Educating Future Software Engineers for Industrial Robotics

Berit Schürrle[1], Philipp Grimmeisen[1], Jérôme Pfeiffer[2], Thilo Zimmermann[3], Andrey Morozov[1], Andreas Wortmann[2]

[1]Institute of Industrial Automation and Software Engineering (IAS), University of Stuttgart, Germany
[2]Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Germany
[3]InnovationCampus Future Mobility, Institut für Strahlwerkzeuge (IFSW), University of Stuttgart, Germany

{first}.{last}@{ias, isw, ifsw}.uni-stuttgart.de

Abstract

The industrial robotics landscape is drastically changing in recent years. New concepts, such as Industry 4.0, Industry 5.0, Software-Defined Manufacturing, IoT, and rapid development of AI require the changes in the education of the future engineers, especially IT specialist responsible for the software part of industrial robots. This paper presents two connected robotics education factories aimed at teaching key concepts in robotics, artificial intelligence, human-robot interaction, Internet of Things (IoT), digital twins, model-driven software engineering, and risk analysis. The factories serve as an educational platform for students and professionals alike, enabling them to learn practical experience in designing, developing, and testing robotics software engineering methods in a safe and controlled environment. The factories are equipped with multiple sensors, actuators, and controllers that facilitate the integration of different technologies, enabling learners to work on scenarios that simulate real-world industrial processes.

This paper introduces (i) seven educational goals that are crucial for modern and future robotic software engineers, (ii) architectures of the model factories and their components, and (iii) the underlying pedagogical approach, highlighting the importance of interdisciplinary collaboration. The paper concludes by discussing the potential of the factories to foster innovation and drive economic growth by equipping learners with the skills and knowledge needed to succeed in the digital economy.

1 Introduction

Modern production and manufacturing involve the integration of advanced software technologies such as Artificial Intelligence (AI), the Internet of Things (IoT), and cloud computing to optimize and streamline production processes. This helps to increase efficiency, reduce waste, and improve quality control. However, it also poses challenges such as the need for specialized skills and the potential for job loss due to automation. The implementation of the new industrial concepts is therefore a significant development for businesses and societies, requiring a collaborative effort from various domains, such as mechanical engineering, electrical engineering, robotics, software engineering, human-robot-interaction, artificial intelligence, and many more. [1]

Robotics, in modern industry, provides the capability to automate production processes and enable the creation of new production models, such as flexible and scalable production lines that can adapt to changing demands.

Software is a main driver of innovation for robotics because it provides the intelligence, decision-making capability, and integration that are necessary for advanced robotics production. The individual robots rely on ever-improving software algorithms. Software also provides the capability for real-time monitoring, prediction, and optimization of entire robotic systems, enabling them to continuously improve and adapt to changing production demands.

Contribution: The industry requires comprehensive education in robotics software engineering and related domains, such as IoT, digital twins, HRI, model-driven development, and many more [2]. To address this demand, we are setting up a distributed robotics software engineering education laboratory at the IAS and ISW of Stuttgart University. In this paper, we illustrate their rationale, configuration, and teaching concepts. With this, we aim to support improving the education of future robotics software engineers.

2 Seven Educational Goals

We identified seven challenging topics (educational goals) of networked factories and robotics, that are crucial for robotics software engineers. This section outlines their relevance and present the state-of-the-art technologies used in these fields.

1. Robotics Software: Robotics provides the means to automate repetitive tasks within the factory environment, for instance, moving goods between conveyor belts, punching iron, assembling parts, or drilling holes in a certain kind of material [3]. Thus, software engineers need to learn about sensor fusion, object detection, trajectory planning, and advanced robot control. Today, Autonomous Guided Vehicles (AGVs) [4] transport goods between factory lines autonomously. Therefore, aspects such as mapping and navigation within the factory, autonomously fetch, carry, and offload items, and even collaboration with

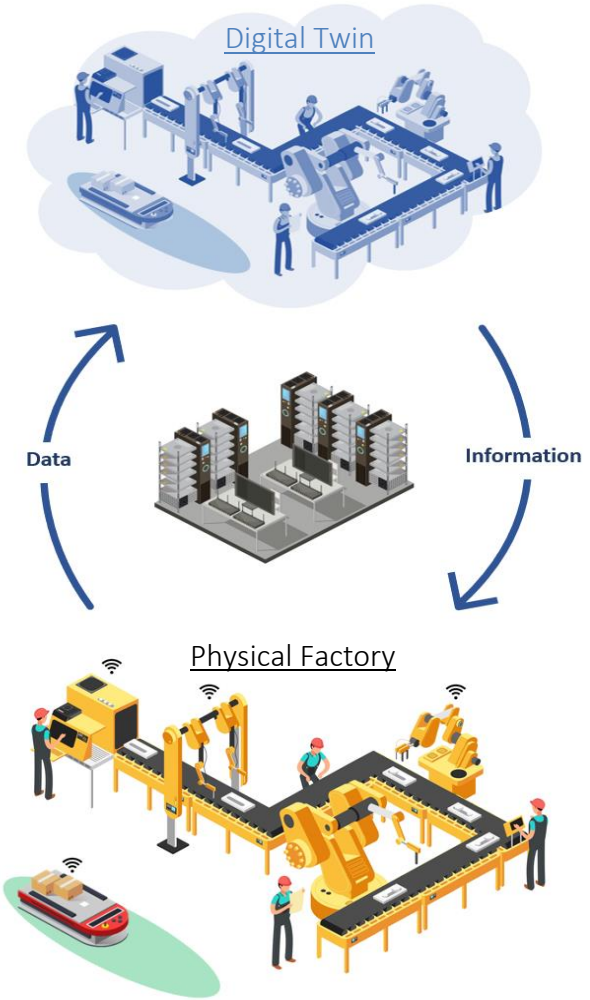

Figure 1 The general concept of the connected factories. The components of the production share their data via IoT-technologies, allowing for the creation of a digital twin. This digital replica helps to gain insights and knowledge about the production process and suggest corrective actions when needed.

other robots become relevant. For these aspects various technologies are crucial to learn. One example is Simultaneous Localization and Mapping (SLAM) [5] helps the robot to detect how the environment looks like (mapping) and where it is actually located (localization). To learn such technologies the engineers should know Robot Operating System (ROS and ROS2[1]) – de facto standard software development environment for robotic applications.

2. Artificial Intelligence: Due to the advancement of Artificial Intelligence (AI) and Deep Learning methods, the area of application is expanding rapidly throughout the entire industrial sector. AI and machine learning algorithms are gaining more and more relevance in industrial automation and are critical for the development of advanced robotics systems that can observe their environments, make decisions, and learn from experience. [6]. As the methods and algorithms of AI applications are quite versatile, they can be applied to various aspects of an automated system, ranging from the camera-based obstacle detection of an autonomous vehicle to predictive maintenance of the machinery. AI is actively used in wide range of industrial robotics tasks from object detection to robot motion control. However, the development of AI software is very different from the development of traditional software. It is important for future engineers to understand the approaches to AI software development, including the data set preparation, selection of a suitable AI architecture (MLP, CNN, LSTM, Transformers, Auto-encoders, GAN etc.), as well as the strengths and limitations of the neural networks. In addition to these theoretical concepts, engineers should also learn the AI development tools, such as TensorFlow[2] and Keras[3].

3. Human-Robot Interaction: In order to ensure safe and efficient collaboration, robotics software engineers need to carefully consider the Human-Robot Interactions (HRI). In the industrial production, there are several types of interaction possible between humans and robots. A common one is the teleoperation, where machinery or robots are remotely controlled by personnel, sometimes using VR. A more safety-critical one is the direct collaboration between humans and machines. In order to assist workers in their respective tasks, robots are often used to physically support and interact with humans. More advanced systems, however, go even further and offer a broader scope of interactions between robots and workers. Newer technologies include for example the use of Augmented Reality (AR), speech interaction through voice commands, gesture control or a visual interaction, where robots use cameras to detect and interpret the human behavior. As these technologies are gradually advancing and more tasks become a shared responsibility between human and robots, it is crucial for future robotic software engineers to understand the HRI concepts, technologies, and their limitations.

4. Internet-of-Things (IoT) [7] is a critical component of Industry 4.0 [8]. IoT describes the concept of connecting distributed, decentralized "things" to the internet or an edge network. IoT becomes more and more relevant to Industry since factory floors are often crowded with different PLCs and embedded computers of robotic manipulators, mobile robots, and other machines. To be controlled and monitored effectively all these components have to be connected via wired or wireless network. IoT aims to provide interoperability through standards that are critical to the seamless exchange of data between systems. Future robotics software engineers should be familiar with standards and protocols for connecting and integrating robotics systems with IoT devices, for instance, AutomationML [9], OPC-UA [10], or MQTT [11]. In our

[1] https://www.ros.org/

[2] https://www.tensorflow.org/

[3] https://keras.io/about/

model factories we aim to teach OPC-UA and MQTT as examples for such protocols.

5. Digital Twins: Through technologies like IoT, modern production facilities have access to large amounts of data. To turn this into valuable insights about the system(s), it has to be analyzed and processed. This can be done through the creation of a so-called digital twin. A digital twin is a software system that uses data, models, and services to purposefully represent and manipulate its (cyber-physical) counterpart [12]. As the digital twin aims to be an accurate representation of its physical twin for a specific purpose, it can be used to simulate its behavior, analyze its performance, and optimize its operation. As depicted in Figure 1, the digital twin is a useful mean to turn process data into knowledge about the factory and its parts, perform predictive diagnostics experiment with alternative futures via simulation and much more, thus, making it an essential part of modern manufacturing [13]. Future robotic software engineers need to be familiar with the concept and implementation of digital twins and, data synchronization, and understand how to retrieve valuable insights through the analysis of digital twins.

6. Model-driven Software Engineering: In Model-Driven Engineering [14], models are the central development artifacts. They bridge the gap between domain-specific problem space and the technical solution space of software, enabling domain experts to contribute (parts of) software solutions without needing to become software experts themselves. An important technology to this end are domain-specific languages (DSLs), which use concepts and terminology of a specific domain instead of general-purpose programming language concepts. Such DSLs include KUKA's KRL [13], the various languages of the Robotics DSL Zoo [13], and, recently, the application of low-code development in manufacturing [15], which relies on DSLs at the heart of every low-code development platform. It is necessary to educate robotic software engineers in these methods and extend them even to digital twins of robots that can also be defined in a model-driven way [16].

7. Risk analysis: Last but not least, systems are becoming exponentially more complex, with increased demands for interoperability and interconnectedness, integration of new technologies such as AI and re-configurable production systems. As a result of this complexity, the evaluation of potential failure scenarios becomes extremely challenging. This is especially true for safety-critical components and systems. In literature [17], risk is defined as a tuple, consisting of a failure scenario, the likelihood, and the resulting consequences. With increasingly complex systems, the search space for scenarios is increasing. Experience with older systems may prove non-transferable and therefore reliance on experience alone may prove inadequate. Established methodologies for likelihood and consequence estimation such as Event Tree Analysis (ETA), [18] Fault Tree Analysis (FTA) [19], or Bayesian Networks [20], may not be sufficient. Especially, when you consider rapidly reconfigurable systems by software or plug-and-play hardware modularization. In these cases, the risk may need to be automatically evaluated on models or digital twins prior to reconfiguration. The demand for engineers is likely to increase, both in terms of quality and quantity. It is essential that aspiring engineers are educated on established risk methodologies and made aware of the challenges mentioned above and the state-of-the-art solutions.

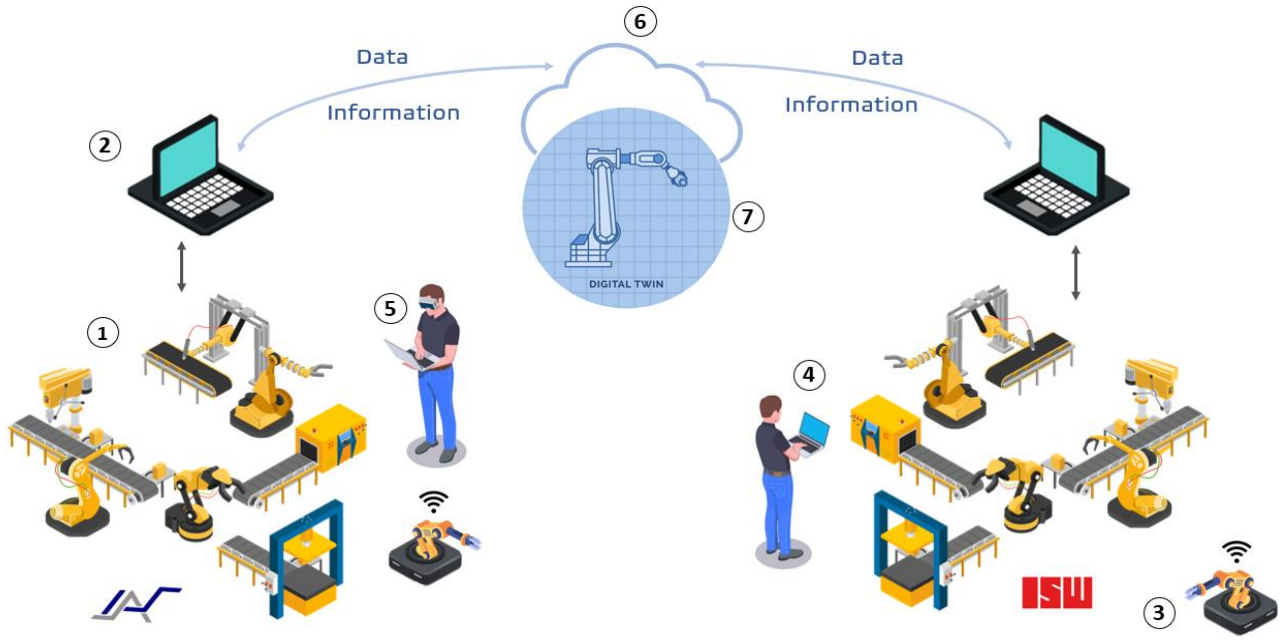

Figure 2 Schematical representation of connected model factories and their components: (1) Production lines, (2) Manufacturing Execution System, (3) Automated Guided Vehicles, (4) Maintenance Dashboard, (5) Augmented Reality, (6) Cloud Connection, (7) Digital Twin.

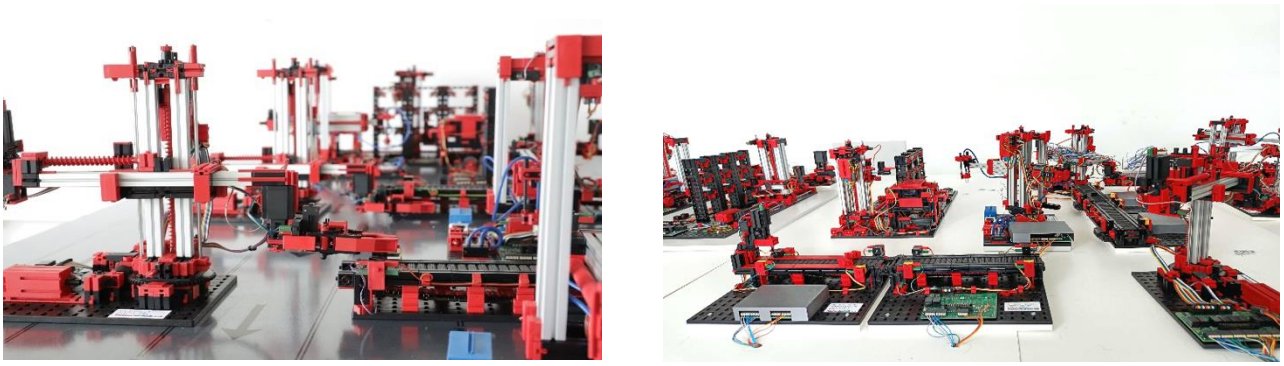

Figure 3 Photos of the Fischertechnik model factories assembled at the ISW (left) and the IAS (right).

3 Architectures of the Mini-Factories

To ensure the state-of-the-art education of future robotic software engineers, the Institute for industrial automation and software engineering (IAS) and the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), initiated a joined laboratory. The goal of this laboratory is to equip students with the important methods and skills addressing the beforementioned technologies and strengthen the understanding by some practical experience. For this reason, we created two miniature factories, imitating a real production environment (see Figure 2).

3.1 Production Lines

Each of our factories (see Figure 3) consist of up to 30 Fischertechnik[4] stations distributed among four production lines. These Fischertechnik stations are equipped with various sensors, such as light barriers or color detectors, as well as actuators like encoder motors and vacuum suction cups. In general, the components can be grouped in four categories: (i) Conveyor belts for the transportation between the stations, (ii) different grippers for the picking and placing of objects, (iii) processing station with various functionalities, such as drilling or sorting and lastly (iv) warehouses to store and retrieve products. For better comparability and to allow replanning in case of an unforeseen incident (fault tolerance), we designed some of the factory lines redundant. In order to process the input data from the sensors and trigger the respective actions of the different actuators, each factory is controlled by a programmable logic controller (PLC), in our case the RevolutionPi[5]. These RevolutionPis are connected to a central manufacturing execution system (MES) that plans and initiates the entire production process, monitors the stock and incoming orders from the web shop and manages the automated transport robots.

3.2 Manufacturing Execution System

The Manufacturing Execution System (MES) of our model factories can receive orders from a web shop, identify which production line is capable of adding specific product features, and plan a production process, i.e. define the steps to produce a certain product, instruct and monitor the factory stations and the AGVs, and manage the inventory of production goods.

We explicitly designed the MES to have multiple process layers, to improve maintainability and portability to other products with the help of two descriptions. The product description contains its features, the production steps, and a mapping to required skills, that have to be provided by a factory station to manufacture a certain feature. For instance, during the manufacturing step, a punching skill is required to put lid on a cup. The description of the capabilities of our factory stations then provides, that for instance, punching machines can punch, grippers can move and manipulate objects, etc.. Thus, from this, the MES can produce a plan to manufacture a product with the given stations of the model factory. This plan is then translated into commands that are sent to the PLCs of the production lines and the AGVs via MQTT topics The lines and AGVs answer with status messages about the current state of execution. Through this, the layers of our MES are loosely coupled, i.e., the command layer that interacts with the factory lines is independent from the product that is produced on it, and we can change the product according to our use case. For instance, for one exercise the factory could produce cereals or yoghurt, in another exercise, the factory could produce an automotive. Both use cases only differ in their product description. For the communication with the digital twin, our MES provides a REST interface.

[4] https://www.fischertechnik.de/en/products/simulating/training-models

[5] https://revolutionpi.com/

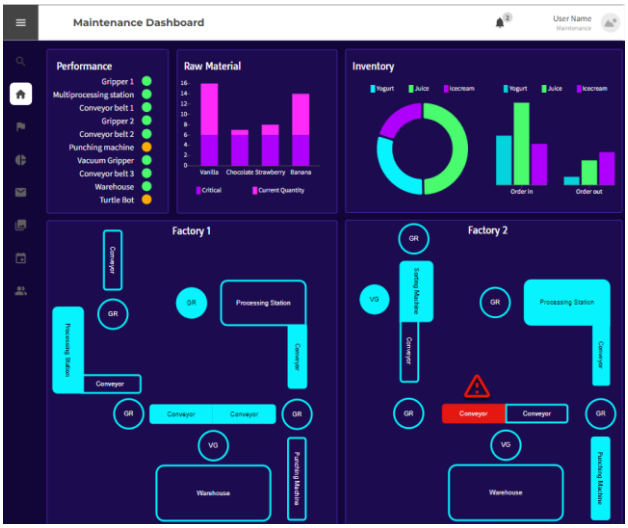

Figure 4 Graphical user interface of the maintenance dashboard

3.3 Automated Guided Vehicles

Automated Guided Vehicles (AGVs) are programmable mobile robot vehicles, that are used for the autonomous transportation of products and their parts. These vehicles with electric motors use computer algorithms to navigate across factories to transport items safely. We use small and configurable Turtlebot3 Waffle Pi mobile robots, shown in figure 5. They are capable of Simultaneous Localization and Mapping (SLAM), navigation and manipulation of objects. To create a map of the surroundings, a 360-degree lidar sensor on top of the Turtlebot is used. Submaps are built through lidar scanning, as the robot moves through the area, adding to the overall global map. The robot's location is estimated using the Adaptive Monte Carlo Localization (AMCL) method, see Cartographer SLAM [5]. Additionally, the lidar sensors, wheel encoders, and IMU sensors of the are used to navigate the robot. Adapted A* algorithm is used for route planning. In order to pick and place the transported items, we equipped the TurtleBot with an OpenManipulator arm. Through the integration of a camera and the implementation of object detection algorithms for computer vision, the TurtleBot can identify the requested object. The distance between the object and the robot can then be determined by the lidar sensor. Combining these sensor information's will allow us to issue instructions to the manipulator arm to pick or place the object from the station. ROS provides several packages for the control of the aforementioned tasks as an open and widely used framework.

3.4 Maintenance Dashboard

The model factories have the maintenance dashboards. Equipped with a Samsung tablets, students are able to monitor and analyze the production process. The aim of the dashboard is to provide the user a visual representation of the most important information. Figure 4 shows the dashboard interface. In the top-left corner, the performances of the individual components of the factories are displayed. These values allow to implement maintenance measures if a component performance is low. Next to it, the user can see the current amount of raw materials, stored in the warehouses. In addition to the current stock, it also displays the critical amount, which is an indicator to reorder certain raw materials before the production has to be paused. The pie chart in the top-right corner illustrates the amount of finished products in stock. In combination with the two bar charts representing the number of incoming and produced orders, the user has a clear understanding of the current warehouse stock.

The bottom part of the dashboard shows the state of the factories on schematic layouts. The user can see the current status of each component. Machines, that are running are highlighted, while the ones, that are idle are shown as passive. In addition, it displays disturbances and failures of the components, so the user can directly see where the problem is located. By clicking on an individual component, important process parameters of the respective component are shown.

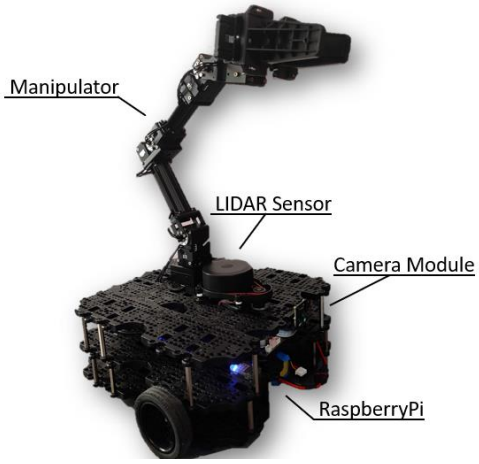

Figure 5 Mobile transport robot (TurtleBot3) and it's technical components

3.5 Augmented Reality

Augmented Reality (AR) serves as an overlay for model factories. The goal is to create a tool that can be used to control and monitor the factory's operations, as well as display general information and identify critical areas in addition to the dashboard.

The second generation Hololens are used. It features a variety of functions, including speech recognition and gesture control, which allow users to operate the headset without physical contact. Additionally, it is equipped with a powerful processor and high-resolution camera, enabling precise capturing of the environment. By leveraging AR technology, the application allow users to view and interact with the factory in real-time. The AR app enable users to visualize the various components of the factory, including the assembly lines, conveyor belts, and robotic grippers. Moreover, the AR offers a range of features designed to

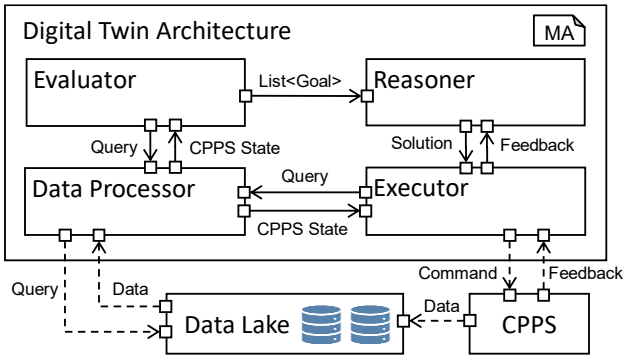

Figure 6 Digital Twin Architecture

simplify the factory's operation, including the ability to control individual components, monitor performance, and identify bottlenecks and other critical issues. For the realization of the software, we use Unity, a popular 3D engine that has become increasingly prevalent in the development of AR applications.

3.6 Cloud Connection

Our model factories share the data and information across both institutes via the cloud. Each factory streams the data to its MES, which in turn enriches the process data with further information about the entire system and transfers this to the digital twin in the cloud as shown in Figure 2. Through this process, the digital twin (further described in the following chapter) becomes a virtual replica of physical production. The MES can also retrieve information from the digital twins in the cloud. Therfore, each MES also has knowledge about the state of the other institute's factory. Through this shared knowledge, we support the collaboration and exchange between both productions. This can be especially relevant when, for example, one production faces problems and components stop working. In this case, the second factory can support and integrate the order in its own production planning to compensate the downtime. Another scenario is the delayed delivery of raw materials. If one factory runs out of raw materials for the production, the second one either produce the required goods or send raw materials from their stock.

3.7 Digital Twin

The top-level entity of our factory architecture that oversees the MES is the digital twin. Our architecture for digital twins (see Figure 6) is developed using MontiArc [21], a component & connector architecture description language [22]. It allows to model architectures as a hierarchy of interconnected components. The digital twin comprises a Data Processor component connected to a Data Lake to aggregate current Cyber-Physical Production System (CPPS) state data. The Evaluator monitors the state of the CPPS and verifies that the CPPS is operating as intended. In the event that it detects any anomalies, it creates goals that specify the intended state of the CPPS. The Reasoner finds a solution to the current situation and passes this solution to the Executor. The executor translates the provided solution into concrete CPPS settings and executes them on the CPPS. To specify the domain knowledge and the behavior of the digital twin, domain experts can use several modelling languages:

- UML/P class diagrams [23] for defining the domain model with its elements and their relationships.
- A language for Case-Based Reasoning (CBR) [24], which is a problem-solving paradigm. A case consists of a description of the situation as conditions properties of the domain model, its solution in terms of actions to be performed, and the situation the solution intends to produce.
- A communication specification language which enables defining communication of data types from a specified endpoint and with a defined protocol. Currently, the architecture supports communication via OPC-UA and MQTT.

4 Teaching concept

In Chapter 2 we have introduced seven goals for the education of future robotic software engineers. This chapter illustrates how we use our model factories (Chapter 3) to fulfil these goals and prepare our students for the real-world application of the important technologies.

4.1 Robotic Software

Students will work with model factories to investigate the capabilities of the production line and AGV software, cloud software that implements DT functions, as well as dashboard widgets. The AGV software is implemented in ROS and C++. Students will learn how to leverage ROS to realize mobile robotics tasks like guidance, navigation, and control, and how to use the manipulator to transport items. The production lines software is written in Python, which is an easy to learn, high-level language. The AGV cameras images will be used for teaching image recognition algorithms, for instance, for QR-code reading. Gazebo and RViz will be used for simulation and testing the students' software before the deployment to the actual robots.

4.2 Artificial Intelligence

To familiarize the students with the topic of artificial intelligence, we plan to take different approaches. The first aspect we want teach is the AI used for the AGV navigation. This not only includes smart path planning, but also the processing of visual information for object detection tasks in order to allow accurate pick and placing with the manipulator. The second aspect we plan to cover with our model factory is deep learning-based anomaly detection and AI-powered risk analysis, see Section 4.7.

4.3 Human-Robot Interaction

As this model factory covers many aspects of a networked production, students can learn about various interfaces for the interaction between robots and humans. One of them is

the augmented reality described in Section 3.4. In addition, the dashboard also allows for a direct interaction and control of the factory. The most safety-critical interaction, however, happens between the self-driving transport robots and their environment. Here we will cover the algorithms and methods for collision avoidance between robots and humans.

4.4 Internet-of-Things (IoT)

For teaching the protocols relevant for IoT, we plan a two-step exercise for OPC-UA where the students first create an OPC-UA server on one of the factory's RevolutionPis to represent the data available. In the second step, the students replace the existing connection of the MES to the RevPis with an OPC-UA client that can communicates with the server created in the first step. The second exercise is creating a MQTT-Publisher/Subscribe network between the factory islands/production lines to enable decentral communication independent of the MES.

4.5 Digital Twins

Our digital twin architecture is configurable by a multitude of modeling techniques that are easily comprehensible in short time for students. For instance, UML class diagram could be leveraged for modeling the data received from the MES. For the scope of a student lab or smaller project, the MontiArc architecture description language enables simple exchange of single components in the architecture. Students could exchange the current implementation of the planner component for case-based reasoning with another planning algorithm, e.g., AI-based, pddl [25]. By this, the students will learn how to create and work with digital twins and get to know their benefits in the industrial environment.

4.6 Model-Driven Software Engineering

The implementation of the MES and loosely coupled MES layers is already designed for model-driven engineering. Students will introduce a new product, write a product description model, and its manufacturing process via UML Activity Diagrams or BPM [26]. The software implementation of the MES performing the manufacturing of this product then will be automatically generated. In the future, low-code approaches [27] will be used to improve the digital twin with new features.

4.7 Risk Analysis

For the teaching of risk modelling and analysis, the first step for the students will be to identify the existing risks and to create and draft a risk model of the factory using event trees, fault trees, Markov chains, and possibly Bayesian networks. Once this is done, this model is then used to evaluate potential risks within the operation. Based on these results, the students' task will be to mitigate these risks and develop a more resilient production system using error detection and mitigation mechanisms and redundant components [28]. The redundancy of the factory is especially important, as it allows students to test and compare different mitigation actions and approaches with each other and compare the results.

5 Outlook

Industrial robotics is changing through the advancement and integration of new concepts like Software Defined Manufacturing, Internet of Things and Artificial Intelligence. Since these developments mainly address the software side of industrial systems, it is essential to prepare the new generation of software engineers for these changes and make them familiar with the seven technology fields presented in this paper. For this, two connected mini factories were developed. They provide the learning basis for our students and combine the theoretical knowledge with some hands-on experience.
Besides the integration of the factories in lectures such as Model-based Software Engineering and Risk Analysis, we continuously develop and improve the curriculum further. One aspect we want to integrate in the future is the collaboration with industry partners, to ensure that the curriculum is relevant and tailored to the latest industry needs. In addition, the extension of this learning concept to universities around the world and hence generating a network of connected factories covering various aspects of modern industry would be a great asset for all participants and the education of their students.

6 Literature

[1] Wortmann, A. et al. (2019) "Modeling languages in Industry 4.0: an extended systematic mapping study," Software and Systems Modeling. Springer Science and Business Media LLC.

[2] Butting, A. et al. (2018) "Teaching model-based systems engineering for industry 4.0," Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. MODELS '18: ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems, ACM.

[3] Goel, R., & Gupta, P. (2020). Robotics and industry 4.0. A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development, 157-169.

[4] Ullrich, G. (2015). Automated guided vehicle systems. *Springer-Verlag Berlin Heidelberg. doi*, *10*, 978-3.

[5] Stachniss, C., Leonard, J. J., & Thrun, S. (2016). Simultaneous localization and mapping. *Springer Handbook of Robotics*, 1153-1176.

[6] Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). Deep Learning. MIT Press.

[7] Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. Information systems frontiers, 17, 243-259.

[8] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., & Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. IEEE

transactions on industrial informatics, 14(11), 4724-4734.

[9] Drath, R., Luder, A., Peschke, J., & Hundigital twin, L. (2008, September). AutomationML-the glue for seamless automation engineering. In 2008 IEEE International Conference on Emerging Technologies and Factory Automation (pp. 616-623). IEEE.

[10] Leitner, S. H., & Mahnke, W. (2006). OPC UA–service-oriented architecture for industrial applications. ABB Corporate Research Center, 48(61-66), 22.

[11] Soni, D., & Makwana, A. (2017, April). A survey on mqtt: a protocol of internet of things (iot). In International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017) (Vol. 20, pp. 173-177).

[12] Kritzinger, W. et al. (2018) "Digital Twin in manufacturing: A categorical literature review and classification," IFAC-PapersOnLine. Elsevier BV.

[13] Dalibor, M., Jansen, N., Rumpe, B., Schmalzing, D., Wachtmeister, L., Wimmer, M., & Wortmann, A. (2022). A cross-domain systematic mapping study on software engineering for Digital Twins. Journal of Systems and Software, 111361.

[14] Beydeda, S., & Book, M. (2005). *Model-driven software development* (Vol. 15). V. Gruhn (Ed.). Heidelberg: Springer.

[15] Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). Low-code as enabler of digital transformation in manufacturing industry. Applied Sciences, 10(1), 12.

[16] Bibow, P., Dalibor, M., Hopmann, C., Mainz, B., Rumpe, B., Schmalzing, D., ... & Wortmann, A. (2020, June). Model-driven development of a digital twin for injection molding. In Advanced Information Systems Engineering: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, Proceedings (pp. 85-100). Cham: Springer International Publishing.

[17] Kaplan, S. and Garrick, B.J. (1981) "On The Quantitative Definition of Risk," Risk Analysis. Wiley

[18] Clemens, P. (2002) Event tree analysis. JE Jacobs Sverdrup 13.

[19] Ruijters, E. and Stoelinga, M. (2015) "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," Computer Science Review. Elsevier BV.

[20] Pearl, J. (2011). Bayesian networks. UCLA: Department of Statistics, UCLA.

[21] Butting, A., Haber, A., Hermerschmidigital twin, L., Kautz, O., Rumpe, B., & Wortmann, A. (2017). Systematic language extension mechanisms for the MontiArc architecture description language. In *Modelling Foundations and Applications: 13th European Conference, ECMFA 2017, Held as Part of STAF 2017, Marburg, Germany, July 19-20, 2017, Proceedings 13* (pp. 53-70). Springer International Publishing.

[22] Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., & Tang, A. (2012). What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering*, *39*(6), 869-891

[23] B. Rumpe, Modeling with UML: Language, Concepts, Methods. Springer International, July 2016. [Online]. Available: http://www.se-rwth.de/mbse/

(1) Analog Devices: Analog Design Seminar, Munich: Analog Devices GmbH, 1989

[24] Bolender, T., Bürvenich, G., Dalibor, M., Rumpe, B., & Wortmann, A. (2021, May). Self-adaptive manufacturing with digital twins. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 156-166). IEEE.

[25] Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., ... & Sun, Y. (1998). Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*

[26] Becker, J., Rosemann, M., & Von Uthmann, C. (2002). Guidelines of business process modeling. In *Business Process Management: Models, Techniques, and Empirical Studies* (pp. 30-49). Berlin, Heidelberg: Springer Berlin Heidelberg.

[27] Dalibor, M., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., ... & Wortmann, A. (2022). Generating customized low-code development platforms for digital twins. *Journal of Computer Languages*, *70*, 101117.

[28] K. Ding, A. Morozov and K. Janschek, "Classification of Hierarchical Fault-Tolerant Design Patterns," 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 2017, pp. 612-619.