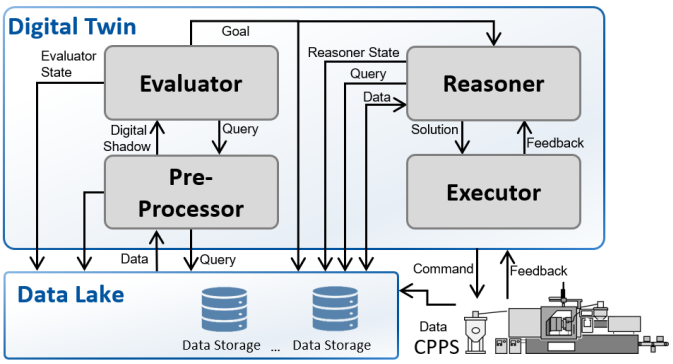

Fig. 1. Essential components of a self-adaptive digital twin architecture

which can be used to predict delays and deviations. Due to the highly complex interplay of information systems as they are typical for industrial settings involving various middleware solutions, process discovery is challenging. We therefore embed into our vision object-centric event logs [17] that document processes involving different types of objects in industrial processes, e.g., order-to-cash (including i.a. customer order, production, shipment, invoicing). These possibly involve various data sources spread over multiple information systems. We enable backward-looking process mining for digital twins by embedding it into their architecture using process models as input of their code generation. The architecture sets the stage for later forward-looking process mining that we describe in the outlook of this paper.

B. Model-Driven Digital Twins

We understand DTs as software systems that can actively represent, control, and/or optimize the behavior of an observed (cyber-physical) system. Such a DT of a system consists of a set of models of the system, a set of Digital Shadows (DSs), and provides a set of services to use the data and models purposefully with respect to the original system [18].

Figure 1 illustrates the software architecture of such a digital twin [19] and interfaces a Cyber-Physical Production System (CPPS) as well as to a data lake providing data from and about the CPPS as well as about the DT's context. The architecture comprises components with different responsibilities to realize a self-adaptive loop [20]: The CPPS produces data and persists it in a data lake, which is a data storage that consists of heterogeneous databases. Based on this data, the pre-processor queries data from the different databases and transforms the data into DSs [20]. A DS is a set of data traces and models enriched with contextual and purposeful that can origin from various sources and enables the DT to perform calculations and reasoning based on it [21]. Given the requested DSs, the evaluator compares the current state of the system with a list of event-condition-action (ECA) rules and creates goals sent to the reasoner to mitigate undesired system states. The reasoner comprises different AI subcomponents to fulfill the received goals (such as AI planners and case-based reasoning [20]). If the reasoner's subcomponents produce multiple possible solutions, it determines the best one using application-specific heuristics, e.g., fewest costs, most time efficient, lowest energy consumption. Finally, the executor translates the selected solution into executable commands and sends these to the CPPS (e.g., via OPA UA). Depending on feedback by the CPPS it can evaluate and alter the commands to contribute to a successful execution of the solution.

C. MontiGem

We have successfully used MontiGem [12], [22], a highly adaptable generator framework, to generate large parts of a digital twin cockpit [19] that aims to integrate humans. Within a MDD approach, we use a set of models as input for the generation process. The models used are UML class diagrams in the textual CD4A language [22] to define the data structure for a specific domain (domain model), data models for aggregated information to be presented in certain pages, Graphical User Interface (GUI) models in the textual GUI-DSL [23] language to describe the user interface and its relation to the data model, and Object Constraint Language (OCL) [24] expressions to constraint the data structure. The generator provides all necessities for a full communication pipe from the database over the backend up to the frontend. A domain expert outlines the system for which MontiGem is able to generate an almost ready-to-use application with few hand-written extensions needed by a software engineer.

In the context of this paper, we aim to extend the process so that the DT application is produced automatically in a single integrated generation step. For example, we can include an additional step into the automated generation process for the described DT architecture. This generation step includes a model in the architecture description language MontiArc [25], describing each component, its subcomponents, the component's interface and the communication to the other components. Other extension points are the possibility to integrate initial process models to the DT.

The generated DT architecture enables networked monitoring and control of CPPS. In the past, Operational Technologies (OT) were mostly used to control industrial equipment [26] but not linked to networks, unlike Information Technologies (IT) responsible for core business operations (e.g., ERP or PLC systems). Today, there is a trend toward OT-IT convergence, with more and more physical devices being integrated into networks. Incorporating process-oriented technologies enables new use cases, such as the process prediction described in the next section.

III. Towards Process Prediction for Model-Driven Digital Twins

We envision a model-driven DT architecture that incorporates process mining techniques and uses models at runtime. Our vision covers five steps: (1) generation of the DT, (2) configuration of the generated DT application, (3) initialization of DSs, (4) process discovery, (5) runtime analysis, and (6) user interaction.

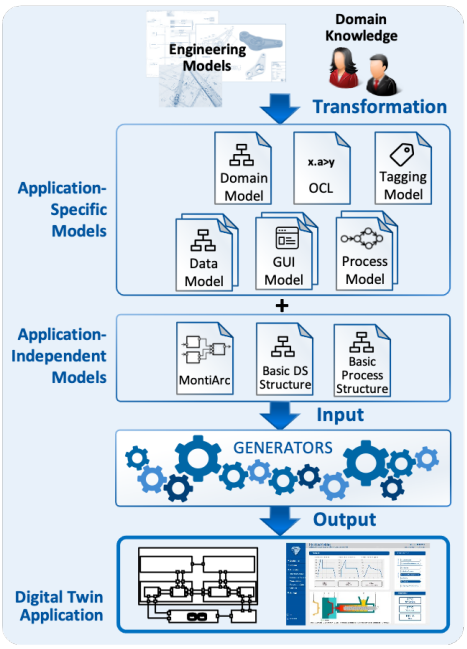

Fig. 2. Process and artifacts involved in the MDD of digital twins

A. Generation of the DT

Figure 2 highlights the model transformations involved in creating a digital twin and resulting in a specific digital twin application. These models can be fully or partly generated or created by hand: We can use information sources such as engineering models (AutomationML, CAD, Modellica, SysML, *etc.*) to extract information relevant for the domain model, ECA rules the DT needs to react upon, its services or constraints. Other important information sources are experts in these domains, who complete the missing models required to derive and operate the digital twins manually. We propose the following minimal set of application-specific models as crucial for the model-driven generation of digital twins: (1) Class diagram domain models describe the concepts and data types available to the digital twin and are the basis for digital shadows; (2) OCL models for validation of input data; (3) Tagging models for defining roles and rights; (4) Data models to show selected parts of the domain model in certain views of the GUI; (5) GUI models for describing the user interfaces; and (6) Process models that describe how the CPPS should behave. Additionally to these application-specific models, our approach relies on application-independent models: (1) A MontiArc architecture model describing the components of the digital twin and their relations; and (2) Two class diagrams describing the basic structure of all digital shadows and process models that the digital twins can operate upon. These models serve as the basis for generating applications of self-adaptive digital twins.

Example. *An engineer aims to create a DT to represent and optimize the behavior of an injection molding machine: she defines relevant concepts, including properties of sensors and actuators of the machine, manufacturing phases, and attributes, such as injection flow, nozzle temperature, or back pressure in the domain model, related constraints in OCL, and specifies the required digital shadows. She adds GUI models for desired user interfaces as well as process models of the intended behavior of the CPPS. Using the model-driven process, she reuses all application-independent models and generates the complete (but empty) digital application consisting of data structures, data transfer objects, self-adaptive loop, architecture implementation, and UI.*

B. Generated DT application

Figure 3 shows the generated digital twin architecture with its components and relevant data and model flows. The architecture comprises the components of the self-adaptive digital twin together with Process Mining components. The architecture has interfaces to the `CPPS`, the data lake including all relevant machine and process data and further relevant 3rd party applications.

Example. *In our running example, the CPPS is an injection molding machine which is connected to the data lake of the IoP, 3rd party applications include, e.g., the production planning system and the quality management system.*

The generated digital twin includes a database layer which provides the domain concepts within the data storage, as well as a data structure for the DS and process model concepts. The other components access relevant information either via interfaces from outside the DT or the interface of the database layer. The following subsections discuss the DT components and handled models in detail.

C. Configuration of the generated DT application

The digital twin application as generated is devoid of behavior. In this step, a DT user or domain expert identifies relevant concepts and behavior and adds corresponding models (e.g., ECA rules or cases for case-based reasoning [20]). Moreover, she defines the digital shadow structures required to reason about the CPPS's behavior and connects these to the behavior models. To this end, she specifies DS types that define, for a given purpose, which parts of the data are relevant, which models should be contained, the represented asset of the CPPS as well as the processes to obtain its data traces from the various data sources available in the data lake. The structure is based on the DS metamodel [21] and is extended to cover further domain-specific requirements.

The `Process-Aware Digital Twin Cockpit` provides a guided methodology: The domain expert chooses a purpose he creates the DS for. This can vary from human-readable text to a specific machine-processable optimization model. In a next step, he defines which data (accessed by a fully qualified address), with respective data points and meta data, should be chosen from the data lake, how the data should

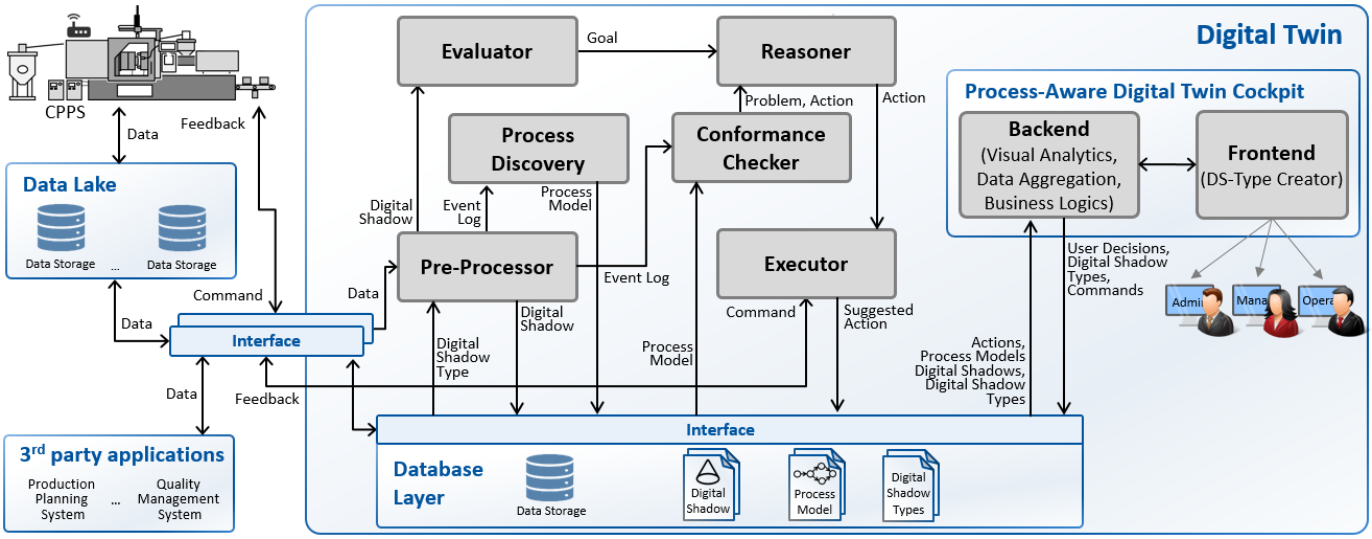

Fig. 3. Main System Architecture of a Digital Twin incorporating Process Mining Services and Models at runtime

be aggregated and what operations are performed on them. Such data points refer to data a DS needs to describe the system's behavior [21]. Data aggregations can be modeled in an expression language or a domain expert can even use a more specific language or provide code in a general purpose language. The domain expert then defines a DS type at design time which is then stored in the DT database. DS types can be modified or evolved at runtime by the user depending on the current stage of investigation and it is possible to define new ones at any point.

> **Example.** *The domain expert defines a DS type with the purpose of minimizing the rejection rate of a molded part. The DS stands for the injection molding production machine. Relevant data traces for this purpose are job histories including rejections rates for the last 6 months.*

D. Initialization of DSs

In a next step we have to create all digital shadows using the DS type definition. This happens once in the initialization step with a selection of DS types, but also continuously throughout the DT run time. The `Pre-Processor` component gets the DS type from the database layer and receives new data from the data lake through its interface. With the defined rules, the `Pre-Processor` decides which of the DSs for which purpose to address and computes data points cumulated in data traces according to the DS type definition. In this step, the DS type definition autonomously decides whether to update the current digital shadow or to archive the current one and create a new one with recent data. All digital shadows are then either updated or newly stored in the DT data base, so each of the components can access this information.

> **Example.** *Using the DS type definition, a DS for the purpose minimizing the rejection rate of a molded part is created. The* `Pre-Processor` *aggregates the rejection rates from every single job to one value per part. Once a week a new DS is created which aggregates again the rejection rates based on the new time slice.*

E. Process Discovery at Digital Twin Runtime

The DT receives event logs from the `Data Lake` via the `Database Layer` and `Pre-Processor` into the `Process Discovery` component. In the next step, it is possible to derive process models out of event logs by applying process discovery algorithms, such as Inductive Visual Miner [27] and Heuristic Miner [28], which are stored in the `Process Discovery` component. If the discovered processes are relevant for a specific purpose and specified as relevant models in the DS type, they are integrated within a purpose-driven DSs. Moreover, they can be visualized to the users via the DT cockpit for validation purposes.

> **Example.** *It is possible to identify the underlying process within the injection moulding machine using event logs with the measurements of temperatures within the injection moulding machine and the geolocation of each molded part. This process is included within the DS with the purpose to minimize the rejection rate for parts.*

F. Analysis at Digital Twin Runtime

At runtime, the DT supports automatic or user-triggered analyses. Within the system architecture of the self-adaptive DT, we consider two ways of runtime analysis using models: the DT self-adaptive loop and the `Conformance Checker` using process mining algorithms.

Within the DT adaptive loop [18], the digital twin's evaluator component uses ECA rules specifying undesired system states through data patterns over DS types and DS instances obtained from the data lake to determine whether action is necessary. If so, it instantiates the goals defined in the ECA

rules and passes these to the reasoner, which aims to find a solution to reach that goal by defining concrete actions. The actions are the input of the executor and either are translated into machine commands or passed on to a human operator.

The `Conformance Checker` supports analysis and prediction functionality with process mining and reasoning techniques. It takes event logs and intended process models both embedded in DSs as input and identifies deviations in the real-world processes. The recognized problem is transferred to the `Reasoner` component to be resolved by concrete actions. It can also store the proposed changes in the database layer to let the user decide to change the original process model.

> **Example.** *Using ECA rules derived from engineering models or specified by domain experts, the digital twin's evaluator component identifies that the nozzle throughput is too low. It instantiates the goal "increase nozzle throughput" and passes it to the reasoner, which leverages its AI subcomponents to produce a plan to heat the material a bit more and increase pressure to improve throughput. This plan is passed to the executor, which translates this into OPA UA commands and passes these to the machine. Concerning the process models, the `Conformance Checker` detects inconsistencies in discrete configurations, that resulted in process steps taking too long.*

G. User Interaction at Digital Twin Runtime

The generated DT provides a graphical interface for different user groups, also referred to as process-aware DT cockpit as a client-server architecture (visual analytics, data aggregation and business logics in the `Backend` component and visualizations in the `Frontend` component).

The users are able to define DS types via the DS type designer of the `Frontend`, view historical information in digital shadows and event logs, change created DSs and validate discovered process models. Operators can receive support for malfunctions in form of concrete actions.

> **Example.** *The GUI shows, e.g., the process with the temperatures in each process step of the injection moulding machine together with detected changes within the rejection rate. Moreover, concrete actions are shown to reduce the temperature in specific process steps of the machine to reduce the rejection rate.*

This vision allows us to create a self-adaptive DT which is able to handle and update models at runtime - both, automatically and by user intervention.

IV. Roadmap and Challenges

We propose a generative approach to combine model-driven with data-driven technologies in the realm of process mining. The digital twin architecture explicitly includes process mining components that leverage the massive amount of data available in the Internet of Things. This allows to gain new insights by combining process-level models with low-level event data. Combining comparative, object-centric, and forward-looking process mining will contribute to creating digital shadows that can be used to manage, control, and improve operational processes. The instantiation of the architecture allows to drill-down processes when a problem emerges, and even suggest actions to human operators. The goal is to continuously improve processes and respond to changes. We, thus, plan to include research on action-oriented process mining [29] in our architecture. The `Process-Aware DT Cockpit` keeps the human in the loop.

Further Model-Driven Software Engineering (MDSE) research for DTs. From the MDSE perspective, the realization of such an approach requires (1) the adaption of existing generators such as MontiGem to be able to handle the additional models in different Domain-Specific Languages (DSLs), (2) the integration of further generation steps, (3) to extend the runtime environment of the DT to be able to handle process models at runtime, and (4) to extend the runtime environment of the DT or provide APIs to include process mining services. Process mining is an integral part of our DT architecture and is included internally. The interfaces required to handle digital shadows and provide user support have to be tailored to specific needs within digital twins and for their user groups.

Deriving DTs from engineering models. Currently, digital twins are engineered ad-hoc, with models created from scratch, or other approaches only loosely based on the represented system. For a truly efficient engineering of DTs, we need to leverage knowledge contained in documents and original engineering models.

Automatically exploiting DT insights. Where digital twins are in place, they can provide tremendous insights about the operation of the particular observed system instance. Often, these insights can be applied to other instances of the same system type, similar systems, and future versions of these systems. Yet, we are lacking means to automatically exploit these insights sustainably. Automatically improving the engineering models of the observed system based on insights produced by its digital twins could be valuable angle to improve this. Together with the aforementioned derivation, this would pave the way for a model-driven DevOps with DTs [30].

Models at runtime in DTs. Szvetits and Zdun [31] have investigated how existing research literature uses models at runtime. In the context of our work, models at runtime are specifically used within self-adaptive systems [32]. Our approach for self-adapting DTs goes one step further to existing approaches and allows for both, autonomic control loops as well as human intervention. Together with process mining we thereby integrate manifold human control possibilities from low-level system events to business processes in IT.

Researching digital shadows. The definition of the structure of DSs required lengthy discussions with different use cases and stakeholders within the excellence cluster Internet of Production [21], with aspects still being researched. From a computer science perspective, the long-term operation of digital twins poses further challenges such as how to handle DS and DS-type evolvement,

We believe our approach helps to overcome the challenges of bi-directional synchronization between digital twins and their actual systems [33], as our approach is able to (1) take the raw runtime data provided by the CPPS via the *Data Lake* and extract the relevant information within the *Pre-Processor*, (2) evaluate and reason about the digital twin models within digital shadows to use this information extracted from the runtime data, and (3) provide concrete actions produced using these digital twin models to be fed into the system during execution in an automated way or via human operators.

ACKNOWLEDGEMENT

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2023 Internet of Production -390621612. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

REFERENCES

[1] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, 16th IFAC Symp. on Information Control Problems in Manufacturing INCOM'18.

[2] J. Reitz, M. Schluse, and J. Roßmann, "Industry 4.0 beyond the factory: An application to forestry," in *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter*. Springer, 2019, pp. 107–116.

[3] T. Ruohomäki, E. Airaksinen, P. Huuska, O. Kesäniemi, M. Martikka, and J. Suomisto, "Smart city platform enabling digital twin," in *Proc. of the Int. Conf. on Intelligent Systems (IS)*. IEEE, 2018, pp. 155–161.

[4] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli, "A microservice-based middleware for the digital factory," *Procedia Manufacturing*, vol. 11, pp. 931–938, 2017.

[5] H. Wang, M. Zhou, and B. Liu, "Tolerance allocation with simulation-based digital twin for CFRP-metal countersunk bolt joint," in *Int. Mechanical Engineering Congress and Exposition*, vol. 52019, 2018.

[6] N. K. Chakshu, J. Carson, I. Sazonov, and P. Nithiarasu, "A semi-active human digital twin model for detecting severity of carotid stenoses from head vibration-—A coupled computational mechanics and computer vision method," *International journal for numerical methods in biomedical engineering*, vol. 35, no. 5, p. e3180, 2019.

[7] L. Reinkemeyer, "Process Mining, RPA, BPM, and DTO," in *Process Mining in Action*, L. Reinkemeyer, Ed. Springer, 2020, pp. 41–44.

[8] C.-S. Shim, N.-S. Dang, S. Lon, and C.-H. Jeon, "Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model," *Structure and Infrastructure Engineering*, vol. 15, no. 10, pp. 1319–1332, 2019.

[9] H. Pargmann, D. Euhausen, and R. Faber, "Intelligent big data processing for wind farm monitoring and analysis based on cloud-technologies and digital twins: A quantitative approach," in *Int. Conf. on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2018.

[10] C. Mandolla, A. M. Petruzzelli, G. Percoco, and A. Urbinati, "Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry," *Computers in Industry*, vol. 109, pp. 134–152, 2019.

[11] M. S. Uysal, S. J. van Zelst, T. Brockhoff, A. F. Ghahfarokhi, M. Pourbafrani, R. Schumacher, S. Junglas, G. Schuh, and W. M. van der Aalst, "Process Mining for Production Processes in the Automotive Industry," in *Industry Forum at BPM'20*, 2020.

[12] K. Adam, J. Michael, L. Netz, B. Rumpe, and S. Varga, "Enterprise Information Systems in Academia and Practice: Lessons learned from a MBSE Project," in *40 Years EMISA (EMISA'19)*, vol. LNI 304, 2020.

[13] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, G. Fortino, A. Gal, U. Kannengiesser, F. Leotta, F. Mannhardt, A. Marrella, J. Mendling, A. Oberweis, M. Reichert, S. Rinderle-Ma, E. Serral, W. Song, J. Su, V. Torres, M. Weidlich, M. Weske, and L. Zhang, "The Internet of Things Meets Business Process Management: A Manifesto," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 6, no. 4, pp. 34–44, 2020.

[14] W. M. P. van der Aalst, T. Brockhoff, A. F. Ghahfarokhi, M. Pourbafrani, M. S. Uysal, and S. J. van Zelst, "Removing Operational Friction Using Process Mining: Challenges Provided by the Internet of Production (IoP)," in *Data Management Technologies and Applications*. Springer, 2021.

[15] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[16] M. Kerremans, "Market Guide for Process Mining," 2019, last access: 28.07.2021. [Online]. Available: https://www.gartner.com/en/documents/3939836/market-guide-for-process-mining

[17] W. M. P. van der Aalst, "Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data," in *Software Engineering and Formal Methods*, ser. LNCS, P. C. Ölveczky and G. Salaün, Eds. Springer, 2019, vol. 11724, pp. 3–25.

[18] P. Bibow, M. Dalibor, C. Hopmann, B. Mainz, B. Rumpe, D. Schmalzing, M. Schmitz, and A. Wortmann, "Model-Driven Development of a Digital Twin for Injection Molding," in *Int. Conf. on Advanced Information Systems Engineering (CAiSE'20)*, ser. LNCS, vol. 12127. Springer, 2020, pp. 85–100.

[19] M. Dalibor, J. Michael, B. Rumpe, S. Varga, and A. Wortmann, "Towards a Model-Driven Architecture for Interactive Digital Twin Cockpits," in *Conceptual Modeling*. Springer, 2020, pp. 377–387.

[20] T. Bolender, G. Bürvenich, M. Dalibor, B. Rumpe, and A. Wortmann, "Self-Adaptive Manufacturing with Digital Twins," in *2021 Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2021, pp. 156–166.

[21] F. Becker, P. Bibow, M. Dalibor, A. Gannouni, V. Hahn, C. Hopmann, M. Jarke, I. Koren, M. Kröger, J. Lipp, J. Maibaum, J. Michael, B. Rumpe, P. Sapel, N. Schäfer, G. J. Schmitz, G. Schuh, and A. Wortmann, "A Conceptual Model for Digital Shadows in Industry and its Application," in *Int. Conf. on Conceptual Modeling (ER'21)*. Springer, 2021.

[22] A. Gerasimov, J. Michael, L. Netz, B. Rumpe, and S. Varga, "Continuous Transition from Model-Driven Prototype to Full-Size Real-World Enterprise Information Systems," in *25th Am. Conf. on Information Systems (AMCIS 2020)*. AIS, 2020.

[23] A. Gerasimov, J. Michael, L. Netz, and B. Rumpe, "Agile Generator-Based GUI Modeling for Information Systems," in *Modelling to Program (M2P)*. Springer, 2021, pp. 113–126.

[24] M. Richters and M. Gogolla, *OCL: Syntax, Semantics, and Tools*. Springer, 2002, pp. 42–68.

[25] R. Heim, O. Kautz, J. O. Ringert, B. Rumpe, and A. Wortmann, "Retrofitting Controlled Dynamic Reconfiguration into the Architecture Description Language MontiArcAutomaton," in *Europ. Conf. on Software Architecture - (ECSA'16)*, ser. LNCS, vol. 9839. Springer, 2016.

[26] A. Hahn, "Operational technology and information technology in industrial control systems," in *Cyber-security of SCADA and other industrial control systems*. Springer, 2016, pp. 51–68.

[27] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Process and deviation exploration with inductive visual miner," in *BPM Demo Sessions 2014 at Int. Conf. on Business Process Management (BPM'14)*, ser. CEUR, vol. 1295. CEUR-WS.org, 2014, p. 46.

[28] A. Weijters, W. van der Aalst, and A. Medeiros, "Process Mining with the Heuristics Miner-algorithm," BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[29] G. Park and W. M. P. van der Aalst, "A General Framework for Action-Oriented Process Mining," in *Business Process Management Workshops*, ser. LNBIP, A. Del Río Ortega, H. Leopold, and F. M. Santoro, Eds. Springer, 2020, vol. 397, pp. 206–218.

[30] B. Combemale and M. Wimmer, "Towards a model-based devops for cyber-physical systems," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment (DEVOPS'19)*, ser. LNCS, vol. 12055. Springer, 2020, pp. 84–94.

[31] M. Szvetits and U. Zdun, "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime," *Software & Systems Modeling*, vol. 15, no. 1, pp. 31–69, 2016.

[32] A. Bennaceur, R. France, G. Tamburrelli, T. Vogel, P. J. Mosterman, W. Cazzola, F. M. Costa, A. Pierantonio, M. Tichy, M. Akşit *et al.*, "Mechanisms for leveraging models at runtime in self-adaptive software," in *Models@ run. time*. Springer, 2014, pp. 19–46.

[33] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, and M. Wimmer, "Towards Model-Driven Digital Twin Engineering: Current Opportunities and Future Challenges," in *Systems Modelling and Management*. Springer, 2020, pp. 43–54.