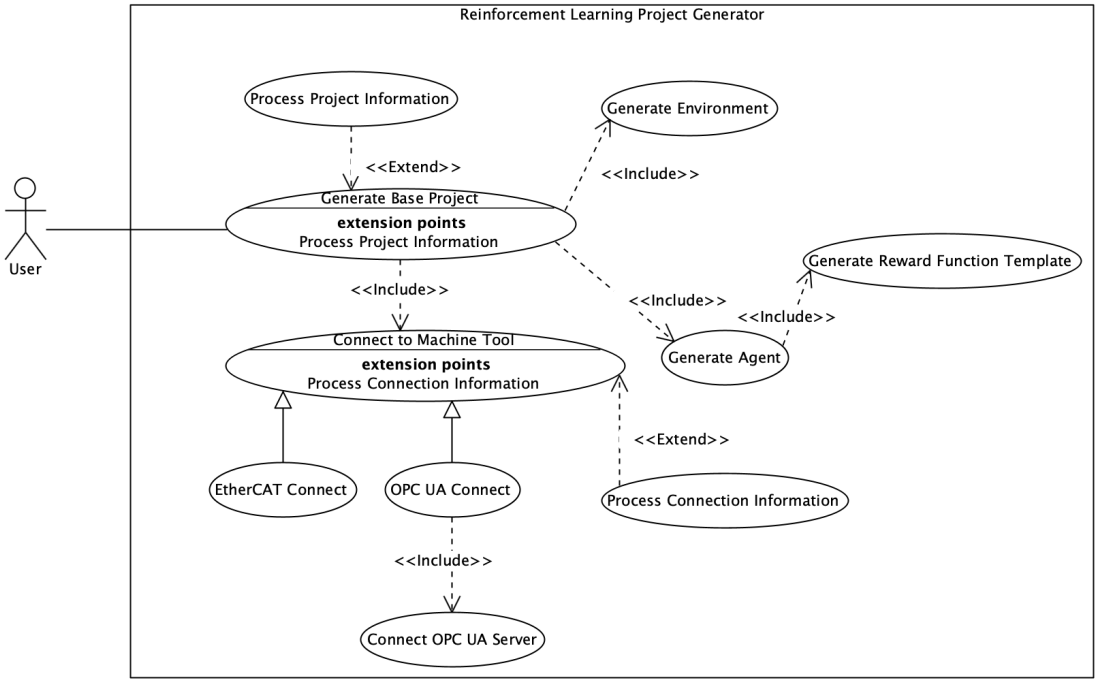

Figure 5.6: UML Use-Case diagram providing a brief overview of general functionalities.

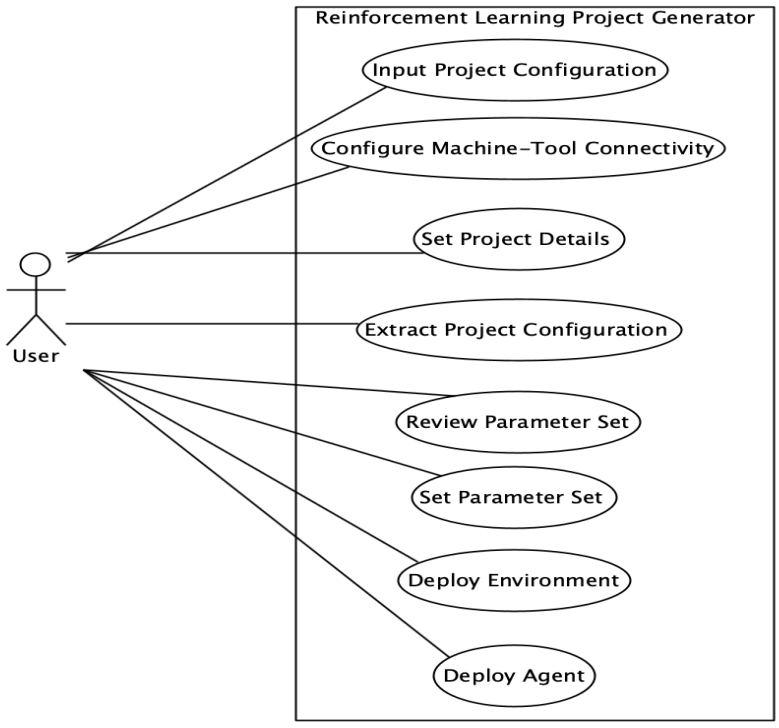

Figure 5.7: UML Use-Case diagram providing a brief overview of the user facing functionalities.

6 Concept

After research was conducted and the results were described in Chapter 4 a concept for the realization of a Reinforcement Learning Environment Generator is presented. The concept presented addresses the identified needs and suggestions of the surveyed domain experts and makes implementation proposals for a reference implementation. By incorporating practices from software engineering and common best practices, a modular, extensible, and loosely coupled software architecture is outlined.

Section 6.1 describes general software design decisions made based on the conducted interviews. Section 6.2 describes the methodology to parse a given machine-tool capability description and the process to generalize it so far that it can be passed to the generation system for further processing. Further, the section outlines the developed approach to distill states and actions from a given machine-tool description, so observations of the environment can be made and communicated bi-directionally to the agent. Section 6.3 deals with the decoupling of the Reward Function from the rest of the implementation. The topic of automatic training of the generated agent and environment is covered by Section 6.4. Finally, Section 6.5 concludes this chapter with a discussion of the concept and by naming some of its limitations.

6.1 Architectural Design

Adhering to common best-practices, the software should be designed as loosely-coupled as possible, modular and extensible. By using the tools the users are already used, the concept tries to fit into the existing workflows without any friction. Therefore, Python[1] was chosen as the programming language for the project. Although a possible reference implementation in the form of a command line interface tool (CLI)could be implemented in another language, the concept sticks to the tools commonly used by the experts. Further, Python is a platform independent language and provides several open source packages for the implementation of CLI tools. Additionally, all required dependencies have reference implementations or are solely implemented in Python. By making this decision, the concept strives to achieve modularity and extensibility.

Since the majority of the experts had at least some experience with the OpenAI Gym framework[2] the concept uses it for showing the integration of a RL framework. It should be noted, that the concept is designed in a framework-agnostic manner and can be extended by further RL frameworks like Google Dopamine.

6.1.1 Components

First, the functionality provided to the user (in the following the provided functionalities are accessed through a CLI) have to be separated into responsible services, namely: (i) a service responsible for the generation of the folder and file structure, generation of standard project boilerplate code like `setup.py` files and package structures as well as the naming of packages and basic dependency management, (ii) a connectivity service which handles the communication with the machine-tool simulation, (iii) a machine-tool description service providing the information needed to determine states and actions for the environment and the agent and finally (iv) a service responsible for the interactions of the user concerning the agent parameters, states and actions.

[1] See: https://docs.python.org/3/reference/
[2] See: https://github.com/openai/gym

By using a facade the concept hides implementation and interface details of the underlying technologies. This allows the integration of different communication protocols and machine-tools. Further, the separation between the communication and the description of its capabilities allows the integration of external description models. For the agent, the machine-tool itself is the environment with which it interacts. As longs as states and actions are provided the agent can be trained and evaluated against the provided environment. A researcher evaluating new algorithms shouldn't be bothered by the implementation details of the underlying machine-tool description model or the way, how the communication between the environment and the machine-tool is established. Figure 6.1 provides a UML component overview about the separated services and interfaces.

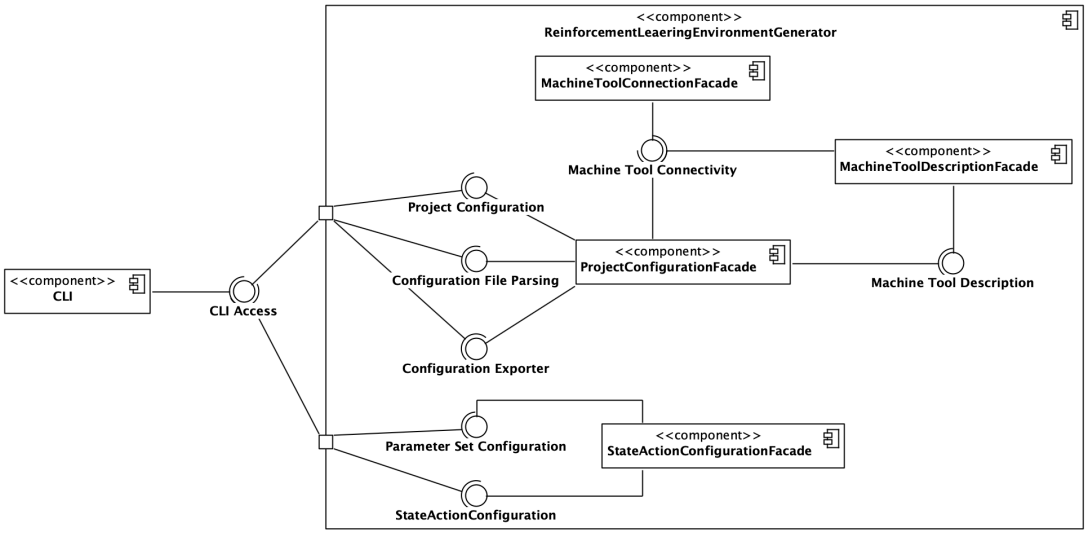

Figure 6.1: The UML component diagram describing the access to the systems functionalities through a CLI application.

As shown in Figure 6.1 the sole entry point for the systems functionalities is a CLI. The CLI provides formalized input parameters and formats. If required the contract between the CLI and the internal system could be migrated to a Web-capable API like REST or GRPC.

6.1.2 Project Configuration

The project configuration service handles the most time-consuming task - the setup of a Python project, installation of all required (and additional) dependencies, verifying that they don't interfere with each other, creation of the required folder structures and setting the right permissions according to the client's operating system. For illustration purposes, the constrained verification for a `framework` parameter has been added to the component diagram. Depending on the requested RL framework, the service creates the required files, manages the required class and file names as well as the package management in the folder structure (which is mandatory in frameworks like the OpenAI Gym) and stores the current configuration to a YAML file.

The possible configuration YAML is shown below.

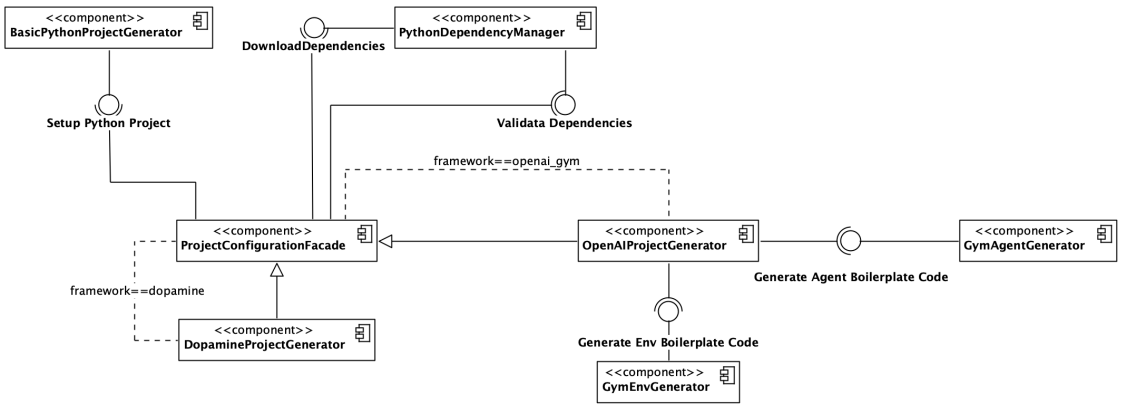

Figure 6.2: The UML component diagram the services which realize the creation of a Python project for RL.

Listing 6.1 Example Project Configuration

```
---
project_name: "foo"
base_project_path: "/path/to/project"
framework: "openai_gym"
dependencies:
- name: "py_dep_a"
   version: "1.2.3"
- name: "py_dep_b"
   version: "4.5.6"
---
```

6.1.3 Machine Tool Connectivity and Description

Both services shown in Figure 6.1 for connectivity are shown in an exemplary manner. The underlying key concept is the separation of the connectivity related tasks from the semantic description of the machine-tool and its capabilities. The concept separates the responsibilities of the services as follows: the connectivity service handles the connection to the machine-tool and provides permanent bi-directional access to it. The description service retrieves the information about the semantic properties of the machine-tool and provides it to interested services for future usage. Figure 6.3 shows the components of the connectivity service, and it's relation to an exemplary OPC UA server attached to the machine-tool simulation.

Analogous to the project configuration service shown in Section 6.1.2, a possible configuration in YAML is shown below.

The shown configuration object is instantiated for OPC UA connections. Future implementations have to handle the configuration files in a contextual manner, meaning that the files have to be parsed in advance and checked for provided parameters. Depending on the provided parameters, the service generates the according source code for the specific connectivity protocol.

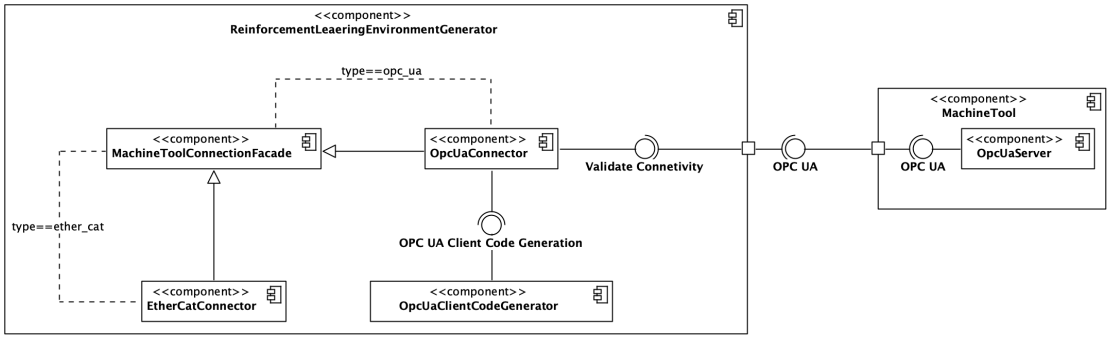

Figure 6.3: The UML component diagram the services which realize the creation of a Python project for RL.

Listing 6.2 Example OPC UA Machine Tool Connectivity Configuration

```
---
type: "opc_ua"
opc_ua_configuration:
    connection_type: "ua_tcp"
    connection_address: machine.tool
    port: 53530
    server_name: "OPCUA/SimulationServer"
    security_modes: "None, Sign, Sign\&Encrypt"
    security_policies: "all"
    bind_address: true
---
```

6.2 Machine Tool Description Parsing

For the generation of the environment and agent, it is necessary to provide the possible states and actions for the agent. If one takes OPC UA Server Model as machine-tool description format, the server provides access to the address space, which is basically an itterable tree. Clients can read and write data to OPC UA Nodes which would be actions in the environment. The observation and state of the environment on the other hand would be the current values in the subscribed objects. Unfortunately, it is a challenging task to map a semantic value to the OPC UA nodes on the server, meaning that without domain expert knowledge, it is not possible to predict the semantics of a given node and therefore not possible to automatically assign it to a specific action of an agent.

But assuming, that provided with a mapping between *NodeIDs* and the corresponding action identifiers of an agent, a generator could create method bodies and the according boilerplate code in the agent implementation. In this way, the user would have to implement the functionality of the method bodies but would at least be sure, that all sets of required actions and states of the environment were generated correctly. If no configuration of states and actions is provided to the software, the concept suggests the following workflow.

First, the description of the machine-tool is fetched and stored in an intermediate format. In this format, all writable nodes in a specific functionality space of the machine-tool are annotated as *states*. Accordingly, intermediate *actions* to write on these nodes are generated. The user is now provided with this intermediate mapping and has to manually select, which states and actions should be kept. After the user of the software finishes the labeling process, all non-marked states and actions are deleted from the intermediate file and the file itself is now transferred to a valid configuration and provided to the environment and agent implementations.

6.3 Reward Function Definition

Chapter 2 described the relevance of reward functions for RL agents. Depending on the algorithm and policy used, the implementation of the reward function requires different approaches. To separate the implementation of reward functions from the agent's step function implementation, we use the strategy pattern as follows:

The generated code contains the call to a parametrized client interface. Depending on the types and amount of variables (which can be functions as well) the interface calls another implementation of the reward function. The result of the function is then calculated and returned to the caller. A schematic representation of the used pattern is presented in Figure 6.4.

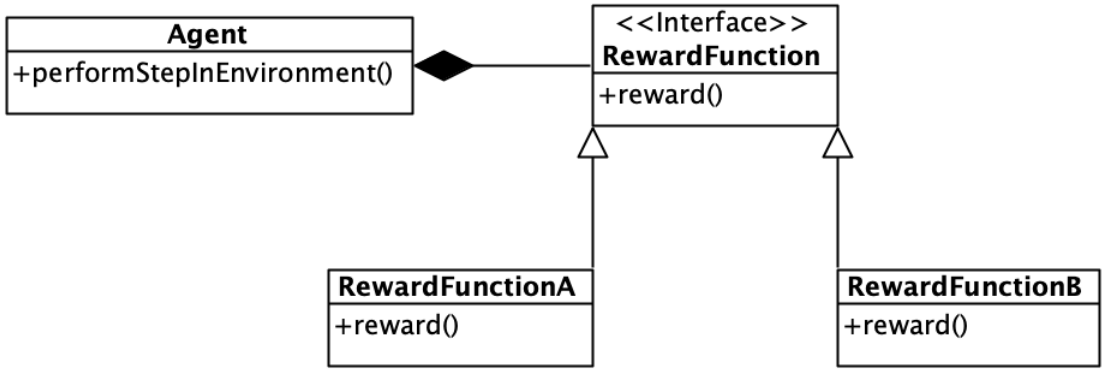

Figure 6.4: UML class diagram of the Strategy pattern implementation for the reward function.

By using callbacks at a global scope in the training environment, it would be even possible to inject another reward function at runtime. This can be realized by adding a global listener which permanently monitors the current training settings and if an asynchronous message for change arrives, replaces the implementation details of the reward function with the required ones. It should be noted that this is implementation and programming language dependent and represents a future research topic.

6.4 Automatic Training

Using the generation system and the tools provided by open source frameworks, parallel training can be achieved using virtualization technologies like Docker[3] and Kubeflow[4]. By relying on computation capacities available through Cloud Computing, multiple approaches can be trained by deploying different virtualization containers in which, agents are trained in parallel.

If an approach reaches a specific threshold of certainty, using the proposed concept, researchers could deploy the agent on a real machine by simply replacing the connectivity of the simulated machine-tool with the real one. Although special attention and excessive testing in advance should be conducted before deploying onto a physical machine-tool and taking cost-sensitive risks.

[3] See: https://www.docker.com/
[4] See: https://www.kubeflow.org/

6.5 Discussion

The presented concept provided a reference architecture for a framework- and machine-tool agnostic generation of RL environments and agents. Generation was realized by splitting the responsibilities of generation steps into several services with a unified interface. General projects can be generated with the required dependencies and structures depending on the selected framework. Potential users have the possibility to create configuration files for their specific needs and when necessary, enhance the implementation details to extend the software's functionalities. Machine-tool interaction is separated from the actual environment generation since only a connection to the machine-tool and read/write access to the control units is required. By this separation existing machine-tool simulations can be integrated using for example OPC UA and reused for different projects if needed. Further, agents can be deployed directly to the machine-tools since the training was already performed on the digital twin of the physical machine.

Limitations: The presented approach is not fully autonomous and still requires the user's domain expertise to specify states and actions in the environment. Nevertheless, using intermediate description files for the state-action mapping, the users can pre-configure the desired machine-tool functionalities and enhance the workflow of generating a RL environment and the according agent.

7 Conclusion and Outlook

Conclusion This thesis proposed a software architectural approach for generation of reinforcement learning environments and agents from machine-tool descriptions. The presented concept is based on the empirical research through semi-structured interviews with domain experts, which was conducted in advance.

First, the foundations of Digital Twins, Machine-Tool Description and Communication methods and Reinforcement Learning were presented. Certain Reinforcement Learning frameworks have been highlighted. By relying on the observations made in the literature review for the foundations and related work, an empirical study could be designed and results aggregated to specific Use-Cases which build the base for the presented concept. Finally, the formulated an approach with which a generator can be implemented to facilitate the daily work of research in the domains of Reinforcement Learning and Manufacturing. Limitations of the concept were discussed to stimulate future research work.

Outlook The presented concept serves as a starting point for further research and development. Future works could create a reference implementation including the testing with a physical machine. An *arena* like experiment similar to the OpenAI *Hide and Seek* paper could be a possible research target as well as the research on decision-making, when a trained agent is sufficiently secure to be deployed onto a real machine-tool.

Bibliography

[AAC+19] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. “Solving rubik’s cube with a robot hand”. In: *arXiv preprint arXiv:1910.07113* (2019) (cit. on pp. 23, 25).

[BCP+16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba. “Openai gym”. In: *arXiv preprint arXiv:1606.01540* (2016) (cit. on p. 25).

[BL15] B. Bagheri, J. Lee. “Big future for cyber-physical manufacturing systems”. In: *Design world* 23 (2015) (cit. on p. 17).

[Can16] A. Canedo. “Industrial IoT lifecycle via digital twins”. In: *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. 2016, pp. 1–1 (cit. on p. 15).

[CEV17] A. Csiszar, J. Eilers, A. Verl. “On solving the inverse kinematics problem using neural networks”. In: *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE. 2017, pp. 1–6 (cit. on p. 15).

[CJOC10] G. Cândido, F. Jammes, J. B. de Oliveira, A. W. Colombo. “SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications”. In: *2010 8th IEEE International Conference on Industrial Informatics*. IEEE. 2010, pp. 598–603 (cit. on p. 18).

[CLQS19] J. G. Campos, J. S. López, J. I. A. Quiroga, A. M. E. Seoane. “Automatic generation of digital twin industrial system from a high level specification”. In: *Procedia Manufacturing* 38 (2019), pp. 1095–1102 (cit. on p. 27).

[CMG+18] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, M. G. Bellemare. “Dopamine: A Research Framework for Deep Reinforcement Learning”. In: (2018). URL: http://arxiv.org/abs/1812.06110 (cit. on p. 25).

[ESLR19] C. Ellwein, A. Schmidt, A. Lechler, O. Riedel. “Distributed Manufacturing: A Vision about Shareconomy in the Manufacturing Industry”. In: *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*. 2019, pp. 90–95 (cit. on p. 15).

[FP19] D. M. Fernández, J.-H. Passoth. “Empirical software engineering: from discipline to interdiscipline”. In: *Journal of Systems and Software* 148 (2019), pp. 170–179 (cit. on p. 31).

[GBCB16] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016 (cit. on p. 22).

[Had06] T. Hadlich. “Providing device integration with OPC UA”. In: *2006 4th IEEE International Conference on Industrial Informatics*. IEEE. 2006, pp. 263–268 (cit. on pp. 18, 19).

[HKY18] S. Ha, J. Kim, K. Yamane. “Automated deep reinforcement learning environment for hardware of a modular legged robot”. In: *2018 15th International Conference on Ubiquitous Robots (UR)*. IEEE. 2018, pp. 348–354 (cit. on p. 27).

[HS14] R. Henßen, M. Schleipen. “Online-Kommunikation mittels OPC-UA vs. Engineering-Daten (offline) in AutomationML”. In: *Tagungsband Automation 2014* (2014), pp. 59–74 (cit. on pp. 18, 19).

[IJ13] J. Imtiaz, J. Jasperneite. “Scalability of OPC-UA down to the chip level enables “Internet of Things””. In: *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE. 2013, pp. 500–505 (cit. on p. 19).

[JCKV18] F. Jaensch, A. Csiszar, A. Kienzlen, A. Verl. “Reinforcement learning of material flow control logic using hardware-in-the-loop simulation”. In: *2018 First International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE. 2018, pp. 77–80 (cit. on pp. 15, 25).

[KCV18] B. Kaiser, A. Csiszar, A. Verl. “Generative models for direct generation of cnc toolpaths”. In: *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE. 2018, pp. 1–6 (cit. on p. 25).

[KKT+18] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn. “Digital Twin in manufacturing: A categorical literature review and classification”. In: *IFAC-PapersOnLine* 51.11 (2018), pp. 1016–1022 (cit. on pp. 17, 18).

[Kre14] D. Krenczyk. “Automatic generation method of simulation model for production planning and simulation systems integration”. In: *Advanced Materials Research*. Vol. 1036. Trans Tech Publ. 2014, pp. 825–829 (cit. on p. 27).

[LLK+20] Y. Lu, C. Liu, I. Kevin, K. Wang, H. Huang, X. Xu. “Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues”. In: *Robotics and Computer-Integrated Manufacturing* 61 (2020), p. 101837 (cit. on p. 17).

[LZYW20] B. Li, H. Zhang, P. Ye, J. Wang. “Trajectory smoothing method using reinforcement learning for computer numerical control machine tools”. In: *Robotics and Computer-Integrated Manufacturing* 61 (2020), p. 101847 (cit. on p. 15).

[MLC20] R. Minerva, G. M. Lee, N. Crespi. “Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models”. In: *Proceedings of the IEEE* 108.10 (2020), pp. 1785–1824 (cit. on p. 15).

[MOW+21] M. C. May, L. Overbeck, M. Wurster, A. Kuhnle, G. Lanza. “Foresighted digital twin for situational agent selection in production control”. In: *Procedia CIRP* 99 (2021), pp. 27–32 (cit. on p. 27).

[MSKV18] G. S. Martıénez, S. Sierla, T. Karhela, V. Vyatkin. “Automatic generation of a simulation-based digital twin of an industrial process plant”. In: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2018, pp. 3084–3089 (cit. on p. 27).

[NB18] S. Niehoff, G. Beier. “Industrie 4.0 and a sustainable development: A short study on the perception and expectations of experts in Germany”. In: *International Journal of Innovation and Sustainable Development* 12.3 (2018), pp. 360–374 (cit. on p. 15).

[NNXR08] A. Nassehi, S. T. Newman, X. W. Xu, R. Rosso Jr. “Toward interoperable CNC manufacturing”. In: *International Journal of Computer Integrated Manufacturing* 21.2 (2008), pp. 222–230 (cit. on p. 15).

[Obj17] Object Management Group. *The Unified Modeling Language Specification*. Ed. by Object Management Group. Dec. 5, 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (cit. on p. 18).

[OPC17a] OPC Foundation. *OPC 10000-1 - Part 1: Overview and Concepts*. Ed. by OPC Foundation. Nov. 22, 2017. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/ (cit. on pp. 19, 20).

[OPC17b] OPC Foundation. *OPC 10000-4 - Part 4: Services*. Ed. by OPC Foundation. Nov. 22, 2017. URL: https://reference.opcfoundation.org/v104/Core/docs/Part4/ (cit. on p. 19).

[PCB+19] F. Pires, A. Cachada, J. Barbosa, A. P. Moreira, P. Leitão. “Digital Twin in Industry 4.0: Technologies, Applications and Challenges”. In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. 2019, pp. 721–726. DOI: 10.1109/INDIN41052.2019.8972134 (cit. on p. 15).

[RN02] S. Russell, P. Norvig. “Artificial intelligence: a modern approach”. In: (2002) (cit. on p. 22).

[RSD10] M. Rostan, J. E. Stubbs, D. Dzilno. “EtherCAT enabled advanced control architecture”. In: *2010 IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. IEEE. 2010, pp. 39–44 (cit. on p. 21).

[RSK20] A. Rasheed, O. San, T. Kvamsdal. “Digital twin: Values, challenges and enablers from a modeling perspective”. In: *IEEE Access* 8 (2020), pp. 21980–22012 (cit. on p. 17).

[RTVG21] S. A. Rahman, L. Tuckerman, T. Vorley, C. Gherhes. “Resilient Research in the Field: Insights and Lessons From Adapting Qualitative Research Projects During the COVID-19 Pandemic”. In: *International Journal of Qualitative Methods* 20 (2021), p. 16094069211016106 (cit. on p. 30).

[SB18] R. S. Sutton, A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 22).

[SHS+18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144 (cit. on p. 25).

[TET12] E. Todorov, T. Erez, Y. Tassa. “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033 (cit. on p. 25).

[TV07] A. S. Tanenbaum, M. Van Steen. *Distributed systems: principles and paradigms*. Prentice-hall, 2007 (cit. on p. 19).

[VOX+05] S. Venkatesh, D. Odendahl, X. Xu, J. Michaloski, F. Proctor, T. Kramer. “Validating portability of STEP-NC tool center programming”. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 47403. 2005, pp. 285–290 (cit. on p. 19).

[Woh19] D. Wohlfeld. “Digitaler Zwilling für die Produktion von Übermorgen”. In: *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 114.1-2 (2019), pp. 65–67 (cit. on p. 15).

[WSJ17] M. Wollschlaeger, T. Sauter, J. Jasperneite. “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0”. In: *IEEE industrial electronics magazine* 11.1 (2017), pp. 17–27 (cit. on p. 15).

[WZLL21] C. Wang, L. Zheng, B. Li, Z. Li. “Design and implementation of EtherCAT Master based on Loongson”. In: *Procedia Computer Science* 183 (2021), pp. 462–470 (cit. on p. 21).

[XSK+21] K. Xia, C. Sacco, M. Kirkpatrick, C. Saidy, L. Nguyen, A. Kircaliali, R. Harik. “A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence”. In: *Journal of Manufacturing Systems* 58 (2021), pp. 210–230 (cit. on pp. 18, 27).

[YJ18] A. Yadav, S. Jayswal. “Modelling of flexible manufacturing system: a review”. In: *International Journal of Production Research* 56.7 (2018), pp. 2464–2487 (cit. on p. 27).

[YNGJ02] S. Yoo, G. Nicolescu, L. Gauthier, A. A. Jerraya. “Automatic generation of fast timed simulation models for operating systems in SoC design”. In: *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*. IEEE. 2002, pp. 620–627.

All links were last followed on June 1, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature