A Digital Twin Exemplar for Fischertechnik Production Line Analysis, Simulation, and Optimization

Phil Pöllmann
phil.poellmann@gmail.com
University of Regensburg
Regensburg, Germany

Falco Wolf
falco_w2004@web.de
University of Regensburg
Regensburg, Germany

Jakob Titz
titz.jakob@gmail.com
University of Regensburg
Regensburg, Germany

Judith Michael
judith.michael@ur.de
Programming and Software Engineering,
University of Regensburg
Regensburg, Germany

ABSTRACT

While digital twin engineering for manufacturing is an interesting topic for students and trainees, getting access to production lines from industry is more than challenging. Educational institutions could overcome this challenge by using Fischertechnik production lines and their sensory information as original system for creating digital twins. This paper presents a digital twin exemplar for a Fischertechnik-based production line, realizing analysis, simulation, and optimization scenarios that was developed by four students during the fourth semester of their Bachelor studies in Computer Science and Data Science. The exemplar visualizes the recorded MQTT data of a Fischertechnik production line, enables the simulation of different scenarios with additionally developed virtual sensors, e.g., running certain machines faster might cause machines to overheat, and calculates optimizations for running machines, e.g., based on their throughput, or consumed energy. This and similar exemplars enable students to learn early on about the process of engineering digital twins, the use of different technologies, the relationship between data and its internal representation and different forms of analyses, what-if simulations and optimizations.

CCS CONCEPTS

• **Software and its engineering → Software architectures**; • **Applied computing → Industry and manufacturing**;

KEYWORDS

Digital Twin, Visualization, Simulation, Optimization, Production

ACM Reference Format:
Phil Pöllmann, Falco Wolf, Jakob Titz, and Judith Michael. 2026. A Digital Twin Exemplar for Fischertechnik Production Line Analysis, Simulation, and Optimization. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3837062.3839093

MODELS Companion 2026, October 04–09, 2026, Málaga, Spain

ACM ISBN 979-8-4007-2903-4/2026/10
https://doi.org/10.1145/3837062.3839093

1 INTRODUCTION AND MOTIVATION

Complex production systems place high demands on related software systems [9, 13, 32]. Production systems consist of many interacting components, generate large amounts of heterogeneous data, and evolve over their lifecycle [1, 12, 20]. At the same time, relevant information must be available as close to real time as possible in order to monitor the status of a plant, analyze processes, identify optimization opportunities, and predict potential failures early on [2, 23]. A Digital Twin (DT) can address these challenges by representing a production system [21, 24] and using the observed system data for purposes such as monitoring, analysis, simulation, and optimization [6, 7].

This paper presents a DT exemplar for a modular Fischertechnik production plant [10] (the physical system). This Fischertechnik setup represents a production process on a small scale [17] well fitting for education [15, 27]. It consists of machines that transport, sort, process, and store workpieces. Although it is small-scale, it represents some of the challenges of industrial production systems, such as having interdependent machine states, material and package flows, sensor and actuator signals, and time-dependent behavior.

With this work, we demonstrate how a digital twin for this production process can be engineered based on recorded data of the production line. The available interface to the physical system is based on Message Queuing Telemetry Transport (MQTT) [22] data and primarily enables monitoring of the system state. Direct access to the internal control logic or direct control of the physical system was not available in the current setup. Therefore, the system behavior had to be inferred from recorded MQTT data, a video recording of the production process, and publicly available documentation. This makes the case particularly relevant for the discussion of grey-box modeling, in which parts of the system are known, while other aspects must be derived from observable data.

This paper focuses on three main aspects. First, it describes the physical Fischertechnik production system and the architecture of the implemented digital twin. Second, it demonstrates how the digital twin supports both live monitoring and what-if analyses. Third, it reflects on the engineering process, particularly the derivation of machine behavior from MQTT data, the construction of a graph-based simulation model, and the integration of an optimization component. The source code is available at GitHub [25].

Paper structure: The next section introduces the physical system, Section 3 the implemented digital twin, and Section 4 the digital twin engineering process. Section 5 discusses the key findings, limitations, and potential improvements. The last section concludes.

2 THE PHYSICAL SYSTEM

Our physical system is a modular production facility by Fischertechnik located at RWTH Aachen University [4] that is used in the Model-Based DevOps (MBDO) project [5]. The system consists of five machines, representing a small-scale production line. It transports packages through multiple stations, where they are sorted, processed, and stored: The system starts by sorting the packages according to their color (red, blue, white) using a sorting line. The transport mechanism consists of a conveyor belt and two vacuum grippers, which serve as routing mechanisms in the system and move the packages from machine to machine. A multiprocessing bay performs multiple processing steps on the packages, while a high-bay storage system allows for automatic storage and retrieval.

For communication outside the system's Programmable Logic Controllers (PLCs) [14] MQTT is used. For each machine, the status and signals are published to a structured topic hierarchy. The data stream serves as a unidirectional interface from the physical system to the digital twin. In our setup, previously captured MQTT data was used to replay earlier system runs and simulate live behavior (see [26] for the used MQTT recorder and replayer library). This enabled stable, reproducible tests and decoupled the development of the twin from the availability of the physical system. Direct control over the physical system is not possible in this setup.

3 THE DIGITAL TWIN

The digital twin was created for two main purposes: (1) to monitor the physical system and (2) for testing different configurations of the physical system in what-if analyses.

Architecture. To run the digital twin in production, we used 4 docker containers, and we used an additional one in development (see Figure 1): a React frontend [19], a Spring Boot backend [31], a PostgreSQL database [29], a Mosquitto MQTT broker [8], and (in development) a MQTT replayer.

Monitoring. The *live observation* is implemented by listening to the MQTT topic the physical system uses to publish its state. In development, we replayed MQTT messages recorded from the physical system. An MQTT packet is handled in a first step by the MQTT broker and passed forward to the spring boot backend. The backend then processes the messages by storing the relevant information in the machine Data Transfer Object (DTO). For keeping track of the packages in the system, they were stored in a package DTO. These data objects are then sent to the frontend for visualization.

Simulation. For the *what-if analysis*, we simulate the physical system. The backbone of the simulation is a simulation graph. Every node in the graph represents a machine of the physical system, and every directed edge gives information about the package flow between the machines. The DT uses one thread to run the *simulation controller* and a thread for each *simulation machine*. The *simulation controller* orchestrates the threads of the different machines and uses the simulation graph to pass packages between them. The simulation machines have the same logic as the actual machines,

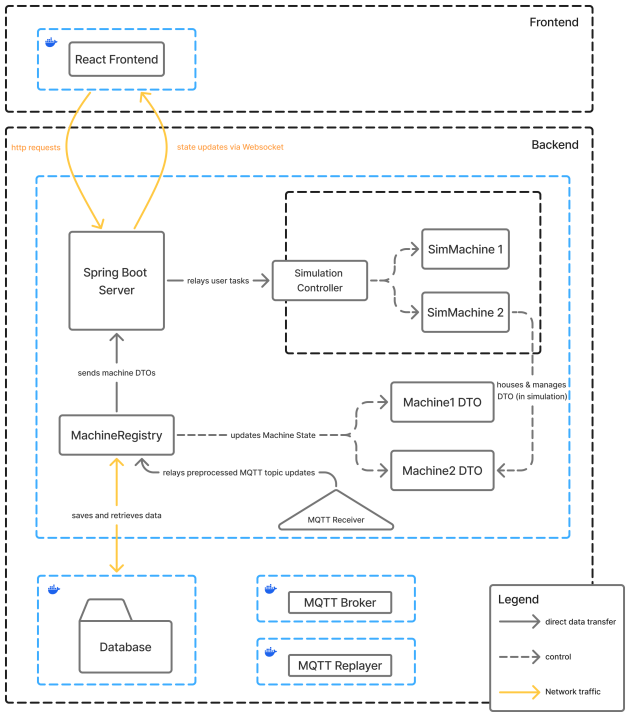

Figure 1: Software Architecture of the DT

and use the machine DTO to keep track of the machine specific data. Users can run the simulation in accelerated mode, with a maximum speedup up to ×500 real-time. Other simulation parameters include the *simulation duration, per-machine speedup multipliers, and the package generation rate.*

Virtual sensors. On top of the machine data coming from the sensors of the factory, we have implemented two "virtual sensors" for each machine: One which simulates *overheating* and one which estimates the *energy consumption*. With increased package generation rate and machine speedup multipliers, the machines are more likely to overheat and fail. When overheating, the machines shut down and restart after a cooling period, which impacts the overall system throughput. Energy consumption is modeled with per-machine power profiles to yield throughput-per-kWh metrics.

Setting Optimization. We have implemented *optimization* for three objective functions: (1) maximizing packages per hour, (2) minimizing energy per hour, and (3) minimizing energy per package. Each optimization problem considers the machine-specific speedup multipliers and the package generation rate as decision variables. Users may optionally fix individual parameters, in which case the optimization is then only performed over the remaining free variables. As the parameter space is too large for an exhaustive search, we use a k-nearest neighbors (kNN)-based optimization approach. First, we generate 1,000 parameter configurations by random sampling and evaluate them in a simulation. The resulting dataset is then used to estimate the objective function values for unexplored parameter configurations using kNN regression. Finally, we select the parameter configuration with the best estimated performance for each objective function and return it to the user.

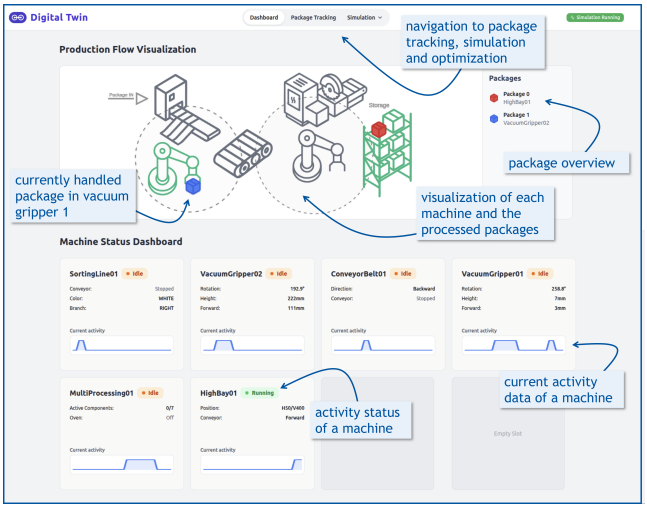

Figure 2: Screenshot of the digital twin dashboard with current status of the production line and machine sensor data (blue boxes as explanatory overlay for this paper)

Dashboard. Monitoring the current status and the simulation share the graphical user interface for machine states and package tracking (see Figure 2). On this main page, we show the specific package location (top) and the status of each machine together with the current sensor data (bottom). On further pages we provide an interface for setting the simulation up, and showing historical simulation results. Furthermore, users can set the optimization objectives, select which parameters should be fixed and see the optimization results.

4 ENGINEERING THE DIGITAL TWIN

Engineering the DT required identifying the behavior of the physical system from observable artifacts rather than from implementation knowledge. Since neither the PLC programs nor the internal control logic were available, the engineering process relied on the following three information sources: (1) a recorded execution of the production process as a video, (2) the corresponding MQTT message stream stored as CSV data, and (3) publicly available Fischertechnik documentation. The engineering workflow consists of five steps (see Figure 3):

Signal Interpretation. The first engineering task was to interpret the recorded MQTT data and identify which signals represented meaningful system behavior. Each MQTT message contains a topic identifying the machine and signal together with a payload carrying the observed value. Besides measurements, the dataset also contains communication metadata such as timestamps and delivery information (the duplicate (dup) and the retain flag, timing fields such as epoch arrival time and latency - time from publish to receive). Since only a subset of the available signals reflect machine behavior relevant for simulation, an initial data analysis was required to extract relevant artifacts.

Grey-Box Behavior Identification. The Fischertechnik documentation provided a structural understanding of the production line and the purpose of the individual machines. However, it neither described the internal control logic nor documented how machine states were represented in the MQTT data. Consequently, the DT was engineered following a grey-box modeling approach, combining documented domain knowledge with behaviour of the physical system inferred from runtime observations. An initial inspection focused on MQTT events because they appeared to represent process events. However, aligning these timestamps with the video recording revealed that the events did not reliably describe machine behavior. Instead, consistent execution patterns were observed in the boolean actuator ("act") signals. These repeated state transitions therefore became the primary source for identifying machine behavior. After identifying the relevant actuator signals, we were able to identify recurring transition patterns for every machine. We validated the patterns against the events in the video recording to make sure that observed signal changes correspond to physical actions of the production line. Based on this analysis, we derived a minimal set of machine attributes (e.g., isExecuting, colorPresent). These attributes form the state representation of the machines.

Simulation Graph Derivation. The identified machine behavior was then transformed into a simulation model. For every physical machine, we created a corresponding simulation class together with a finite set of abstract execution states. These states describe package processing as well as synchronization points between neighboring machines. Connecting these state machines through directed edges resulted in the simulation graph representing the package flow across the production line.

Simulation Implementation. The simulation engine executes every machine as an independent concurrent component coordinated by a central simulation controller. State transitions are evaluated on fixed simulation ticks to preserve the temporal characteristics of the production process while allowing accelerated execution. Additional virtual sensors were integrated into the simulation model to estimate machine temperature and energy consumption. These virtual sensors extend the observable behavior of the physical system and enable analyses that are not directly supported by the available measurements.

Optimization Implementation. Finally, the simulations were coupled with an optimization component that exploits the accelerated execution of the DT. Since the exhaustive exploration of the parameter space is computationally infeasible, the optimization combines random sampling with k-nearest-neighbor regression to estimate good parameter configurations. This enables users to evaluate alternative production settings with respect to throughput and energy-related objectives.

5 DISCUSSION & LESSONS LEARNED

Modeling effort dominated implementation effort. While implementing the software architecture mainly consisted of combining established technologies, identifying the production process from data required substantial effort. The most challenging task was identifying the semantics of observable messages and deriving an executable behavioral model from observations. This indicates that, in DT engineering, one of the main challenges lies in behavioral modeling [11, 18].

Grey-box modeling. Our experience indicates that grey-box modeling can be sufficient for implementing useful monitoring and

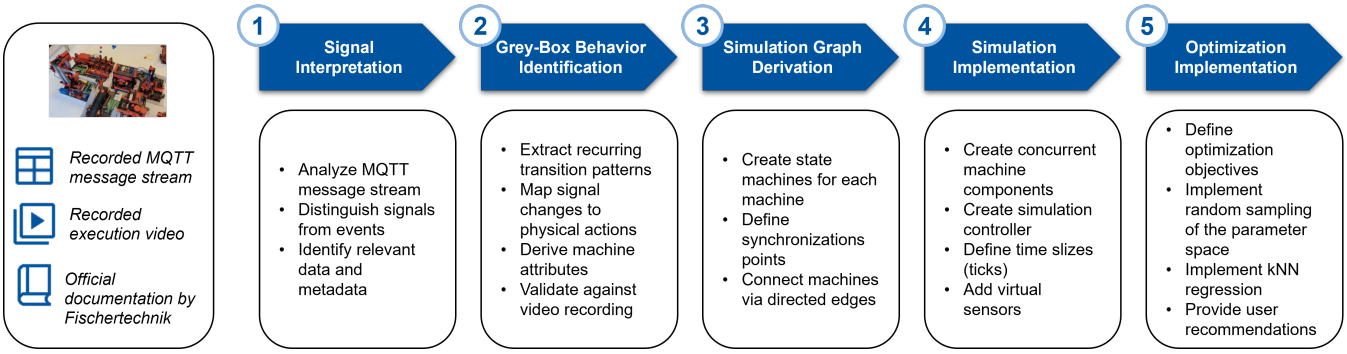

Figure 3: Engineering process starting from recorded data, process video and official documentation

simulation capabilities. Complete access to PLC logic is not necessarily required, provided that sufficiently informative runtime observations are available. However, the resulting model is limited to externally observable behavior and cannot explain or reproduce internal implementation decisions.

Virtual Sensors. Introducing virtual sensors significantly increased the value of the DT. While the physical production line does not provide measurements for machine temperature or energy consumption, both quantities could be estimated. This illustrates that DTs are not restricted to mirroring observable reality but can enrich the available information with additional system properties.

Simulation. Representing the production line as a directed graph separated the process topology from machine behavior. Consequently, extending the production line primarily requires adding machine nodes and package-flow edges without modifying the execution engine. This separation improves extensibility and maintainability. However, the multi-threaded simulation introduced nondeterminism at high speedup. A step-based (single-threaded) engine would improve reproducibility and allow higher speedups. In addition, a comparison between physical and simulated behaviour with quantitative measures is missing.

Optimization. The optimization can explore alternative operating points entirely within the virtual environment. The chosen parameter optimization approach based on k-nearest neighbors provides a computationally efficient solution for exploring the parameter space, but it remains a heuristic. In particular, it may fail to capture narrow optima or highly non-linear relationships between parameters and objectives. More advanced optimization techniques, such as adaptive sampling or probabilistic search methods, could improve the robustness of the results. In addition, the optimization results have to be validated against the real machine and runtime data in further work.

Bidirectional connection to the Physical System. The current implementation corresponds most closely to a digital shadow according to Kritzinger et al.[16] because communication is unidirectional. Nevertheless, the presented architecture intentionally separates data acquisition from simulation and optimization, allowing bidirectional interaction to be incorporated as soon as a real-life Fischertechnik production line is connected and write access to the physical system becomes available. Moreover, there is no difference in the system architecture if the DT is running with previously recorded or live MQTT data.

Usability for Education. From an educational perspective, the exemplar exposes students to the engineering lifecycle of a DT, including data interpretation, behavioral modeling, simulation, and optimization. This makes it suitable as a reusable case for teaching DT engineering that is near enough to industrial needs [15, 28].

Generalizability. Although the presented exemplar is based on a Fischertechnik production line, the overall engineering process is independent of the specific physical system. The sequence of interpreting runtime data, identifying behavior, deriving an executable model, and extending it with simulation and optimization capabilities can be transferred to other systems with comparable observable interfaces and process video recordings.

6 CONCLUSION

The developed prototype demonstrates that a DT can offer practical value under limited engineering conditions. Although direct access to the physical system's internal control logic was unavailable, recorded MQTT messages, video recordings, and documentation provided sufficient information to implement a first version with monitoring, simulation, and optimization capabilities. This shows that simple DTs can be meaningfully engineered from observable behavior and domain knowledge without requiring complete system transparency. A key insight of this work is that the true challenge lies not in implementing the software, but in understanding and modeling system behavior. Identifying machine states, packet transfers, and temporal behavior was essential for building an executable simulation model. This example underscores the importance of gray-box modeling, when combining documentation, runtime data, and real-world observations. Overall, this exemplar demonstrates how even a small-scale production system can capture many of the engineering challenges of industrial DTs. Future work can build on this foundation by adopting a model-driven engineering approach [3] and refining the simulation architecture and models using inspiration from [30].

ACKNOWLEDGMENTS

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Model-Based DevOps – 505496753. Website: https://mbdo.github.io.

REFERENCES

[1] Tarek Alskaif, Önder Babur, Francis Bordeleau, Loek Cleophas, Benoit Combemale, Joachim Denil, Ø ystein Haugen, Judith Michael, Phu Nguyen, Tiberiu Seceleanu, Mark van den Brand, and Hans Vangheluwe. 2025. Evolution at the Core of Digital Twin Engineering. In *Int. Conf. on Engineering Digital Twins (EDTconf'25)*. IEEE.

[2] Serkan Ayvaz and Koray Alpay. 2021. Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time. *Expert Syst. Appl.* 173 (2021), 114598. https://doi.org/10.1016/j.eswa.2021.114598

[3] Constantin Buschhaus, Arkadii Gerasimov, Jörg Christian Kirchhof, Judith Michael, Lukas Netz, Bernhard Rumpe, and Sebastian Stüber. 2024. Lessons learned from applying model-driven engineering in 5 domains: The success story of the MontiGem generator framework. *Journal Science of Computer Programming* 232 (Jan 2024), 103033. https://doi.org/10.1016/j.scico.2023.103033

[4] Arvid Butting, Sinem Konar, Bernhard Rumpe, and Andreas Wortmann. 2018. Teaching Model-based Systems Engineering for Industry 4.0: Student Challenges and Expectations. In *Proc. of MODELS 2018. Educators Symposium*.

[5] Benoit Combemale, Jean-Marc Jézéquel, Quentin Perez, Didier Vojtisek, Nico Jansen, Judith Michael, Florian Rademacher, Bernhard Rumpe, Andreas Wortmann, and Jingxi Zhang. 2023. Model-Based DevOps: Foundations and Challenges. In *Int. Conf. on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. ACM/IEEE, 429–433. https://doi.org/10.1109/MODELS-C59198.2023.00076

[6] Manuela Dalibor, Nico Jansen, Bernhard Rumpe, David Schmalzing, Louis Wachtmeister, Manuel Wimmer, and Andreas Wortmann. 2022. A cross-domain systematic mapping study on software engineering for Digital Twins. *Journal of Systems and Software (JSS)* 193 (November 2022).

[7] Digital Twin Consortium. 2022. Capabilities Periodic Table. https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/ Last accessed: 2026-06-20.

[8] Eclipse Foundation. 2026. *Eclipse Mosquitto*. https://mosquitto.org/

[9] Kevin Feichtinger, Kristof Meixner, Felix Rinker, István Koren, Holger Eichelberger, Tonja Heinemann, Jörg Holtmann, Marco Konersmann, Judith Michael, Eva-Maria Neumann, Jérôme Pfeiffer, Rick Rabiser, Matthias Riebisch, and Klaus Schmid. 2022. Industry Voices on Software Engineering Challenges in Cyber-Physical Production Systems Engineering. In *Int. Conf. on Emerging Technologies and Factory Automation (ETFA'22)*. IEEE.

[10] Fischertechnik GmbH. [n. d.]. *Industry and Universities*. https://www.fischertechnik.de/en/products/industry-and-universities Last accessed: 2026-06-20.

[11] Fabrizio Fornari, Ivan Compagnucci, Massimo Callisto De Donato, Yannis Bertrand, Harry H. Beyel, Emilio Carrión, Marco Franceschetti, Wolfgang Groher, Joscha Grüger, Emre Kilic, Agnes Koschmider, Francesco Leotta, Chiao-Yun Li, Giovani Lugaresi, Lukas Malburg, Juergen Mangler, Massimo Mecella, Oscar Pastor, Uwe Riss, Ronny Seiger, Estefania Serral, Victoria Torres, and Pedro Valderas. 2025. Digital Twins of Business Processes: A Research Manifesto. *Internet of Things* 30 (2025), 101477. https://doi.org/10.1016/j.iot.2024.101477

[12] R. Harrison, D. Vera, and Bilal Ahmad. 2021. A Connective Framework to Support the Lifecycle of Cyber–Physical Production Systems. *Proc. IEEE* 109 (2021), 568–581. https://doi.org/10.1109/jproc.2020.3046525

[13] Germán Herrera-Vidal, Jairo R. Coronado-Hernández, and Julien Maheut. 2025. Complexity Management Challenges in the Industry 4.0 Era: A Systematic Review in Production Systems. *Results in Engineering* (2025). https://doi.org/10.1016/j.rineng.2025.105329

[14] International Electrotechnical Commission. 2013. IEC 61131-3: Programmable Controllers – Part 3: Programming Languages. Edition 3.0.

[15] Fan Jiang, Feizhou Jiang, Yitong Xu, Yang Zhuang, Jun Qian, Hong Liu, Xiaoxiong Su, Senyang Chen, and Xue Chen. 2026. A six-dimensional digital twin model for flexible manufacturing systems in engineering education. *Journal of King Saud University - Computer and Information Sciences* 38, 5 (2026). https://doi.org/10.1007/s44443-026-00589-7

[16] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. 2018. Digital Twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline* 51, 11 (2018), 1016–1022.

[17] An Ngoc Lam, Gøran Brekke Svaland, Miguel Ángel Barcelona, Shane Keaveney, Wissam Mallouli, Luong Nguyen, Assia Belbachir, Xiang Ma, Akhilesh Kumar Srivastava, and Ahmed Nabil Belbachir. 2025. SINDIT: A Framework for Knowledge Graph-Based Digital Twins in Smart Manufacturing. In *Internet of Things. 7th IFIPIoT 2024 International IFIP WG 5.5 Workshops*, Gaëtan Rey, Jean-Yves Tigli, and Erwin Franquet (Eds.). Springer Nature Switzerland, Cham, 33–52.

[18] Jürgen Mangler, Ronny Seiger, Janik-Vasily Benzin, Joscha Grüger, Yusuf Kirikkayis, Florian Gallik, Lukas Malburg, Matthias Ehrendorfer, Yannis Bertrand, Marco Franceschetti, Barbara Weber, Stefanie Rinderle-Ma, Ralph Bergmann, Estefanía Serral Asensio, and Manfred Reichert. 2026. *From Internet of Things Data to Business Processes: Challenges and a Framework*. Springer Nature Switzerland, Cham, 25–61. https://doi.org/10.1007/978-3-031-90746-3_2

[19] Meta Platforms, Inc. 2026. *React*. https://react.dev/

[20] Judith Michael, Loek Cleophas, Steffen Zschaler, Tony Clark, Benoit Combemale, Thomas Godfrey, Djamel Eddine Khelladi, Vinay Kulkarni, Daniel Lehner, Bernhard Rumpe, Manuel Wimmer, Andreas Wortmann, Shaukat Ali, Balbir Barn, Ion Barosan, Nelly Bencomo, Francis Bordeleau, Georg Grossmann, Gabor Karsai, Oliver Kopp, Bernhard Mitschang, Paula Muñoz Ariza, Alfonso Pierantonio, Fiona A. C. Polack, Matthias Riebisch, Holger Schlingloff, Markus Stumptner, Antonio Vallecillo, Mark van den Brand, and Hans Vangheluwe. 2025. Model-Driven Engineering for Digital Twins: Opportunities and Challenges. *INCOSE Systems Engineering* 28, 5 (September 2025), 659–670. https://doi.org/10.1002/sys.21815

[21] Elisa Negri, Luca Fumagalli, and Marco Macchi. 2017. A Review of the Roles of Digital Twin in CPS-based Production Systems. *Procedia Manufacturing* 11 (2017), 939–948. https://doi.org/10.1016/j.promfg.2017.07.198 27th Int. Conf. on Flexible Automation and Intelligent Manufacturing.

[22] OASIS Message Queuing Telemetry Transport (MQTT) Technical Committee. 2019. MQTT Version 5.0. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. OASIS Standard, 7 March 2019.

[23] Ricardo Silva Peres, Andre Dionisio Rocha, Paulo Leitao, and Jose Barata. 2018. IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0. *Computers in Industry* 101 (2018), 138–146. https://doi.org/10.1016/j.compind.2018.07.004

[24] Jérôme Pfeiffer, Jingxi Zhang, Benoit Combemale, Judith Michael, Bernhard Rumpe, Manuel Wimmer, and Andreas Wortmann. 2025. Towards a Unifying Reference Model for Digital Twins of Cyber-Physical Systems. In *Int. Conf. on Emerging Technologies and Factory Automation (ETFA'25)*. IEEE.

[25] Phil Pöllmann, Falco Wolf, Jakob Titz, and Judith Michael. 2026. *UR-Programming-and-Software-Engineering/DT- Fischertechnik-UR: archived version v1*. https://doi.org/10.5281/zenodo.21028197

[26] RPDSWTK. 2025. *MQTT Recorder: Simple CLI Tool for Recording and Replaying MQTT Messages*. https://github.com/rpdswtk/mqtt_recorder GitHub repository, Last accessed: 2026-06-20.

[27] Roberto Sala, Fabiana Pirola, and Giuditta Pezzotta. 2023. On the development of the Digital Shadow of the Fischertechnik Training Factory Industry 4.0: an educational perspective. *Procedia Computer Science* 217 (2023), 640–649. https://doi.org/10.1016/j.procs.2022.12.260 4th International Conference on Industry 4.0 and Smart Manufacturing.

[28] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15, 4 (2019), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

[29] The PostgreSQL Global Development Group. 2026. *PostgreSQL*. https://www.postgresql.org/

[30] Hans L.M. Vangheluwe. 2000. DEVS as a common denominator for multi-formalism hybrid systems modelling. In *CACSD. Conference Proceedings. IEEE International Symposium on Computer-Aided Control System Design (Cat. No.00TH8537)*. 129–134. https://doi.org/10.1109/CACSD.2000.900199

[31] VMware. 2026. *Spring Boot*. https://spring.io/projects/spring-boot

[32] Birgit Vogel-Heuser, Alexander Fay, Ina Schaefer, and Matthias Tichy. 2015. Evolution of software in automated production systems: Challenges and research directions. (2015), 107–108. https://doi.org/10.1016/j.jss.2015.08.026