Software and Systems Modeling
https://doi.org/10.1007/s10270-025-01264-7

THEME SECTION PAPER

Model-driven engineering for digital twins: a systematic mapping study

Daniel Lehner[1] · Jingxi Zhang[2] · Jérôme Pfeiffer[2] · Sabine Sint[1] · Ann-Kathrin Splettstößer[2] · Manuel Wimmer[1] · Andreas Wortmann[2]

Received: 16 September 2023 / Revised: 11 November 2024 / Accepted: 21 December 2024

Abstract
Digital twins (DTs) are proliferating in a multitude of domains, including agriculture, automotive, avionics, logistics, manufacturing, medicine, smart homes, etc. As domain experts and software experts both have to contribute to the engineering of effective DTs, several model-driven engineering (MDE) approaches have been recently proposed to ease the design, development, and operation of DTs. However, the diversity of domains in which MDE is currently applied to DTs, as well as the diverse landscape of DTs and MDE applications to DTs, makes it challenging for researchers and practitioners to get an overview of what techniques and artifacts are already applied in this context. In this paper, we shed light on the aforementioned aspects by performing a systematic mapping study on the application of MDE automation techniques, i.e., model-to-model transformation, code generation, and model interpretation, in the context of DTs as well as on the characteristics of DTs including the twinned systems to which these techniques are applied in different domains. We systematically retrieved a set of 189 unique publications, of which 66 were selected for further investigation in this paper. Our results indicate that the distribution of employed MDE techniques (136 applications of automation techniques) is balanced between the different techniques, but there are significant variations for different DT types. With respect to the different domains, we found that even though applications are available in many domains, a small number of domains currently dominate applications of MDE to DTs, i.e., more than half of included papers are in the manufacturing and transportation domains.

Keywords Model-driven engineering · Digital twin · Digital twin engineering · Digital twin modeling languages · Literature review

1 Introduction

Digital twins (DTs) are increasingly used to enable applications such as anomaly detection [74], virtual experimentation [4], or reactive planning [S22] for cyber-physical systems (CPSs) [3] in various domains, e.g., manufacturing [67], agriculture [58], or energy management [75] to mention just a few. However, as DTs comprise of multiple and diverse software components, which highly vary based on specific use cases, they are considered as complex software systems [19]. Thus, designing, developing, and operating DTs require considerable development effort [39] as well as

Communicated by T. Clark, S. Zschaler, V. Kulkarni, and D. E. Khelladi.

✉ Daniel Lehner
daniel.lehner@jku.at

Jingxi Zhang
jingxi.zhang@isw.uni-stuttgart.de

Jérôme Pfeiffer
jerome.pfeiffer@isw.uni-stuttgart.de

Sabine Sint
sabine.sint@jku.at

Ann-Kathrin Splettstößer
ann-kathrin.splettstoesser@isw.uni-stuttgart.de

Manuel Wimmer
manuel.wimmer@jku.at

Andreas Wortmann
andreas.wortmann@isw.uni-stuttgart.de

[1] CDL-MINT, Institute of Business Informatics - Software Engineering, Johannes Kepler University Linz, Science Park 3, 4040 Linz, Austria

[2] Institute for Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Seidenstr. 36, 70174 Stuttgart, Germany

Published online: 24 March 2025

communication and cooperation between software experts and domain experts.

Model-driven engineering (MDE) [11] is a prominent approach to handle complexity and facilitate communication by leveraging the abstraction power of models. These models are designed to be human-understandable symbolic models that simultaneously provide machine-readable and processable representations of real-world entities. By applying MDE, these models can be (i) transformed to other models (e.g., Business Process Model and Notation (BPMN) models to Digital Twin Definition Language (DTDL) models [S60]), (ii) used as a basis for code generation (e.g., BPMN models are translated to Java code [S60]), or (iii) interpreted by software systems, e.g., a simulation controller model is directly interpreted to read simulation results and send commands from/to a running Unity simulation [S53]). More precisely, we use the following characterization of MDE throughout this paper.

Model-Driven Engineering (MDE): In this study, we refer to MDE as the process of using symbolic models to drive certain activities by some degree of automation. Thus, we focus on MDE techniques which enable the automated processing of models. We differentiate in this paper between the following techniques: (i) model-to-model transformation, which translates a model into another model, (ii) code generation, which translates a model into code, and (iii) model interpretation, where models are directly processed by software systems.

Several researchers have already discussed the potential benefits of using MDE in the context of DTs [10, 18], and recent MDE applications already provide evidence of the expected benefits [36]. Since DTs are complex software systems, as already mentioned before, the application of MDE in this context may also be referred to as model-driven software engineering (MDSE) [11]. As MDE is the general term that also includes MDSE, we decided to use the term MDE in the remainder of this paper.

To further advance the current state of the art, the research community requires an overview and classification of existing approaches to (i) understand which application contexts of automation techniques yield most promising results that can be built upon, (ii) how different characteristics of DTs influence this application context, and (iii) which application domains currently see the highest level of adoption of MDE for DTs, and which domains are lacking such activities. We expect that such a documentation of the current body of knowledge can be helpful for the adoption of MDE in practical settings for different DT types and domains, and align future research efforts in this area. Although the application of MDE to DTs has been extensively investigated in recent years, such an overview and comparison for general orientation is still missing.

In order to tackle this shortcoming, in this paper, we shed light on these aspects by performing a systematic mapping study [53] on existing applications of MDE automation techniques to DTs. In particular, we performed a keyword-based search in multiple literature databases. Based on the results obtained from the included papers, we classify applications of MDE for DTs and derive challenges that can be tackled by the research community in the future by proposing concrete action points.

The remainder of this paper is structured as follows.

Section 2 gives an overview of related work and further motivates this study. Subsequently, Sect. 3 gives a detailed description of the applied research method, i.e., systematic mapping study. Section 4 elaborates on the results obtained from using this method, and Sect. 5 discusses these results in the context of the stated research questions. Section 6 presents threats to validity of the obtained results. Section 7 identifies the challenges for advancing the application of MDE to DTs and proposes action points to address them. Finally, Sect. 8 concludes the paper.

2 Related work

As described in the previous section, our research interest focuses on identifying and characterizing existing MDE applications to DTs. Given the diverse landscape of DTs [66, 73], it is currently challenging for researchers and practitioners to achieve a comprehensive overview of existing MDE techniques and accompanying modeling artifacts. This makes it essential to investigate not only what has been done for single applications, but also how these applications differ for the diverse DT types and domains.

In the following, we present existing investigations of related aspects, i.e., applications of MDE to different fields (potentially outside the area of DTs), the usage of different DT types, and widespread application domains of DTs. We also highlight gaps in these existing surveys to motivate the need for the survey presented in this paper and summarize the found differences of existing surveys in Table 1.

2.1 Reviews on MDE applications

In the last decades, a plethora of surveys on the application of MDE in different domains, from traceability [60] over robotics [1, 16] to cyber-physical systems [49] and Industry 4.0 [73], have been published. Moreover, surveys are available on investigating approaches following the models@run.time paradigm [7, 23], which can be seen also related to DTs. Even though the term DT has been present since its initial introduction in 2005 [30] for a considerable amount of time, to the best of our knowledge, there is still no dedicated survey on the applications of MDE to DTs.

Table 1 Related work—comparison table

Survey	Research focus	MDE-specific Automation techniques	Modeling Artifacts	DT specific	Considered Years[1]
[42]	DTs for product design and development			√	≤ 2021
[73]	Modeling languages in Industry 4.0		√		≤ 2018
[59]	Enabling technologies for DTs		√	√	≤ 2021
[46]	Enabling technologies and tools for DTs			√	≤ 2022
[54]	Modeling languages of DT platforms		√	√	≤ 2022
[41]	Applications of 3D modeling for DTs		√	√	≤ 2022
[19]	Software engineering for DTs		√	√	≤ 2019
[62]	Potentials of DTs in different domains			√	≤ 2022
[66]	DT modeling		√	√	≤ 2021
[7]	Applications of runtime models	√	√		≤ 2017
[23]	Runtime models and IoT	√	√		≤ 2021
[24]	DTs for predictive maintenance			√	≤ 2022
[49]	MDE for CPS	√	√		≤ 2021
[16]	MDE for mobile robotic systems	√	√		≤ 2022
[1]	MDE for robotics	√	√		≤ 2021
[60]	Traceability provided by MDE tooling	√	√		≤ 2012
[38]	DTs in manufacturing			√	≤ 2018
[33]	DT characterizations			√	≤ 2020
[35]	DTs in construction			√	≤ 2019
[28]	Trends of DTs in the energy domain			√	≤ 2022
[17]	DTs in manufacturing			√	≤ 2019
[51]	DTs in CPS-based production systems			√	≤ 2017
[57]	DTs for zero defect manufacturing			√	≤ 2022
[44]	Process modeling, monitoring, and control in smart manufacturing		√		≤ 2018
This study	MDE applications for DTs	√	√	√	≤ 2023

[1]For papers that did not explicitly mention the considered years in the study setup, we used the publication years of the papers as a generous upper bound

Some studies are surveying technologies [46, 59], modeling languages [54, 66], or 3D modeling tools [41] that are available for engineering DTs. However, this existing work focuses solely on the presentation of technologies or models, with no investigation of how models can or should be processed using automation techniques in the context of DTs.

Various surveys also focus on particular life cycle phases of the underlying system, e.g., the design, production, usage, maintenance, and end-of-life phase of a product [42]. These studies show that most research is done in the production phases of systems. Other examples often focus on particular life cycle phases, e.g., maintenance [24] or particular technologies applied in these life cycle phases, such as predictive maintenance [21]. Although these studies investigate different life cycle phases of the underlying system, the DT and the used automation techniques in the respective life cycle phases have not been investigated yet. Thus, we address this limitation in our mapping study.

In summary, while the current body of knowledge offers comprehensive surveys of studies exploring (i) various application domains for MDE and (ii) existing technologies and modeling capabilities used in the context of DTs, the investigation of automation techniques to DTs is still unexplored. We address this shortcoming in our mapping study by investigating existing applications of automation techniques to DTs. For instance, this involves identifying the application frequencies of the different automation techniques, or the source and target artifacts used, the employed research type according to [52].

2.2 Characterizations of DTs

Several studies have already investigated the different characterizations of DTs and their usage. A characterization that is frequently cited is the differentiation between digital twins, digital shadows, and digital models by Kritzinger et al. [38], which was also extended with, e.g., the presence of software representations [47, 59, 67], the modeling of different aspects of the Physical Twin (PT) [66], or different purposes such as model construction, assembly, fusion, and verification [66]. Jones et al. [33] give a broader overview of 19 DT characteristics in total, ranging from the environment and connection to the fidelity of the physical and virtual counterpart in a DT system [33]. To augment the body of knowledge and in comparison with existing work, we aim to understand which automation techniques are used for which types of DTs.

One of the most comprehensive overviews of DT characteristics used in different domains is given in a cross-domain mapping study by Dalibor et al. [19], in which the authors investigate twinning targets, system life cycle phases, and twin life cycle phases of more than 350 publications on DT applications. However, compared to our work, this mapping study has a broader view on software engineering for digital twins and does not investigate automation techniques for DTs.

To sum up, whereas these existing works focus on characterizing existing DTs, our study investigates to characterize the application of automation techniques for DTs that represent different twinning targets in the different life cycle phases.

2.3 Investigations of DT application domains

To understand the range of domains and to identify the most common domains that use DTs, Singh et al. [62] reviewed publications from different literature databases and identified manufacturing, energy, and construction as the predominant domains based on the number of identified publications. These results are also confirmed by Dalibor et al. [19], who identified manufacturing as the predominant application domain of DTs, followed by information, energy, and construction. Several publications investigate the use of DTs in these particular domains identified above. Whereas domains such as construction [35] or energy [28] offer some selected reviews, different researchers have already extensively investigated manufacturing. For instance, Kritzinger et al. [38], as already discussed before, derived one of the most common definitions of DTs from examining existing works in manufacturing in 2018, and another work classified DT applications in manufacturing [17]. Several reviews also focus on specific concepts related to manufacturing, such as Industry 4.0 [22], digital factories [48], cyber-physical production systems [51], zero defect manufacturing [57], and manufacturing process modeling [44].

Even though this existing corpus of reviews gives an extensive overview of the application domains of DTs and specific insights into particular domains, in particular for the manufacturing domain, there is still no overview on whether and how MDE automation techniques are applied in these different domains. Thus, we aim to close this gap by investigating the application context of automation techniques for different domains and how this application context varies for the different DT kinds.

2.4 Summary

In the following, we summarize the differences between our work and existing surveys based on the information provided in Table 1. In the context of DTs, several surveys investigate enabling technologies for DTs [46, 59], existing DT characterizations [33, 38, 66], or potentials of applying DTs in different domains such as manufacturing [17, 51, 57], construction [35], or energy [28]. All these surveys neglect the use of modeling artifacts and MDE automation techniques in this context.

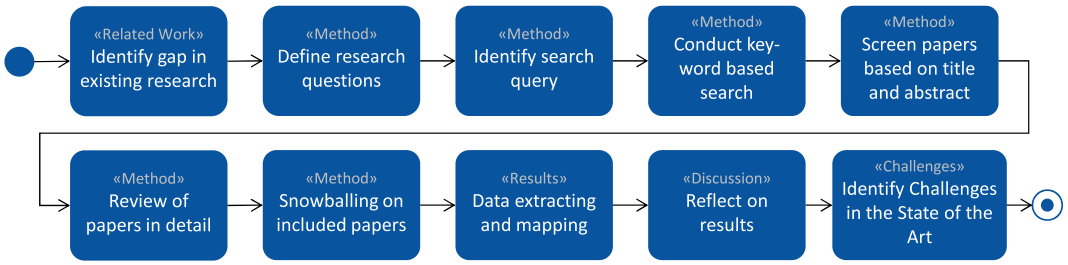


Fig. 1 Systematic mapping process (based on [53]). The blue boxes depict the performed tasks. The stereotypes in the boxes refer to the respective section of this paper in which we describe the results of the respective task

Additionally, five surveys investigate modeling languages used for DTs [19, 41, 54, 59, 66]. However, their scope is more narrow than we use in this paper, focusing on particular DT platforms or 3D modeling, or more broad, describing modeling languages used in the general field of software engineering, without the requirement of an automation technique being applied. Our work targets publications that use MDE automation techniques in the context of DTs. Other papers that also explore the application of such automation techniques are in domains that are related to DTs, i.e., runtime models [7, 23], cyber-physical systems [49], or robotics [1, 16]. One survey focuses particularly on modeling languages used in Industry 4.0 [73]. However, none of these mentioned studies puts an explicit focus on DTs for their discussions, which is unique to the study at hand.

Finally, all related surveys consider publications years before 2023. As digital twins are currently gaining traction in academia (as we will see, the data presented later in this study shows that most included papers were published in 2023), we expect our survey to update the insights presented in previous work to a large extent.

3 Method

In the following, we describe the method applied to obtain the results of our mapping study, following established guidelines [37, 53]. The process for conducting our study is shown as an activity diagram in Fig. 1.

3.1 Research questions

In this subsection, we define our Research Questions (RQs) to specify the review scope of our mapping study. In our work, we investigate the application of MDE automation techniques (i.e., the solution space) to DTs (i.e., the problem space) following a schema which is illustrated in Fig. 2.

With respect to the solution space, each automation technique uses a set of artifacts, i.e., code generation uses a source model to generate target code, a model-to-model transformation transforms a source model into a target model, and for model interpretation, an interpretation engine interprets a source model. Each paper found in our study applies one or more automation techniques and evaluates this application. Based on the extensiveness of this evaluation, we can classify the respective research type of a paper into solution proposal, validation research, or evaluation research, according to Petersen et al. [52].

In each paper, automation techniques are applied to a particular DT, which we regard as our problem space. This DT is in a particular life cycle phase (i.e., Twin Life Cycle Phase (TLCP)). According to Dalibor et al. [19], the possible life cycle phases of DTs are as-manufactured, as-designed, and as-operated. As-manufactured DTs describe the manufacturing process of the represented system, whereas as-designed twins describe the manufactured system in its design phase, and as-operated DTs characterize its usage during operation. The DT can also represent different kinds of systems (i.e., Twinning Targets (TTs)). Dalibor et al. [19] propose a set of categories of twinning targets that we use in this paper, i.e., individual systems, system-of-systems, processes, products, biological beings, and other counterparts. This twinned system also has different life cycle phases (i.e., SLCP), which advance orthogonally the twin life cycle phases, e.g., an as-designed DT can also be used during system operation.

As already described in Sect. 1, we aim to develop an overview of the state of the art in peer-reviewed literature, characterizing (i) the application of automation techniques, (ii) types of DTs, and (iii) domains to which these automation techniques are applied. With our RQs, we aim to investigate the solution space (i.e., the application of automation techniques, cf. RQ1) and problem space (i.e., the DT to which automation techniques are applied, cf. RQ2, as well as the domain in which these techniques are applied, cf. RQ3).

Concerning the solution space (RQ1), we investigate the particular automation technique and the artifacts used by this technique for each application of automation techniques identified in the included papers. We also investigate the research type of the work resulting from the application of automation techniques in a particular publication. As we are also interested in the DT to which automation techniques are applied (RQ2), we investigate different DT

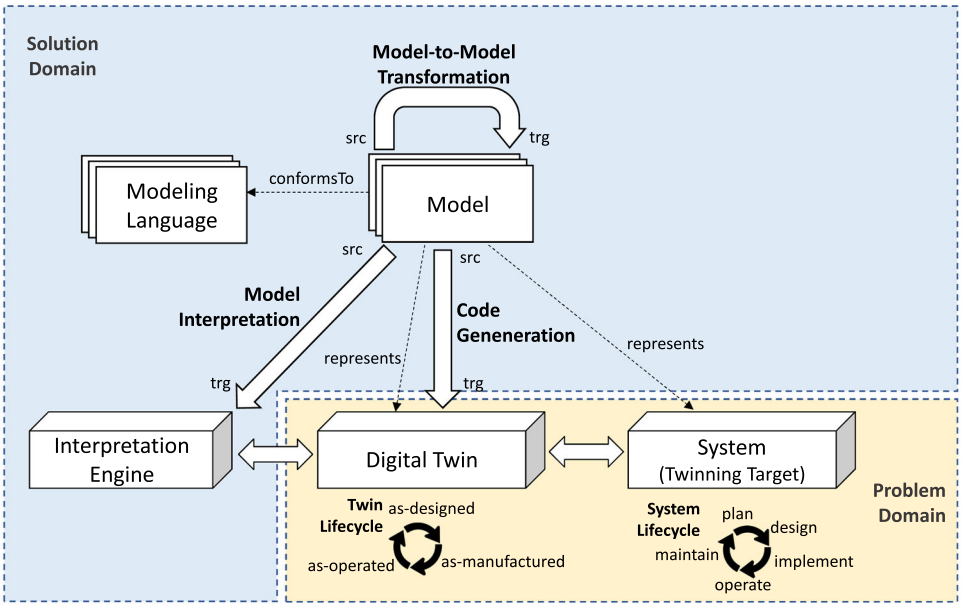


Fig. 2 Schema overview of MDE and DT concepts and their connections as investigated in our survey

types based on the twinned system and its SLCP (RQ2.1), as well as the TLCP (RQ2.2) and the influence of DT types on the frequency of automation techniques and the resulting research type (RQ2.3). Finally, we investigate the domain in which automation techniques are applied to DTs (RQ3).

RQ0: What are the bibliometric key facts of peer-reviewed literature documenting applications of MDE to DTs?

- RQ0.1: In which years are they published?
- RQ0.2: In which types of venues are they published?

RQ1: How and how often are automation techniques applied to DTs in peer-reviewed literature?

- RQ1.1: How often are the different automation techniques applied in the context of DTs?
- RQ1.2: Which modeling artifacts and software artifacts are used by these automation techniques?
- RQ1.3: Which combinations of input and output artifacts are used by these automation techniques?
- RQ1.4: What is the research type of the studies that apply these automation techniques?

RQ2: To which types of DTs are automation techniques applied in peer-reviewed literature?

- RQ2.1: Which TT does the DT represent, in which SLCP of the TT are automation techniques applied, and what is the TLCP of DTs to which automation techniques are applied?
- RQ2.2: How does the application of automation techniques (identified with RQ1) vary for different DT types?

RQ3: In which domains are automation techniques applied to DTs in peer-reviewed literature?

- RQ3.1: For which domains are automation techniques applied to DTs?
- RQ3.2: How does the application of automation techniques (identified with RQ1) vary for the identified domains?
- RQ3.3: How does the DT type (identified with RQ2) vary for the identified domains?

3.2 Inclusion and exclusion criteria

For our initial search, we included peer-reviewed publications published in journals, conferences, or workshops written in English. In addition, we used the following inclusion criteria:

- `Automated model processing`: Based on the definition of MDE provided in Sect. 1, we require papers to present approaches which automatically process symbolic models using model-to-model transformation(i.e., transform one model to another model), code generation (i.e., transform a model to code), or model interpretation (i.e., interpret the model for different purposes such as simulation). More precisely, this means that we require that the presented approach provides some automation component that supports the processing of models. Such automation components may be written in languages

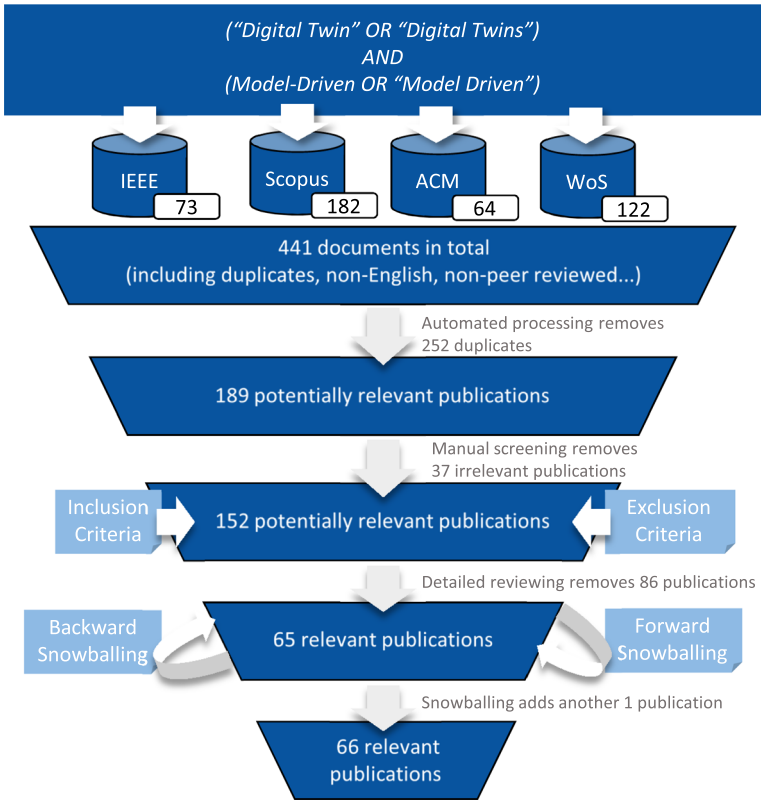


Fig. 3 Iterations of our literature search, based on keywords in the title, abstract, keywords, or the title of the venue at which the paper was published, and processing steps to identify relevant publications.

such as ATL [34] for model-to-model transformation or Xtend [8] for code generation. Still, they can also be written directly in general-purpose programming languages such as Java. For model interpretation, example automation components can also range from sophisticated simulation tools to Java-based components that access model information through some kind of API. To sum up, we do not require particular technologies which are used to process the models, but the models have to be automatically processed to some degree.

- `DT as application context`: For a paper to be included, we require that it documents an application of an automation technique in the context of DTs. We retrieved this information from the paper by finding either DTs mentioned as part of the presented approach (e.g., DT is one component of a figure depicting the architecture of the approach) or that the approach is used in the context of DTs as part of the evaluation. Publications that mentioned the term DTs somewhere in the abstract or as related work, but DTs have not been used as part of the presented approach or evaluation, were omitted.

On top of these aspects required for inclusion, we also made use of the following exclusion criteria:

- `Philosophical, Opinion, or Experience paper`: According to the classification proposed in [52], we only included solution proposal, validation research, and evaluation research papers, i.e., works that provide a concrete solution that actually applies an automation technique. This means that we excluded philosophical, opinion, and experience papers. Most papers that were excluded for this reason contained a collection of challenges or opportunities for (i) applying MDE to DTs, or (ii) applying IoT or DTs to different domains, such as process prediction or sustainability.
- `Secondary Study`: We excluded papers that solely presented reviews of existing literature for our review. Although these studies give a good overview of existing work, they do not provide novel applications of automation techniques (as primary studies would do) that could be studied in our investigation. However, we present these studies as related work in Sect. 2.

3.3 Search strategy and paper selection

Figure 3 gives an overview of the overall search strategy performed to obtain the dataset for this study. The respective data to reproduce each step is available online.[1]

1. Keyword-based Search: Initially, we performed a keyword-based search on November 19, 2024, in differ-

[1] Zenodo Repository collecting all data for this study: https://doi.org/10.5281/zenodo.14569169

ent computer science literature databases, namely Scopus, Web of Science (WoS), IEEE Xplore, and ACM digital library. We chose these databases as they include various publishers, such as Springer and Elsevier, and thus cover the scientific landscape very well. We searched these databases for papers containing the search term *("Digital Twin" OR "Digital Twins") AND (Model-Driven OR "Model Driven")* in either title, abstract, keywords, or title of the venue at which the paper was published similarly to related studies in model-driven engineering [19, 68]. This restriction to the aforementioned metadata ensures that the found publications focus on the application of MDE for DTs. This keyword-based search resulted in an initial corpus of 441 papers.

2. Screening: After removing 252 duplicates on this collected dataset, we manually screened the remaining 189 papers based on titles and abstracts. In this manual screening step, each paper was processed by at least two authors. If all authors independently decided to exclude a paper and noted the same reason for exclusion, the paper was excluded. If at least one of the authors argued for including the paper or stated different reasons for exclusion, the paper was forwarded to the next step. Following this manual screening procedure, 37 papers were excluded, leaving 152 papers for detailed reviewing.

3. Detailed Reviewing: In this step, each of the 152 remaining papers was carefully read by three different authors. Suppose at least two of these authors decided that after reading the paper, one of the exclusion criteria applied to it, it was marked as *to be excluded*, including the proposed reason for exclusion. Each marked paper was then again reviewed by two other authors. If at least one of these authors disagreed with the exclusion, the paper was discussed again among all authors. If, after this discussion, all authors agreed on an exclusion criterion for this paper, it was excluded.

4. Snowballing: In the last step, we performed forward and backward snowballing [71] on the 65 papers included after the detailed reviewing phase. With this step, we expected to identify further papers we did not find during the initial keyword-based search. After (i) removing duplicates and papers we already found in our initial search and (ii) applying the same search term as in the keyword-based search, one paper was left from the snowballing results. This paper was added to this study's corpus, resulting in 66 relevant publications identified in the search process.

3.4 Data extraction and mapping

For the 66 papers that resulted from our search procedure, three authors independently extracted the information described in Table 2 in a data extraction sheet. This extraction is inspired by the classification schema for digital twin literature reviews proposed by [19]. If there was more than one entry for one of the extracted parameters (e.g., Dalibor et al. [S5] propose software artifacts that use model interpretation for their configuration and also generate visualization software artifacts using code generation), each entry was stored in a dedicated row for this paper. After this extraction, the resulting data collected by individual authors was consolidated for each paper. This consolidated data sheet is available in our repository. Additionally, we extracted the following metadata for each paper: (1) authors, (2) publication year, (3) publication venue, (4) publisher.

3.4.1 Categorization of sources and targets of automation techniques

As the obtained models and software artifacts obtained as sources and targets of automation techniques contained many unique values, we categorized the extracted entries to allow for more generalized insights. Most categories were derived from the available data, using keywording [53]. We performed the following iterative process to develop this set of categories.

1. First, the main author of this paper clustered all available data for sources and targets of automation techniques and proposed an initial set of categories based on these clusters.
2. This initial categorization was reviewed and refined by all other authors.
3. This refined categorization was applied to the available data by two authors of this paper. Based on this application, the categories were once again refined.

This process resulted in a set of categories described below. For model artifacts that are used as a source of an automation technique, or as transformation target of model-to-model transformations, we created the following categories:

- *Data model:* a model representing the structure of data omitted by a single physical entity or software entity.
- *Architecture model:* a model containing information on the structure and communication between several physical entities or software entities.
- *Ontology model:* a machine-readable version of knowledge. This model usually contains a structural representation and a definition of rules that can be applied to the modeled knowledge.
- *Discrete behavior model:* a model providing information on the discrete behavior of the PT [12].
- *Continuous behavior model:* a model providing information on the continuous behavior of the PT [12].
- *UI model:* a model containing information on the visual appearance of a user interface for a DT or PT.

Table 2 Summary of our data extraction template

RQ	Dimension	Rationale	Possible values
RQ1	RQ1.1: Automation technique	Based on the definition of MDE provided in the introduction, the models used in a paper need to be processed somehow	• model-to-model transformation • code generation • model interpretation
	RQ1.2: Source and target artifacts	Artifacts used by the extracted automation technique. • For model-to-model transformations, the source and target models of the transformation are collected. • For code generation, the source model is collected as source, and the code generation target as target. • For model interpretation, the interpreted model is collected as source, and the interpretation engine as target	Free text description, followed by a bottom-up keywording procedure to come up with a set of terms
	RQ1.4: Research type	To judge the maturity of the MDE application in a paper, we use the following research types proposed by [52]	Solution Proposal [52]: "A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation." Validation Research [52]: "Techniques investigated are novel and have not yet been implemented in practice. Techniques used are, for example, experiments, i.e., work done in the lab." Evaluation Research [52]: "Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation)."
RQ2	RQ2.1: SLCP	The life cycle phase of the underlying system [2] in which the automation technique is applied	• plan • design • implement • operate • maintain
	RQ2.1: TLCP	The twin cycle phase [19] in which the MDE technique is applied	• as-designed: representing the physical counterpart during its design phase • as-manufactured: characterizing the manufacturing process of the physical counterpart • as-operated: representing the usage and operation of a physical counterpart
	RQ2.1: TT	We investigate the TT represented by the DT by reusing the categories proposed by Dalibor et al. [19]	• individual system • system-of-systems • process • product • biological being • other counterpart

Table 2 continued

RQ	Dimension	Rationale	Possible values
RQ3	RQ3.1: domain	The domain which the MDE application is concerned with, according to a statistical classification of economic activities by the European Parliament [26]	• A - Agriculture, Forestry and Fishing • B - Mining and Quarrying • C - Manufacturing • D - Electricity, Gas, Steam and Air Conditioning Supply • E - Water Supply, Sewerage, Waste Management and Remediation Activities • F - Construction • G - Wholesale and Retail Trade; Repair of Motor Vehicles and Motorcycles • H - Transportation and Storage • I - Accommodation and Food Service Activities • J - Information and Communication[1] • K - Financial and Insurance Activities, • L - Real Estate Activities • M - Professional, Scientific and Technical Activities • N - Administrative and Support Service Activities • O - Public Administration and Defense; Compulsory Social Security • P - Education • Q - Human Health and Social Work Activities • R - Arts, Entertainment and Recreation • S - Other Service Activities • T - Activities of Households as Employers; Undifferentiated Goods and Services Producing Activities of Households for Own Use • U - Activities of Extraterritorial Organizations and Bodies
	RQ3.1: Example	Example that is used to demonstrate/instantiate the MDE application	Free text description

[1] We note here that for applications that implemented information systems for other domains, such as manufacturing or construction, we decided to categorize them in the respective domain in which the information system is used (e.g., manufacturing or construction), and not in the information and communication category

We also defined the following categories for software artifacts that are used as generation targets for code generation and interpretation engine for model interpretation automation techniques:

- *Visualization (GUIs):* a software artifact that is used to provide a visual representation of the DT or PT.
- *Simulation:* software that runs or controls a simulation of the PT.
- *Machine learning (ML):* software that uses machine learning models.
- *Control:* software that implements the behavior of the PT to control the different physical devices.
- *Data processing:* software that processes data omitted by the PT.
- *Data storage:* software used to store data, such as databases or data lakes.
- *Other software:* any piece of software that cannot be assigned to any of the above software artifact categories.

3.4.2 Mapping between different data points

RQ2.2, RQ3.2, and RQ3.3 are answered by comparing different extracted data points. In the following, we provide more details on this comparison.

- For RQ2.2, we use the data extracted for RQ2.1 (i.e., TT, SLCP, and TLCP), and compare it to the data extracted for RQ1.1 (i.e., automation techniques) and RQ1.4 (i.e., research type). In addition, we also compare SLCP and TLCP to understand how the life cycle of the DT compares to the life cycle of the twinned system.
- For RQ3.2, we use the data extracted for RQ3.1 (i.e., domains), and compare it to the data extracted for RQ1.1 (i.e., automation techniques) and RQ1.4 (i.e., research type).
- For RQ3.3, we use the data extracted for RQ3.1 (i.e., domains), and compare it to the data extracted for RQ2.1 (i.e., TT, SLCP, and TLCP).

For each comparison, we use the extracted data for one data point, and map it to extracted data for the other data point that we compare to.

4 Results

This section presents our findings on the bibliometric key facts of the included papers, automation techniques used in the context of DTs in these papers, the DT types to which MDE is applied, and the domains where MDE is applied for engineering DTs. We answer the respective research question at the end of each subsection, summarizing our findings.

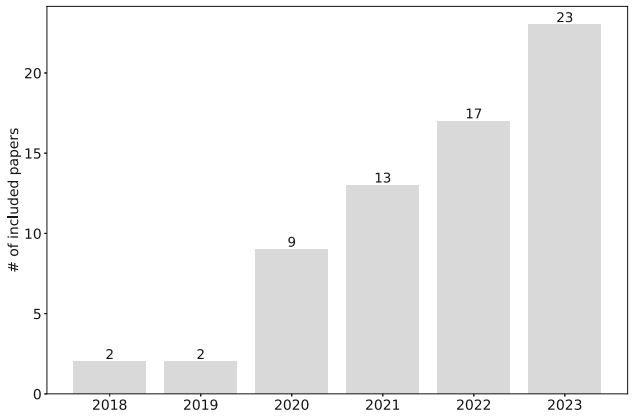


Fig. 4 Overview of publication years of included papers

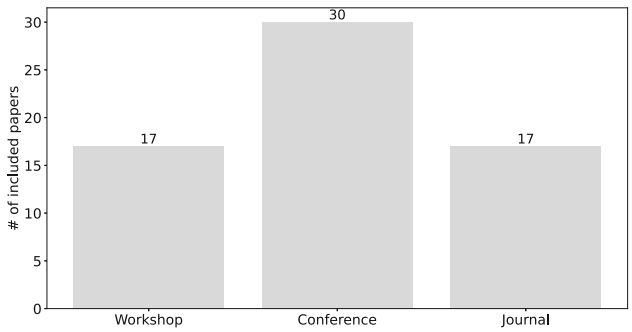


Fig. 5 Overview of venue types of included papers

4.1 RQ0: What are the bibliometric key facts of peer-reviewed literature documenting applications of MDE to DTs?

In the following, we describe our analysis of included papers with respect to their publication year and the type of venue at which they are published.

Main findings of RQ0.1 (In which years are they published?): Fig. 4 gives an overview of the publication years of MDE for DT papers. While in 2018 and 2019, only two papers were published each year, 2020 seems to have been the year when MDE for DTs gained traction in research as the number of published papers rose to 9, with a steady increase of papers in the succeeding years. 2023 shows a peak with a total of 23 publications identified as part of this study. Overall, we can observe that more than one third of the identified papers were published in 2023, and around 60% have been published in 2022 or 2023. This emphasizes that the studied topic seems to be relevant and it is definitely in its growing phase.

Main findings of RQ0.2 (In which types of venues are they published): With respect to the venues at which the included papers are published (Fig. 5), almost half of

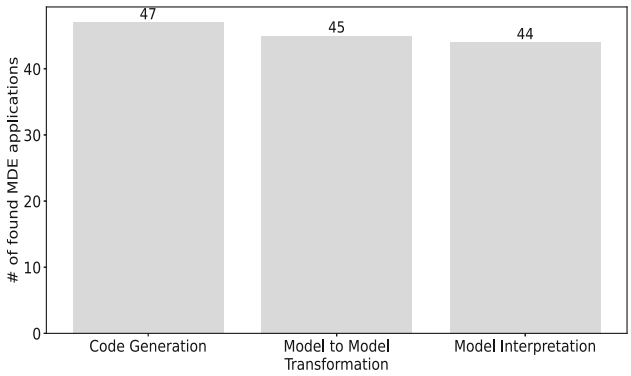


Fig. 6 Number of applications for each automation technique

them (about 48%) can be classified as conferences. The conferences with the highest number of publications are the International Conference on Model Driven Engineering Languages and Systems (MODELS) with 5 included papers and the International Conference on Emerging Technologies and Factory Automation (ETFA) with 3 included papers. There are 17 entries for journals and for workshops. Whereas the journals are published in rather diverse venues, 15 from 17 workshop papers are published in the companion proceedings of the MODELS conference, mostly for the International Workshop on Model-Driven Engineering of Digital Twins (MoDDiT).

4.2 RQ1: How and how often are automation techniques applied to DTs in peer-reviewed literature?

In the set of the 66 considered publications, we identified 136 applications of automation techniques. In the following, we present the results regarding the application of these automation techniques.

Automation techniques: As already mentioned, we have gathered 136 instances of utilizing automation techniques from the set of considered papers. More precisely, there were 23 papers in which we identified the application of 2 automation techniques, 10 publications with 3 automation techniques, and 6 papers with more than 3 automation techniques. In the remaining 27 publications, only one single automation technique was applied. The distribution of the techniques' applications is relatively evenly spread, with each automation technique being applied between 44 and 47 times, as illustrated in Fig. 6.

Main Findings for RQ1.1 (How often are the different automation techniques applied in the context of DTs?): We identified 136 applications automation techniques in 66 papers. Whereas, 27 publications applied only one single automation technique, the remaining papers applied 2 or more techniques within the documented research. The distribution of the application of automation techniques is rather evenly spread between model interpretation, model-to-model transformation, and code generation.

Software artifacts: Software was used in the context of automation techniques either as model interpretation engine or as code generation target. Overall, 91 of the 272 identified artifacts are software artifacts. (The rest is categorized as model artifacts, which is described in the next paragraph.) With 24 occurrences, data processing software is the most commonly used software artifact. It is used in 65% of cases for model interpretation. 19 automation techniques use simulation engines as their target. In around 80% of these cases, these engines use model interpretation. In these cases, the engines are simulation tools, mostly referred to by the authors with generic names such as "DT simulation" [S62] or "simulator" [S21, S26]. Code generation usually generates code to control such simulations [S52, S39, S31]. For the 18 occurrences of visualization software, 78% use code generation, usually to generate some sort of graphical user interface. The only additional category of software artifacts that is used more than 10 times is control software. 70% of the control software artifacts use MDE to interpret a discrete behavior models. In the other 3 occurrences, the control software artifacts are generated based on BPMN [S14], AADL [S31], or MontiThings[S34] models.

Model artifacts: Table 3 details the categorization of model artifacts used as source model for model-to-model transformation or code generation (136 overall), or used as generation target for model-to-model transformation (45 overall) as described in Sect. 3.4.1. This categorization reveals that most of the 136 identified source models are categorized as data models (54 applications, which is 40% of the overall source models) or discrete behavior models (36 applications, which is 26% of overall source models). Data modeling of source models predominantly relies on UML class diagrams (10 occurrences) and SysML BDDs (6 occurrences), whereas discrete behavior models are most often represented by BPMN models (6 times). While discrete behavior models were used only 8 times as transformation target of model-to-model transformation, data models were again used extensively (20 occurrences, which is almost 50% of overall target models). In general, even though these standardized languages such as UML, SysML, or BPMN are the most frequently used languages for source models, there is a rather diverse landscape of used behavior and data modeling languages (56 different languages for source and 25

Table 3 Categories of source and target models of automation techniques found in the literature

Category	Source model	Target model
Architecture model	CPSAML [S23], MontiArc [S19, S4, S35, S20, S59], Domain-Specific Language (DSL) components [S19], Architecture Analysis & Design Language (AADL) [S31], MontiThings [S34], SysML [S6], Architecture Model [S44]	SysML [S23], MontiArc [S35], AADL [S31], MontiThings [S34]
Continuous behavior model	Semantic Reference Model [S21], Unity [S53, S52], Modelica [S16], DTD-WPL [S41], Physics-based model [S40], simulation model [S63]	Simulink [S64], Linear No-Threshold (LNT) Model [S31]
Discrete behavior model	Fault Trees [S51, S15], behavior model [S37, S62], SysML v2 Action Sequences [S27], SDL model [S50], SysML State Machines [S64, S31], Discrete Markov Decision Process [S12], Behavioral AI Model [S12], Petri-net [S18], Event–Condition–Action Model [S19], Case-based Reasoning Model [S19], Process Model [S11], behavioral model of the PT [S54] Business Process Modeling Notation (BPMN) [S14, S60, S33, S49, S6, S40], Causal Directed Acyclic Graphs (DAG) [S58], Activity Diagram [S49], Henshin model [S22], Twin Process Model [S39], IoTBPM [S6], Application-Specific Behavior Model [S59], Protocol State Machine [S28], DTD-WPL [S41], Reasoning Model [S44], Process Model [S44], Finite State Machine [S17], simulation model [S25]	Behavior Trees [S15], DT Modeling Language [S37], Markov Chains [S27], Fault Trees [S27], Discrete Event System Specification [S50], Snap! [S50], Finite State Machine [S1], Activity Diagram [S49], IoTBPM [S6]
Data model	Hybrid Model (Physical, Geometric, Behavior, Rule) [S66], runtime data model [S66], SysML Block Definition Diagram (BDD) [S51, S27, S64, S23, S3, S36], AutomationML [S56, S38, S5], Electrical System Model [S57], DT Operations-dedicated Model (ODM) [S15], Data Structure Model [S37, S11, S65, S41], DT-Metamodel [S38, S2], DataTime DSL [S42], Communication specification [S19], UML Class diagram [S19, S43, S4, S35, S20, S47, S46, S59, S45, S17], Asset Administration Shell (AAS) [S7, S30, S8], M&CML [S1], raw data model [S65], CityGML model [S61], structural model of the PT [S54], Runtime Snapshot modeled as UML Object Diagram [S47], DT Physical Model [S62], Geometric Model [S62], OPCUA Model [S59], DSML for air conditioning [S48], Goal Structure Notation (GSN) Model [S13], Intermediate Representation of Abstract Syntax Tree [S8], Domain Model [S44], OCL Model [S44], JSON [S55], Object Event Table [S17], building model [S24], Speech Interaction Model [S32]	Operations-dedicated model (ODM) [S15], DT Modeling Language [S37], DT-Metamodel [S38], Digital Twin Definition Language (DTDL) [S38, S60], UML Profile [S29, S30], AutomationML model [S5], Unity Model [S3], M&CML Model [S1], SysML BDD [S65], cyber-physical spaces (3D Topology) Model [S61], UML Class Diagram [S49], JSON [S48, S55], ROS Launch Files [S13], configuration model [S41], Abstract Syntax Tree (AST) [S8], medicine dispenser structure instance model [S55], raw data model [S25]
Ontology model	Ontological Modeling Language (OML) [S29, S30], SADL [S1], Web Ontology Language (OWL) [S30, S10], Knowledge Base Model [S26], ADOxx-based DSL for air conditioning facilities[S48], ODP Model [S10], AAS Metamodel in Domain-Specific Language [S9], Device Domain Model [S55], medicine dispenser structure metamodel [S55]	OWL [S30, S10], Knowledge Base Model [S26], Explanation Model [S44], Feature Model in XML [S10], Bayesian network model for risk analysis [S24]
UI model	MontiGem GUI model [S5, S43, S35, S20, S44], GUI Model [S11]	MontiGem GUI model [S43]

Source models are used as transformation source by either model-to-model transformation or code generation, or processed by model interpretation approaches. Target models are transformation targets of model-to-model transformations

different languages for target model artifacts). For architecture modeling, MontiArc is used in 5 of 11 cases as source model and 1 of 4 cases as target model; thus, it is the most often used language for architecture modeling. For UI configuration models, in all cases, dedicated GUI modeling languages were created by the authors. These GUI modeling languages are also mostly used as source models of code generation approaches. For ontologies, OML and OWL are the only two out of the 9 source and 5 target languages that are used more than once as either source or target.

Main Findings of RQ1.2 (Which modeling and software artifacts are used by these automation techniques?): Data processing software is the most common software artifact used by automation techniques (accounts for 26% of all software artifacts), followed by simulation engines, visualization, and control software. Whereas model interpretation is more often used than code generation for simulation engines (80%) and control software (67%), code generation is the preferred automation technique for visualization software (78%) and data processing (67%) software.

With respect to modeling artifacts, data models or discrete behavior models are used in around 80% of identified cases. In total, 47% of modeling artifacts represent structural aspects of the PT. For discrete behavior, data, and ontology models, a large variety of languages is applied, even though standardized languages such as UML, BPMN, and SysML are the most frequently used ones across these categories.

Combinations of inputs and outputs. A detailed mapping of source and target artifacts for each extracted paper is available in Appendix A. In the following, we focus on the combination of artifact categories, based on Fig. 7, to create a more generalized mapping.

Regarding model-to-model transformation (see Fig. 7a), ontologies are mostly transformed into data models [S29, S1, S30, S55]. For instance, OML models are transformed into an UML Profile for annotating AAS models [S30]. In one case, an ontology model is transformed endogenously into an instance model for capturing runtime data [S26]. Concerning discrete behavior models, most often they are transformed again into discrete behavior models for refinement [S15, S27, S50, S49, S6]. For instance, BPMN models are transformed into activity diagrams in a multi-paradigm modeling approach for digital twins [S49]. Second most popular, model-to-model transformations are used to transform discrete behavior models into data models [S15, S37, S60, S49]. For instance, one approach [S60] transforms each BPMN lane representing an IoT device into an interface in the DTDL language. The created DTDL model then can be processed by the Azure Digital Twin Explorer. Another approach transforms discrete behavior models into ontology models for explainability [S44]. In most cases, data models are transformed again into data models. These transformations, for instance, are used for refinement of the input model [S37] or for reusing design time models as input for simulations [S38, S3]. Second most model-to-model transformation applied to data models are transformations into discrete behavior [S27, S1]. For instance, the actions and their dependencies to parts of a SysML BDD of a robotic arm are analyzed and mapped to a fault tree that can be leveraged for risk analysis [S27]. Similar to discrete behavior and data models, architecture models are transformed mostly for the purpose of refinement [S23, S35, S34], and in one case, they are transformed into continuous behavior models [S31].

For code generation, input models are UI configurations, ontology models, discrete or continuous behavior models, data models, or architecture models (see Fig. 7b). From UI configuration models, in all cases, some kind of DT cockpits or visualization dashboards are generated [S5, S43, S35, S11, S20, S44]. Discrete behavior models are utilized to generate control code [S14, S6], simulation software [S39], visualization software [S11], data processing software [S17], or other kind of software for digital twins [S60]. From data models, in most cases, data models, e.g., class diagrams or SysML BDDs, are used to generate data processing software [S5, S4, S20, S59, S8, S17]. In the second and third most cases, visualization software is generated [S23, S43, S35, S11, S44]. Continuous behavior models are always leveraged for generating simulations of the PTs by DTs [S52]. From architecture models, in most cases, some kind of software architecture is generated [S19, S4, S35, S20] that orchestrates components for data processing and controlling the CPS. In two cases, the code for the control software is generated from the architecture model [S31, S34], and another case, the architecture model is used to generate simulation software [S31]. Regarding ontologies, they are used to generate data models that can be instantiated in the form of, e.g., JSON [S9] or APIs that are called for communication between PT and DT [S55]

Concerning model interpretation, ontologies are interpreted either by control software [S26], data processing software [S48], simulation software [S26], software developed by the authors of [S30] to check capabilities of a CPS based on the ontology model. Discrete behavior models are usually interpreted by control software [S51, S19, S59, S44] or simulation software [S18, S58, S22, S62, S28, S41]. However, there are also two single applications where they are interpreted by data processing software [S33] and machine learning software [S12]. Data models are the most frequently used model category for model interpretation. We found 19 applications of model interpretation for data mod-

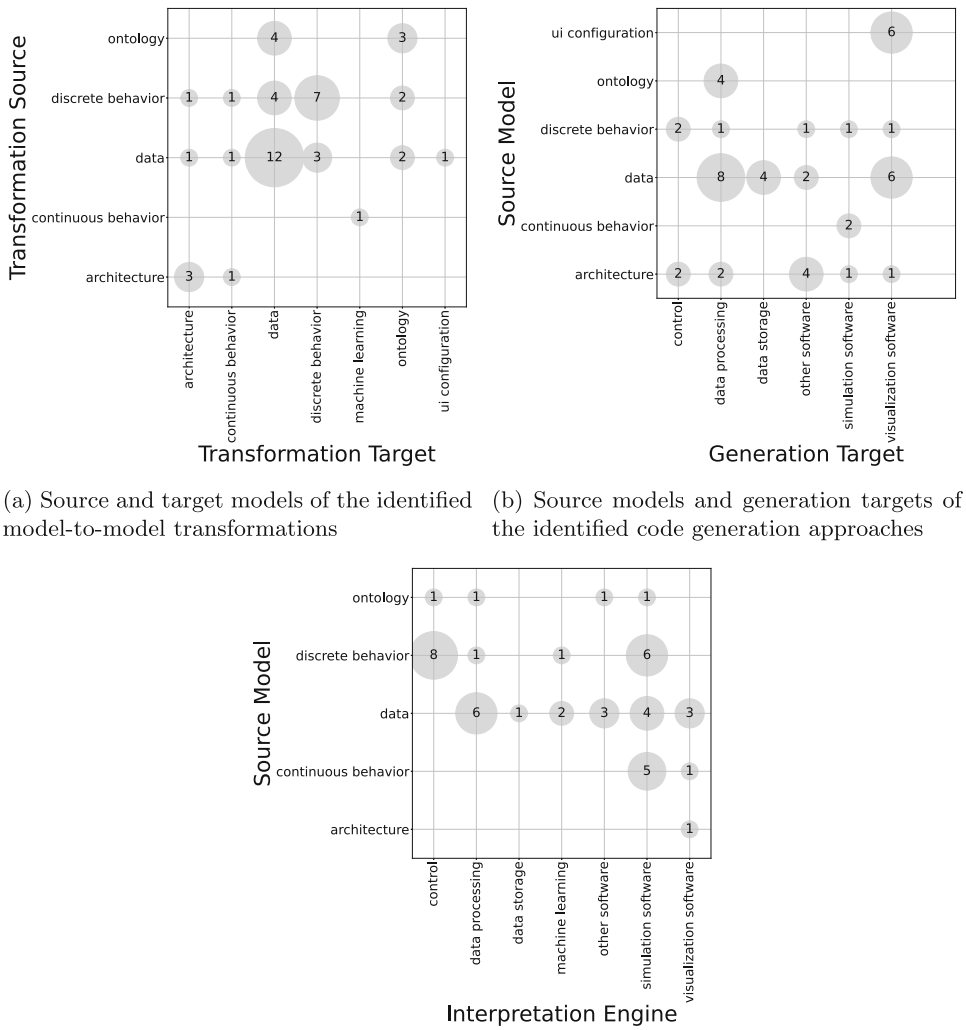


Fig. 7 Artifacts used by different automation techniques as source and target

(a) Source and target models of the identified model-to-model transformations

(b) Source models and generation targets of the identified code generation approaches

(c) Source models and interpretation engines of the identified model interpretation approaches

els. Most data models are interpreted by data processing software, followed by simulation software, and visualization software. For data processing, data models, for instance, are used to establish a typed communication between the DT and the PT [S5]. Regarding visualization, for instance, AAS models are interpreted to visualize data received from the CPS [S30]. Simulations interpret physical and geometrical models to simulate the CPS behavior with high fidelity [S62]. Continuous behavior in the form of, e.g., Modelica models [S16], is always interpreted by simulation software [S21, S53, S16]. Architecture models are only interpreted in one case, where they are utilized for visualization purposes [S5].

Main Findings of RQ1.3 (Which combinations of input and output artifacts are used by these automation techniques?): Model-to-model transformations most often transform data models again into data models, and second most, discrete behavior models again into discrete behavior models. Code generation approaches usually utilize data models to generate data processing software and UI configurations to generate visualizations. Discrete behavior models are synthesized into control software, simulation, and visualization software. With respect to model interpretation, discrete behavior models are frequently interpreted by control or simulation software, and data models are frequently interpreted by data processing or simulation software.

Research Type: In each included paper, the application of one or several automation techniques led to a technological

solution that was applied in the context of DTs. When investigating the type of this presented research (cf. Figure 8a), we can observe that only 20% (13 from 66) investigated papers actually describe evaluation research. When investigating the application of automation techniques within these publications (cf. Figure 8b), the same pattern emerges (26 from 136 applications of automation techniques provide evaluation research). Interestingly, most evaluation research papers (50%) use model interpretation. Model interpretation is in general the automation technique with the highest maturity with respect to the employed research types, as 27 out of the 44 identified techniques are at least in the stage of validation research.Code generation contains the highest number of applications that are still classified as solution proposal.

In most model interpretation approaches that are classified as evaluation research, a simulation software interprets a discrete behavior model [S20, S41], continuous behavior model [S48, S21], or data model [S66], or a data processing software interprets a data model [S66, S19], discrete behavior model [S61], or ontology model [S48]. Interestingly, most model-to-model transformations that are part of evaluation research transform discrete behavior into other discrete behavior models [S50], or data models into other data models [S65, S48, S13, S55]. Most code generation applications used in evaluation research papers generate code for data processing software [S19, S4, S55].

Main Findings of RQ1.4 (What is the research type of the studies that apply these automation techniques?): 40% of the approaches proposed in the surveyed papers are only at the level of solution proposals, whereas only 20% provide evaluation research. Model interpretation is the most used automation technique in the latter approaches. Evaluation research papers usually use model interpretation within simulation software or data processing software, and model-to-model transformation is almost always used in evaluation research to transform one model into another model of the same type (i.e., data models into data models, discrete behavior models into discrete behavior models), similarly to the overall results presented in RQ 1.3.

4.3 RQ2: To which types of DTs are automation techniques applied in peer-reviewed literature?

With respect to the type of DTs, we investigated the target system which the DT represents in the included papers, together with the DT's life cycle phase for which the automation technique is applied. In the following, the results of this investigation are presented.

Twinning Targets: Seven out of the 66 included papers did not include sufficient information to extract the intended TT of their proposed solution, or the solution was described on an abstraction level that did not restrict its application to particular TTs. The remaining 59 publications (Fig. 9) were mostly focused on DTs that represent individual systems (69%), followed by systems of systems and processes. MDE is not yet applied to other TTs from the used categorization (cf. Sect. 3), i.e., DTs that represent products, biological beings, or other counterparts.

With respect to the distribution of automation techniques among TTs (*cf.* Fig. 9b), code generation is the most frequent automation technique for individual systems, whereas model-to-model transformation is most common for systems of systems and processes. In the context of DTs representing processes, model-to-model transformation is usually used to transform between different process representations, e.g., from BPMN to UML activity diagrams [S49]. Whereas for DTs representing individual systems, automation techniques are applied almost twice as often during implementation than during other SLCPs, design is the SLCP most often used for both processes and systems of systems.

SLCP: For each automation technique, we also classified the system life cycle phase (SLCP) of the twinning target in which the technique is applied. While MDE is applied in all SLCPs, most applications occur during implementation, followed by design and operation. Applications in the remaining SLCPs (maintain, plan, test) are negligible ($\leq$ 5 applications of automation techniques per SLCP, *cf.* Fig. 10a).

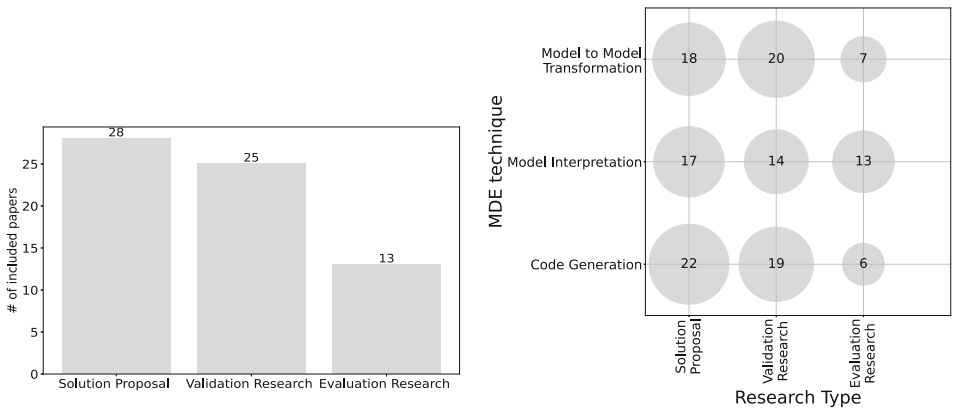
Whereas 63% of applications in the implementation and design phases employ at least validation research, over 50% of automation techniques applied during operation have not yet advanced from the stage of solution proposals (*cf.* Fig. 10b).

The predominant set of MDE automation technique applications consists of code generation during implementation, followed by model interpretation during operation and model-to-model transformation during system design (*cf.* Fig. 10c).

Code generation approaches in the implementation phase usually use dedicated UI models in combination with UML class diagrams to generate visualization software in combination with data processing software [S5, S43, S35, S20], or use architecture models [S5, S4, S35, S20, S31, S34] to generate different kinds of software, such as simulation software [S31] or control software [S31, S34]. During system design, code generation is used to generate data stores [S51], or code templates that can be used later during implementation [S60, S59]. During system operation, code is generated that is directly used by either simulation software [S56, S52, S39] or data processing software [S4].

Model-to-model transformation approaches that are applied during the design phase of the underlying system usually transform discrete behavior models into other discrete behavior models [S50, S49], data models [S37, S60, S49]

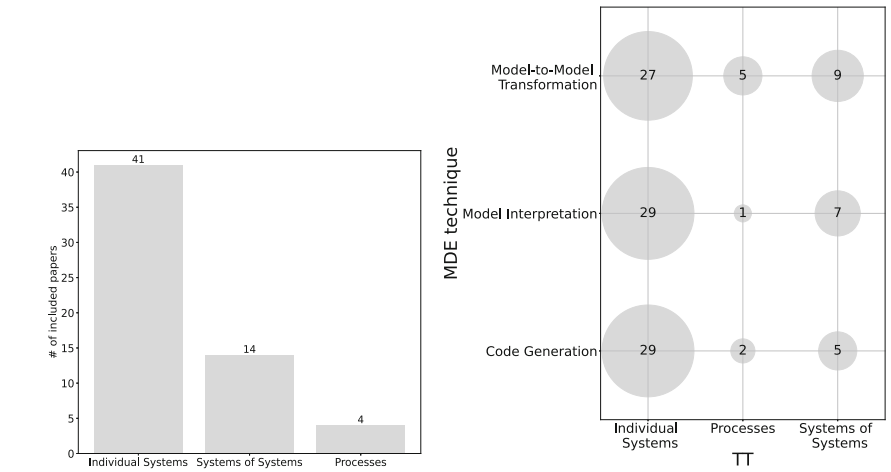
Fig. 8 Overview of research types



(a) Overview of research types of the included papers

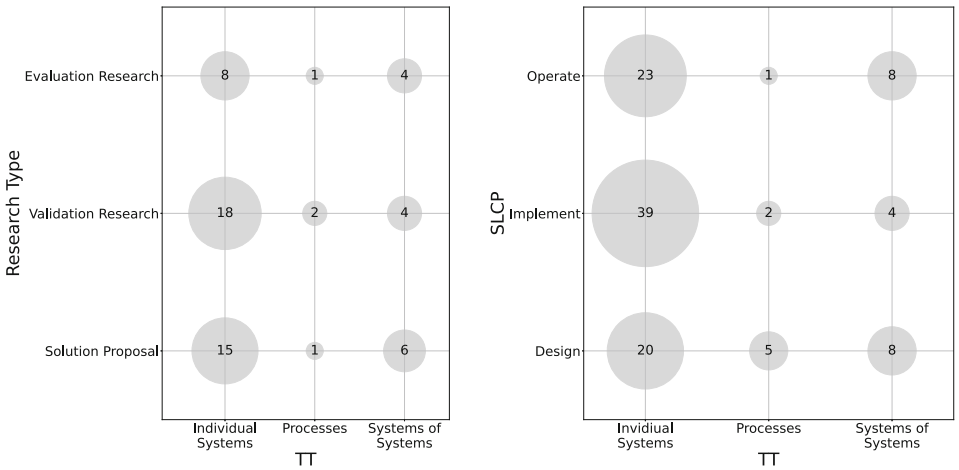
(b) Overview automation techniques for the different research types

Fig. 9 Results for TT



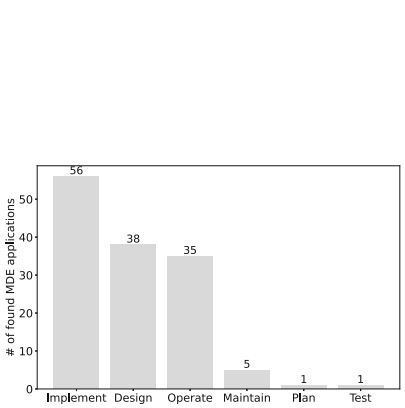
(a) Overview of identified twinning targets

(b) Automation techniques applied for the different twinning targets

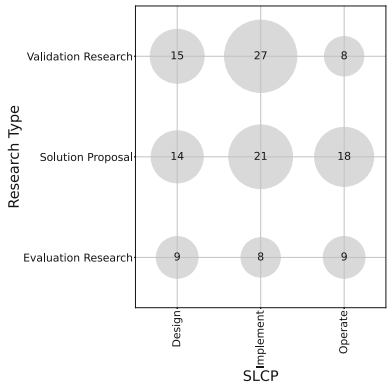


(c) Research types of applied approaches for the different twinning targets

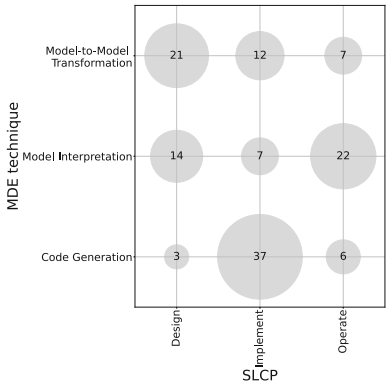
(d) most common SLCPs of the different twinning targets



(a) Overview of identified SLCPs



(b) Distribution of research types of approaches for most common SLCPs



(c) Distribution of automation techniques among most common SLCPs

Fig. 10 Results for system life cycle phases (SLCPs)

or architecture models [S31] into other data models [S37, S3], continuous behavior models [S64], architecture models [S35], or ontology models [S30], or ontologies into other ontology representations [S26] or data models [S29, S30]. Further details on the identified SLCPs can be found in Fig. 10.

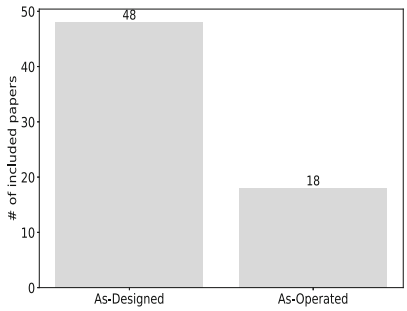
TLCPs: With respect to the life cycle phase of the DT (i.e., [19], as described in Table 2), most of the 66 included publications describe as-designed DTs, followed by as-operated twins, whereas MDE is not yet applied to as-manufactured DTs (*cf.* Fig. 11a).

In comparison with the overall distribution of automation techniques (cf. Figure 6), Fig. 11b shows that model interpretation is less frequently applied to as-designed DTs, but more often to as-operated DTs. These results are similar to the SLCP, where model interpretation is also most often used during operation.

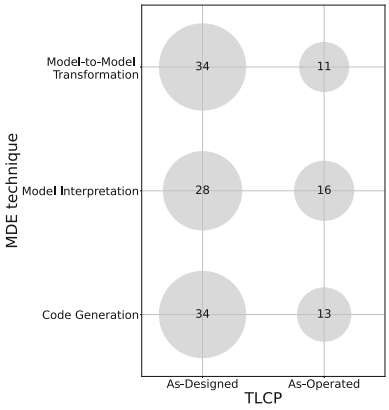
When looking into the distribution of different SLCPs for different TLCPs (cf. Figure 11d), it is observable that as-designed DTs are most often used in the design or implementation phase of the underlying system. However, there are also 16 applications of as-designed DTs to the operations phase, where they usually use as a source model for model interpretation [S57, S2, S53, S46], or a generation target for code generation [S52, S4, S22], even though as-operated twins are more frequently (19 applications) used during system operation.

Main Findings of RQ2.1 (Which TT does the DT represent, in which SLCP of the TT are automation techniques applied, and what is the TLCP of DTs to which automation techniques are applied?): Most applications of automation techniques are performed in the implementation phase of the twinned system, followed by design and operation phases. Most identified papers apply automation techniques to as-designed DTs (86%) representing individual systems (69%). There are less applications to systems of systems and processes as well as as-operated twins.

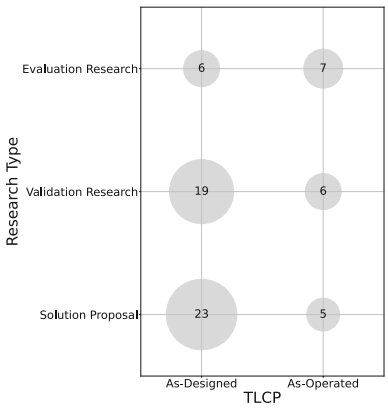
Fig. 11 Results for TLCPs



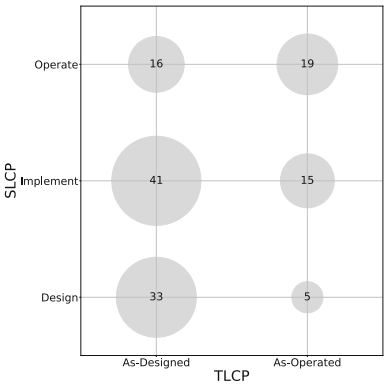
(a) Overview of identified TLCPs



(b) Distribution of automation techniques among TLCPs



(c) Research types of approaches for different TLCPs



(d) Distribution of most common SLCPs for different TLCPs

Main Findings of RQ2.2 (How does the application of automation techniques (identified with RQ1) vary for different DT types?): Similarly to the general picture (cf. RQ1.1), for individual systems, the application of MDE techniques is mostly equally distributed. However, for systems of systems and processes, model-to-model transformation is the most frequently applied automation technique. Similarly to automation techniques applied in the operation phase of the twinned system, model interpretation is also the most frequently applied automation technique to as-operated DTs. Whereas there is no significant difference in the research type of solutions targeting different TTs, MDE applications to as-operated DTs are more mature compared to applications to as-designed DTs. With respect to the TLCP, most applications of automation techniques applied to as-operated twins are performed during the operation phase of the twinned system.

4.4 RQ3: In which domains are automation techniques applied to DTs in peer-reviewed literature?

To investigate the domains in which MDE is used for DTs, we classified papers according to the statistical classification of economic activities [26]. We contrast these extracted domains with the previous surveyed classifications, e.g., employed research type and used automation techniques, to detect differences in the MDE application between application domain. The results of this investigation are presented in the following.

Domains: Overall, only 58 out of the 66 analyzed papers evaluated or instantiated their proposed solution in a particular domain. The 58 extracted data points are spread across 9 out of the 21 domains used for classification (cf. Figure 12). Furthermore, there are only 4 domains that are used in more than 3 papers. Among these 4 domains, manufacturing contains the highest number of papers (21 papers, which is 36% of extracted domains), followed by transport with 12 papers (21%), construction (9 applications), and energy (5 applications). In manufacturing, most use cases are con-

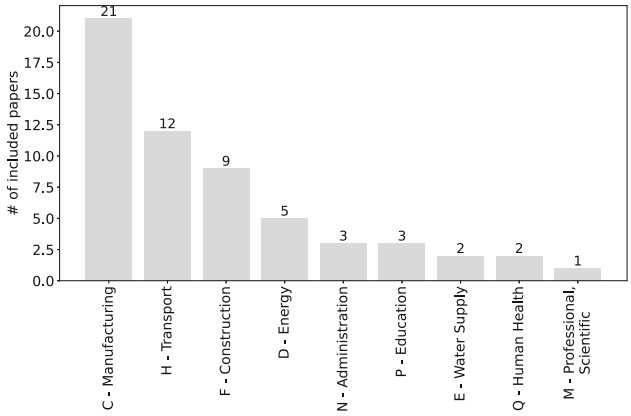


Fig. 12 MDE coverage of application domains

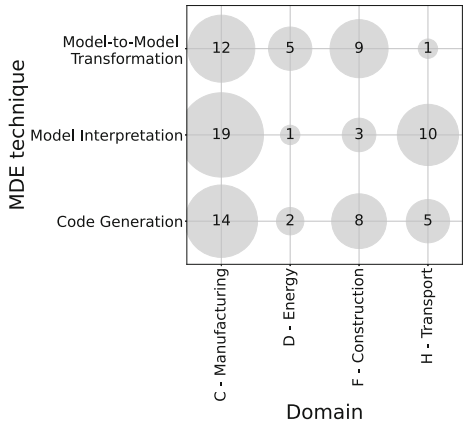


Fig. 13 Distribution of automation techniques among the most targeted domains

cerned with modeling, monitoring and controlling of shop floor equipment (mostly injection molding machines [S5] and robotic arms [S47]). Transport mostly contains use cases that are concerned with some sort of logistics (e.g., monitoring and optimization of the public transport system of the city of Rennes [S42] or Malaga [S45], simulation of autonomous cars [S21, S13], or access control of limited traffic zones [S51]. In construction, use cases are primarily focused around smart homes, from the management of a floor cleaning robot [S23] to CO_2 measurement [S60] and air conditioning automation [S13].

Main Findings of RQ3.1 (For which domains are automation techniques applied to DTs?) The most common domain in which MDE is applied for DTs is manufacturing (21 included papers), followed by transport (12 included papers), construction (9 included papers), and energy (5 included papers). There are 5 more domains with only marginal (<5) applications. For all other domains, no MDE automation technique application for DTs is documented.

Automation techniques and their application per domain Manufacturing contains not only the most MDE applications overall, but also the most applications for each individual automation technique, and the most mature solutions (approaches categorized as evaluation research), as shown in Fig. 14b. Model interpretation is the most common automation technique in manufacturing. Model interpretation is also the automation technique used in the most mature solutions, which might explain the higher maturity of papers in the manufacturing domain with respect to their employed research type.

The distribution of automation techniques among the domains is shown in Fig. 13. Manufacturing, as the domain with most applications, uses all 3 automation techniques. Whereas energy and construction mostly use model-to-model transformation and code generation, the transport domain uses model interpretation as predominant automation technique.

Main Findings of RQ3.2 (How does the application of automation techniques (identified with RQ1) vary for the identified domains?) There is a definite impact of the domain on the application of automation techniques. Whereas in manufacturing, all automation techniques are applied, construction favors code generation and model-to-model transformation, while in transport model interpretation is most often used. In contrast to the overall results (cf. RQ1), there is a clear difference between the number of applications of different automation techniques for different domains. In the most common domain, i.e., manufacturing, model interpretation is the most common automation technique, followed by code generation. In the second most common domain, construction, code generation is by far the most common automation technique. Even though most automation techniques are rather immature in all domains (similarly to the overall results), most mature applications are applied to manufacturing.

DT type per domain: Manufacturing is the domain that contains most MDE applications (8 applications) to as-operated twins, and the domain that contains most applications to DTs that represent either individual systems, systems-of-systems, or processes. Generally, in all domains the DTs represent mostly individual systems, however, in the top 4 domains also DTs of systems-of-systems have been targeted. The energy domain is the only one which only contains MDE applications targeting the as-designed phase. A more detailed overview of how MDE is applied in different domains is given in Fig. 15. Manufacturing shows a balanced distribution of automation technique applications across SLCPS, whereas construction and energy mostly use automation techniques in the implementation phase, and transport in the design and operation phases.

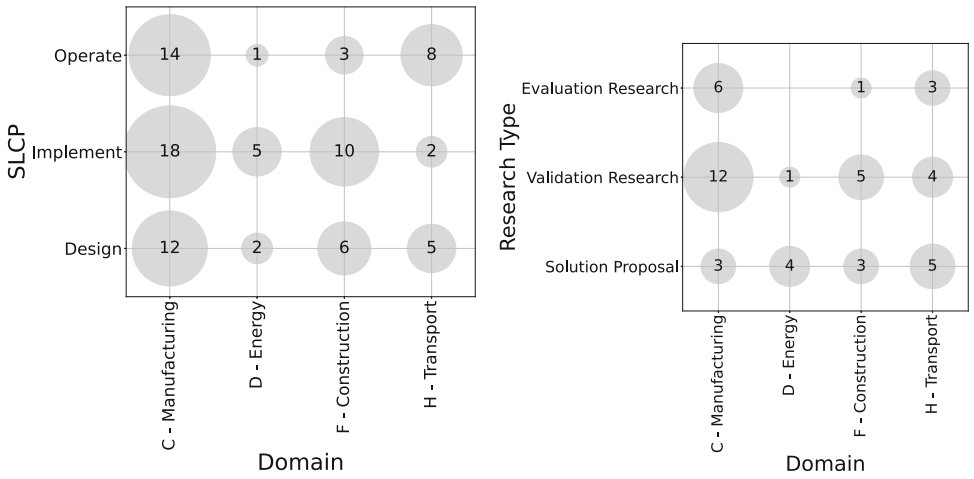


Fig. 14 Results for the most targeted domains

(a) System lifecycle phases (SLCPs)

(b) Research types of applications

Main Findings of RQ3.3 (How does the DT type (identified with RQ2) vary for the identified domains?) There is an observable impact of the domain on the TLCP of DTs to which automation techniques are applied. Whereas manufacturing contains most applications to both as-designed and as-operated twins, other domains hardly contain any applications to DTs in the as-operated life cycle phase. With respect to the twinning target, the predominance of applications to DTs representing individual systems is observable across all investigated domains. With respect to the SLCP, the results in manufacturing are comparable to the overall results (cf. RQ2.1), whereas transport and energy both show a higher use in the implementation phase and transport a less frequent application of automation techniques in the implementation phase.

5 Discussion

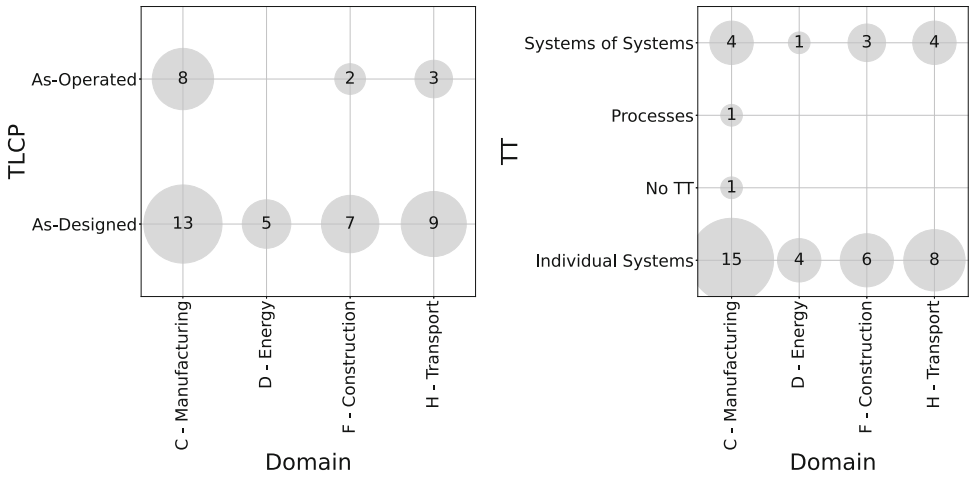
5.1 Discussion of RQ1: How and how often are automation techniques applied to DTs in peer-reviewed literature?

Our results show that even though there is no significant difference in the application frequency, code generation is the most often used automation technique in the context of DTs. It enables the generation of (i) data processing infrastructures, which represent a core functionality for DTs to keep in sync with the state of PTs, (ii) data storage, and (iii) graphical user interfaces to represent data from the PT. As all three categories deal with data, it is not surprising that data model is the most frequently used source model category to generate these software artifacts, followed by dedicated UI models to generate the graphical user interfaces.

Next, model-to-model transformations are the second most frequently employed automation technique. They most often transform models representing either discrete behavior or data into other models of the same category to be reused, e.g., in another tool or in another context. The reason for this could be that tools employed in different SLCPs require different representations of similar models. Model-to-model transformations seem to be a good fit for providing interoperability in such tool chains [43]. For instance, SysML models are transformed into Simulink models [S64] to be used within the Simulink tool, or AutomationML models are transformed into DTDL models to be reused in the Microsoft Azure cloud platform [S38]. Furthermore, source and target models need to describe the modeled subject on a similar abstraction level to be transformed automatically. Otherwise, information will be lost or needs to be added manually in the target model, or provided by the model-to-model transformation somehow.

Even though model interpretation is the least frequently used automation technique, it only shows slightly fewer applications than code generation and model-to-model transformation. The reason for this could be that a simulation that interprets a behavioral model of a DT is often seen as an essential part of a DT [61]. We found that more than 35% of the applications of model interpretation use simulation software to interpret a combination of continuous or discrete behavior models, data models, and ontologies. Besides, control software mostly interprets discrete behavior models. This indicates that interpretation relies mostly on discrete behavior models specifying the behavior of the PT for the purpose of control or simulation. For the latter, however, because there is no live data available of the PT, additionally, data models and models describing the continuous behavior are necessary to accurately simulate the PT.

All in all, we observed that the source models highly depend on the automation technique used and that data mod-



(a) Twin lifecycle phases (TLCPs) (b) Twinning targets (TTs)

Fig. 15 Figures showing DT characterizations for the most targeted domains

els are the most frequently employed models in automation techniques in the context of DTs. Together with the fact that code generation is the most frequently used automation technique to generate data processing and visualization software, this shows that main purposes of DTs, which are supported by MDE techniques, seems to be monitoring, processing, and storing of data from the PT.

Although model interpretation is the automation technique with the least applications within the surveyed papers, solutions that use model interpretation show the highest maturity of the employed research type, having the most occurrences of evaluation research and the lowest number of solution proposals. This might imply that there is already software available that uses model interpretation (e.g., simulation engines that interpret simulation models, control software that interprets behavior models, or data processing software that interprets data models) that drive the maturity of research using this automation technique, because researchers can build upon these solutions in their work (thus fewer solution proposals) to perform more sophisticated evaluations (thus more evaluation research papers). On the other side, code generation might still not reach these technological maturity, as it still shows a high number of solution proposals, and a rather low number of evaluation research.

Interestingly, MDE applications during operation show a high number of solution proposals (>50% of identified automation techniques), even though this phase contains most model interpretation applications, and model interpretation is generally used by a high number of validation and evaluation research work. On the other hand, more than 60% of solutions applied in the design and implementation phases are in the stage of validation or evaluation research, even though these phases contain mostly model-to-model transformation and code generation applications. This implies that in the operation phase, rather solution proposals using model interpretation are being published, whereas the validation and evaluation research is rather performed during the design and implementation phases. As described above, this higher maturity level with respect to the research type is often connected to existing tooling, such as sophisticated simulation engines, being available for reuse. It seems that such tools are not yet sufficiently available for applications during system operation.

5.2 Discussion of RQ2: To which types of DTs are automation techniques applied in peer-reviewed literature?

Automation techniques are mostly used for DTs that twin an individual system. The low number of applications to systems of systems is most likely explained because composition of DTs is rather unexplored today [45]. The fact that DTs are usually referred to as *virtual representations of physical systems* [38] in the literature might also be the reason for the very low number of occurrences of DTs representing processes, and the lack of applications of automation techniques to DTs representing biological beings, even though there are already discussions, outlined challenges, and experiences reported for such DTs [6, 13, 20, 50].

With respect to SLCPs, MDE is mostly applied during the design, implementation, and operation phases of the twinned system. Unsurprisingly, model-to-model transformation is mostly applied during the design, code generation during implementation, and model interpretation during operation. We expect that this is the case because (i) with code generation, code that is used to implement the twinned system is generated, (ii) with model-to-model transformation, models are refined and ported between different tools during system design, and (iii) for model interpretation, models are interpreted by parts of the running system (e.g., a simulation

engine). However, there are also applications of automation techniques that diverge from this pattern. For example, some code generation applications produce code during the operation phase of the underlying system, e.g., to be used by a simulation or data processing software, or code templates are generated during system design. There are even some model-to-model transformations execute during operation and model interpretation is also performed during system design. Some of the publications using these automation techniques are already in the state of evaluation research.

Automation techniques are usually applied to DTs in the as-designed phase, but we did not find any paper for the as-manufactured phase. This indicates that the majority of MDE applications for DTs is concerned with creating DTs in a model-driven fashion, e.g., to generate a simulation of the system from models or to generate the DT software architecture based on architecture models, in contrast to using existing DT models describing the system operation within the MDE application. As-manufactured DTs, which represent the twinned system during its manufacturing process, are currently neglected by automation techniques, because (i) this TLCP is very specific to the manufacturing domain, (ii) even though this domain contains most included papers, the automation techniques applied in these papers focus on the design, implementation and operation of production equipment, or describing the design of the product being manufactured, rather than focusing on the manufacturing process itself.

Whereas there is no trend in the application of automation techniques to as-designed DTs observable, model interpretation is the predominant automation technique applied to as-operated DTs. Since models at runtime of the system are updated frequently, we assume that interpreting these models might promise a higher efficiency. There is also a clear impact of the TLCP on the research type of the created solution (applications to as-operated DTs show higher numbers of validation and evaluation research), and an impact of both TLCP and TT on the SLCP in which automation techniques are applied. This implies that certain application contexts of automation techniques seem to be preferable, depending on the DT type to which these techniques are applied. For the automation techniques leveraged by DTs describing a system as-operated, usually data or behavior models of the system during the operation phase of the underlying system are used.

5.3 Discussion of RQ3: In which domains are automation techniques applied to DTs in peer-reviewed literature?

There is a clear dominance of the manufacturing and transport domains when it comes to MDE applications to DTs. More than 50% of included papers fall into one of these two domains. Most applications are performed in manufacturing. This is also in line with other studies that investigated DT application domains [19, 62, 66], even though in our survey, manufacturing is not as dominant as, e.g., in the study conducted by Dalibor et al. [19], where 72% of the papers are classified in this domain. Even though other studies [19, 62] also classify construction and energy as domains with a high number of DTs, in our study, transport is the domain with the second highest number of automation techniques for DTs. This means that whereas construction and energy show similar numbers of DT and MDE occurrences, there is a lower number of MDE applications than available DTs in manufacturing, but a higher number of MDE application in relation to the overall available DTs in transport. Still, manufacturing shows significantly a higher number of publications with a high maturity with respect to the employed research type. There are also dedicated surveys available that investigate explicitly the application of DTs in manufacturing [17], energy [28] and construction [35].

The high number of MDE applications, and particularly the high maturity of the research type of the respective papers, are probably due to the fact that DTs are becoming a core concept of digital transformation efforts in manufacturing, e.g., see Industry 4.0/5.0 [25]. This trend is also evident from an increasing amount of industry consortia being formed to push the adoption of DTs in manufacturing, such as the Industrial Digital Twin Association,[2] the Open Industry 4.0 Association,[3] or the Association Industry 4.0,[4] just to name a few. Model interpretation is also by far the most commonly applied automation technique in manufacturing, which is probably because simulation is often used in manufacturing for virtual commissioning and simulation of individual machines before they are assembled in a real-world factory. This also explains why automation techniques in manufacturing usually represent individual systems (i.e., the machine) and are in the as-designed TLCP (i.e., based on information that is available before the machine is assembled).

When looking into common use cases for the transportation and construction domain, they concern highly digitalized areas, such as smart homes, autonomously driving cars, or the digitalization of transportation systems. On the other hand, many domains with a lack of MDE applications, such as social work or real estate, are usually associated with high manual labor effort and a lower level of digitalization. Thus, this level of digitalization might be an indicator of a good fit of a domain for the use of DTs, and the application of MDE to these DTs.

[2] https://industrialdigitaltwin.org

[3] https://openindustry4.com

[4] https://plattformindustrie40.at/?lang=en

6 Threats to validity

For identifying the threats to the validity of our study, we follow the four basic types of validity threats according to Wohlin et al. [72]. In this respect, our study is subject to threats to construct validity, internal validity, and conclusion validity. With respect to the threats to external validity (generalizability), it is important to note that the results of this study can neither be generalized to problem domains other than DTs nor to solution domains other than MDE. As we only considered publications from peer-reviewed venues in our study, the results cannot be generalized to applications of MDE or applications to DTs outside academia (i.e., in an industrial context).

Concerning construct validity, our presented findings are only valid for our sample of papers. Thus, we tried to include as many papers as possible from ACM, IEEE, Scopus, and Web of Science, and only excluded papers according to our exclusion criteria. Additionally, we intentionally designed our search query to be relatively broad and inclusive. This approach allowed us to identify papers related to the concept of model-driven approaches without imposing strict limitations, such as exclusively focusing on software engineering or specific MDE techniques. However, it is worth noting potential limitations in our search strategy. Our corpus contains only papers published from 2018 and following. This may be attributed to our search query that included the term "digital twin," i.e., although researchers had explored analogous concepts before 2018 they did not explicitly label them as digital twins. Furthermore, our search query did not encompass the term "model-based", and we conducted searches within title, abstract, keywords, and venue title fields, without conducting full-text searches. We also found that the results of the Web of Science database seem to depend on the subscription of a research institution to the Web of Science core database, as also described by other researchers [31, 56]. To be as inclusive as possible, we used all search results obtained from the subscriptions of two different universities, namely from the Johannes Kepler University Linz and the University of Stuttgart. To enhance the comprehensiveness of our paper selection, we conducted forward and backward snowballing, aiming to identify papers that might not have been captured by our initial search query. Another potential threat to the construct validity of our study pertains to our chosen inclusion and exclusion criteria, which were applied solely to the title, abstract, and keywords during the screening phase. To minimize the risk of inadvertently excluding relevant publications, we included papers where uncertainty existed among one or more authors, subjecting them to further investigation during the detailed review process. Throughout both the screening and detailed review phases, a collaborative effort involving multiple authors was employed to read and discuss the papers, ensuring a rigorous evaluation process. Excluding philosophical, opinion, and experience papers in this process has resulted in 10 exclusions during the screening and 20 exclusions during the detailed reviewing phase. Even though these papers could have included more diverse perspectives in our results, we decided to only include publications that provide a significant level of technical contribution and description which can be analyzed to extract the necessary information for this study.

Our mapping study is also subject to the so-called publication bias [29], i.e., our study can only report on published results. This limitation implies that our study may not encompass all relevant research, as unpublished or non-peer-reviewed work is not included. For example, since our focus was on peer-reviewed literature, we did not find industrial solutions, such as the Azure Digital Twins service,[5] or the Ansys Twin Builder,[6] which might have been accessible via blog posts or preprint articles. Including such publications in a multi-vocal literature review [27] might be one line of future work, as this gray literature could include solutions with higher technological readiness levels, such as the examples given above. Additionally, we included publications that claim to utilize MDE without providing extensive examples. In such cases, we relied on the authors' integrity and adherence to scientific principles, as we must trust the accuracy of their claims. Another aspect of construct validity is the so-called mono-method bias, referring to the use of a single metric to analyze the selected papers. We counteracted this bias by providing several metrics to answer each of our research questions. To mitigate the issue that one paper might apply several aspects that are relevant for our study, i.e., several automation techniques, we extracted all automation techniques for each paper as a dedicated entry. To cope with the option that an extracted automation technique regarded aspects of several categories for a particular metric, we allowed the assignment of automation techniques to multiple categories for each metric.

A threat to internal validity is that our study relies on the quality of papers that were included. Additionally, our categorization may not comprehensively encompass all facets of the described data sets. This limitation stems from the fact that certain papers offer only high-level overviews, while others provide extensive details.

Threats to conclusion validity arise drawing wrong conclusions and from the study's replicability. Regarding the former, we have discussed issues that could lead to wrong conclusions in the context of threats to internal validity. For study replicability, we describe the method we used to derive the described results in Sect. 3 and provide a replication pack-

[5] https://azure.microsoft.com/products/digital-twins/

[6] https://www.ansys.com/products/digital-twin/ansys-twin-builder

age containing all primary and secondary data, as well as the automation scripts we used to process this data, online.[7]

7 Challenges and action points

In the following, we present a list of challenges which have to be tackled in order to further advance the application of MDE to DTs based on the results of our survey. In addition, we propose several action points to tackle these challenges in future activities.

Challenge: Increase the maturity of available research to make it industry-ready. In our survey, we found already 136 applications of automation techniques in the 66 surveyed articles (cf. RQ1.1). However, in RQ1.4, we also found that the approaches using these automation techniques are still to be evaluated in an industrial context. A considerably high number is classified as solution proposals, whereas only a rather low number classifies as evaluation research. Thus, one may conclude that providing evaluation research for solutions using MDE applications in the context of DTs is challenging or considered as subject for future studies. However, we also found clusters of MDE applications with a particularly high number of evaluation research in RQ1.4, RQ2.2, and RQ3.2. We assume that this is due to the availability of existing software that can be easily reused by researchers in these areas, which allows them to focus on evaluation research about using these existing technologies.

Action Points. Validating existing solution proposals, or extending the evaluation of validation research work to practical contexts appears to be an efficient way of advancing the maturity of existing MDE applications for DTs. Existing evaluations can be extended by applying existing solutions to new areas (e.g., new DT kinds or new domains), or applying new methods (e.g., through applications in industrial settings through field studies or technical action research). As there are 26 solution proposal papers and 25 validation research papers available, this research direction provides high potential for future work in this field.

In areas that already contain a high number of evaluation research (e.g., applications of model interpretation (RQ1.4), or applications to DTs representing individual systems (RQ2.2)), it seems promising to build upon this existing work by replicating evaluations for different application contexts, DT kinds, and domains, or reuse the tools described in these papers for future work. For the manufacturing and transport domain, there are also many real-life use cases available that researchers can build upon, as described in detail in RQ3.1. However, even though data from these use cases are available, applying automation techniques to the physical systems used in these domains (e.g., injection molding machines, transportation systems) might be rather cost-intensive and require field access by the researchers. Thus, the construction domain also seems to be a good target for reusing existing solutions, as most presented use cases use low-cost equipment (floor cleaning robots or air quality sensors) that can easily be rebuilt in lab environments. To share access to high-cost physical systems, we need cross-disciplinary research between communities that already perform research in domains such as manufacturing or transport and communities that can build DTs and automation techniques for available systems.

Challenge: Master the variety of artifacts used by automation techniques In RQ1.2, we found that to represent DTs, researchers are already reusing existing general-purpose modeling languages such as UML, SysML, and BPMN for their application to DTs. Nevertheless, many researchers also develop their own languages to represent DTs. As a result, have a plethora of different but similar languages being used for the different aspects of DTs. Of course, this might hinder the improvement of the maturity of this research and its industrial application. We also see that different categories of languages are used for different purposes by different automation techniques (cf. RQ1.3), which further increases the variety of used languages for MDE applications to DTs.

Action Points. The adoption of existing research highly depends on the usage of common languages. As DTs combine different models from different categories, such as behavioral models used by simulations, data models used by data processing or storage solutions, or UI models used by graphical user interfaces, we need a family of DT languages covering these different concerns, that can also be flexibly adapted based on the DT kind (cf. RQ 2) [55]. Common languages for these different concerns are currently being defined by consortia built from both industrial and academic members, such as the Industrial Digital Twin Association[8] or the Digital Twin Consortium[9] by (i) extending existing general-purpose modeling languages such as UML, SysML, and BPMN, (ii) adapting existing domain-specific languages for DTs, such as the Asset Administration Shell, to the specific requirements of MDE for DTs research, or (iii) building new languages from scratch. An investigation of existing languages of DT platforms showed that extending languages that are already used to engineer twinned systems, such as UML class diagrams, provides a solid basis for such common languages [54]. One advantage of reusing such languages is that we can also reuse existing models defined in these languages that already contain information on the twinned system for generating the DT of this system [S38], or reuse existing MDE tooling that is built based on these languages.

[7] https://github.com/cdl-mint/mde4dts

[8] https://industrialdigitaltwin.org/

[9] https://www.digitaltwinconsortium.org

Challenge: Strengthen the connection between academic and industrial DT applications. We also note that our study explicitly focused on peer-reviewed publications. Some of the identified clusters of MDE applications might be different in a purely industrial context or applications that are not published in peer-reviewed venues. For example, we observe in our research projects with industrial partners that several industrial DT applications currently make use of languages such as the Asset Administration Shell or Digital Twin Definition Language, and there are also emerging interpretation engines available for these languages, such as the Azure digital twins explorer[10] or Eclipse BaSyx project.[11] We also observe a drift between the above-mentioned technology, and the tools and languages used in academia.

Action Points. Investigating applications of MDE in non-peer-reviewed literature can be used to confirm, contradict, or even extend the results presented in this paper. Comparing industrial DT practices to the scientific results presented in this study can show interesting gaps between research and practice. To survey non-peer-reviewed literature, a follow-up study can, e.g., make use of the guidelines proposed by Garousi et al. [27]. Additionally, researchers can perform interview studies with industry and domain experts to understand what these stakeholders need, and what is currently missing in the state of the art to fulfill those needs. To better understand industrial requirements, we can also investigate further features and languages used by existing DT technologies by extending the work by Qi et al [59] and Lehner et al. [40]. However, as these technologies are constantly evolving, it also makes sense to provide dedicated online repositories or websites to monitor these platforms.[12] Besides industrial applications, we can also explore DT-specific standards, such as the Asset Administration Shell or the ISO 23247 Digital Twin Framework for Manufacturing [32], and compare them with existing standards, such as UML, SysML, and AutomationML. On a technological side, we can provide bridges between these different modeling languages proposed by these standards to allow transitioning between them.

Challenge: Expand the applications of automation techniques. In RQ1, we found that there are still open spots with respect to modeling languages used by automation techniques. Shedding light on these spots might reveal new potentials and insights of MDE applications in the context of DTs. In the following, we discuss particular action points that can be taken in this regard.

Action Points. We uncovered that a considerable amount of automation techniques is transforming data or discrete behavior models into a different representation of the same category, e.g., for refinement or interpretation of behavior models using simulators for experimental investigations. Using different combinations of source and target artifacts, or broadening the applications of certain model categories, might lead to interesting new insights. For example, enriching DTs with semantic information seems promising [9, 76], but MDE is currently applied to ontologies for rather limited purposes, i.e., mostly to derive data models or configure control or simulation software. For model interpretation and code generation in particular, using different model types increases the information that is available to the software used as generation target or interpretation engine, and thus extends the possible range of applications. For example, behavioral models are currently rarely used for code generation, in addition to generating simulation code, e.g., to automatically generate integration code that connects different components of DT architectures, or to allow interaction with the dynamic behavior of the PT using visualization or control software.

Challenge: Adopt MDE for new DT kinds and their variations. In RQ2, we found that most automation techniques are currently applied to as-designed DTs representing individual systems in their design, implementation, or operation phase. Even though some DTs are also in the as-operated TLCP, representing systems-of-systems or processes, other DT kinds, e.g., in the as-manufactured TLCP, or representing biological beings, are still neglected. Experimenting with the application of MDE to these DT kinds might lead to new insights, or the generalization of existing knowledge.

Action Points. To tackle this challenge, we can make use of existing proposals, such as applying DTs to processes [13] to provide solutions proposals that can then be further validated and evaluated. We can also reuse existing DTs, and start applying automation techniques to them. For example, DTs have already been applied to biological beings, such as humans [6] or cows [50], or overlapping categories such as socio-technical DTs [5] and cyber-biophysical systems [20]. With respect to the SLCP of the twinned system, the research community has already gathered a considerable amount of knowledge on how automation techniques are applied in the testing [69] or maintenance [70] phases of a system. Another aspect which seems not to be investigated until now is the application of MDE for the requirements phases of DTs.

Challenge: Adopt MDE for DTs in new domains benefiting from digitalization. The results of RQ3 show a large imbalance with respect to the domains to which MDE is currently applied. More than half of the papers focus on two domains only. Thus, the generalization of MDE approaches applied in these domains seems to be a critical research direction for the MDE community as modeling is considered as a universal discipline [15]. In manufacturing, modeling languages are already widely used [73], and Industry 4.0 (having the RAMI reference framework) and Industry 5.0 (mentioning DTs as one base technology) seem to be additional

[10] https://github.com/Azure-Samples/digital-twins-explorer

[11] https://eclipse.dev/basyx

[12] https://github.com/cdl-mint/DT_Platform_Comparison

initiatives that drive the adoption of DTs and applications of automation techniques to DTs. In other domains, the level of digitalization to realize DT applications might still be lower. In general, when adopting MDE in new domains, we might be faced with technological boundaries, i.e., connecting existing MDE tools to domain-specific standards and modeling languages, as these domains often still lack basic technologies, such as widely adopted middleware and modeling languages, which are, e.g., available in manufacturing with OPCUA and AutomationML.

Action Points. For many domains (that is, 10 out of the 21 investigated domains), the application of MDE has not yet been explored in peer-reviewed publications. For these domains, generalizing existing approaches from more prominent domains seems promising. For domains that already contain a certain amount of MDE applications, such as construction, there are still some unexplored areas with respect to the application context of automation techniques. For example, there is only one application of model-to-model transformation to the transport domain. Automation techniques in the transport domain also currently focus on the operation phase, and applications in the construction domain focus on implementation. Transferring applications in these life cycle phases to other domains (e.g., implementation automation techniques from construction to transport) can help the research community efficiently close existing gaps. For performing this knowledge transfer, we can collaborate with domain experts, and build bridges between domain-specific tools and existing modeling languages and software artifacts used for applying automation techniques to DTs. Interdisciplinary research communities, such as the *Engineering Digital Twins* community,[13] can be a good starting point for such initiatives. However, we can also leverage existing surveys [28, 35] or vision papers [63, 64] on potentials for DTs in certain domains as a starting point to apply automation techniques to these DTs.

Challenge: Prevent reinventing the wheel. Modeling languages and the application of automation techniques based on models defined in these languages have been well researched in more than two decades. Even though our survey shows that researchers started implementing automation techniques for DTs in 2018, there is a profound source of knowledge from the MDE community [14], investigating (i) concepts that can be reused for DTs, such as runtime models or (ii) applications of MDE techniques to particular domains in which DTs are also applied. If we disregard this existing knowledge, we run into the risk of reinventing the wheel.

Action Points. Researchers can leverage existing knowledge by investigating existing surveys that describe the application of MDE in related fields, such as Industry 4.0 [73], robotics [1, 16], cyber-physical systems [49], and runtime models [23, 65] as a starting point. Particular action points that researchers should focus on are, e.g., reusing languages and techniques from the runtime models community to perform self-adaptation and connect automation techniques with AI-based methods. From the cyber-physical systems domain, connection of discrete and continuous simulations should be more leveraged for DTs. Researchers may also experiment with applying automation techniques to new sorts of languages, that, e.g., provide 3D modeling capabilities for geometry or kinematics, similar to the COLLADA language which is included in the AutomationML language family.

8 Conclusion

The systematic mapping study performed in this paper investigated where and how MDE automation techniques have been already applied for DTs. The presented maps illustrate already available clusters of MDE approaches for DTs, but also point out some spots which are interesting areas to work on by researchers from the MDE community, potentially in interdisciplinary settings to target new application domains.

Future research endeavors may focus on building upon the existing solutions that were identified in our study, increase their maturity, and explore their application to other domains, DT kinds, and application contexts. Furthermore, since we identified a variety of employed modeling languages, another research direction is to work toward a unified modeling language or family of languages for DTs. The corpus of techniques and artifacts collected in this study should be a good starting point for this direction. Finally, our study may be extended by including gray literature and industrial applications as well as DT platforms to derive further strategies for enhancing the adoption of MDE for DTs in industry.

Appendix A Mapping of source and target artifacts for different automation techniques

[13] https://edt.community/

Table 4 Paper processing techniques overview

Paper	Model processing technique	Source	Target
[S66]	Model interpretation	Hybrid model (physical, geometric, behavior, rule)	Performance degradation measurement algorithm
	Model interpretation	Runtime data model	Simulation engine
[S51]	Code generation	SysML BDD	Knowledge Base of software digital twins
	Model interpretation	Fault Trees	FaultTreeAnalyzer
[S56]	Code generation	AutomationML	DT runtime memory
	Model interpretation	AutomationML	Restful API, HTML pages
[S57]	Model interpretation	Electrical system model	
[S15]	Model interpretation	ODM model	UI-based monitoring tools
	Model-to-model transformation	Fault Trees	Behavior Trees (BTs)
	Model-to-model transformation	Fault Trees	Operations-Dedicated Models (ODMs)
[S37]	Model-to-model transformation	Behavior model	Structure + behavior model (DTML)
	Model-to-model transformation	Structure model	Structure + behavior model (DTML)
[S38]	Model-to-model transformation	AutomationML	DT-MM
	Model-to-model transformation	DT-MM (Ecore model)	DT-MM (JSON representation)
[S27]	Model-to-model transformation	SysML v2 action sequences	Markov chains
	Model-to-model transformation	SysML v2 structure models	Fault Trees
[S21]	Model interpretation	Semantic Reference Model of physical product parts	Simulator
[S50]	Model-to-model transformation	SDL model	Discrete Event System Specification (formal model for simulation engines)
	Model-to-model transformation	SDL model	Snap!
[S64]	Model-to-model transformation	SysML models (BDD, Sequence Diagram, Parametric Diagram, State Machines)	Simulink
[S23]	Code generation	SysML	CPS software code (ROS Packages, DT, Monitoring Cockpit). Languages: Java, Typescript, Python, Docker
	Model-to-model transformation	CPSAML	SysML
[S42]	Model interpretation	DataTime DSL	DataTime Framework
[S12]	Model interpretation	Discrete Markov decision process	Risk Analyzer; linear ML methods
[S2]	Model interpretation	System DT (Metamodel proposed by authors in Figure 5)	Learning Agent (Reinforcement Learning Algorithm)
[S53]	Model interpretation	Unity	ROS Bridge
	Model interpretation	Unity	Simulation Controller
[S18]	Model interpretation	Petri-net	Real-time simulation of state
[S29]	Model-to-model transformation	OML Vocabulary	UML Profile
[S16]	Model interpretation	Modelica	Modelica Simulator
[S19]	Code Generation	MontiArc model	Digital Twin Implementation: Java code, Software Architecture Components + Glue Code

Table 4 continued

Paper	Model processing technique	Source	Target
	Code Generation	ECA	Evaluator Component of DT Architecture
	Code Generation	Class diagram	Digital Twin Implementation: Java code (Backend)
	Code Generation	CBR	Reasoner Component of DT Architecture
	Code Generation	Communication specification	Java Code for communication between CPS and DT
	Code Generation	DSL component + GUI models for editor and viewer + DT components	Digital Twin Implementation: Low-code editor for DT configuration
	Code Generation	GUI Model	Digital Twin Implementation: DT Cockpit
[S5]	Code generation	MontiArc	Digital Twin Implementation: Java code, Software Architecture Components + Glue Code
	Code generation	UML CD	Digital Twin Implementation: Java code (Backend)
	Code generation	MontiGem GUI model	Digital Twin Implementation: DT Cockpit
	Model interpretation	Event-Condition-Action (ECA) model	Evaluator Component of DT Architecture
	Model interpretation	Case-based reasoning (CBR) model	Reasoner Component of DT Architecture
	Model interpretation	Communication specification	Java Code for communication between CPS and DT
	Model interpretation	Domain-Specific Language (DSL) components + MontiGem GUI models for editor and viewer + DT components	Digital Twin Implementation: Low-code editor for DT configuration
[S43]	Code generation	MontiGem GUI model	Part of DT UI Cockpit (Typescript)
	Code generation	UML CD	Part of DT UI Cockpit (Python + Typescript)
	Model-to-model transformation	UML CD	MontiGem UI model
[S7]	Code generation	AAS	GPL (shown for C#/Java) Stubs + Contracts
[S3]	Model-to-model transformation	SysML	Unity Model
[S52]	Code generation	Unity	Simulation Controller
	Code generation	Unity	ROS Bridge
[S1]	Model-to-model transformation	M&CML control systems model	Finite state machine
	Model-to-model transformation	SADL ontology	M&CML control systems model
[S65]	Model-to-model transformation	Raw data model	SysML BDD
[S4]	Code generation	MontiArc	Java
	Code generation	UML CD	Java
[S35]	Code generation	MontiArc	GPL code for CPS
	Code generation	UML CD	Digital twin information system
	Code generation	MontiGem GUI model	Web pages
	Model-to-model transformation	MontiArc	New MontiArc ADL with components receiving tagged ports
	Model-to-model transformation	MontiArc, UML CD	Extended MontiArc Model

Table 4 continued

Paper	Model processing technique	Source	Target
[S61]	Model-to-model transformation	CityGML	Cyber-physical spaces (3D topology) model
[S11]	Code generation	MontiGem GUI model	Web app
	Code generation	Process model	Web app
	Code generation	Data model	Web app
[S54]	Model interpretation	Structural model of the PT	Model Synchronizer + Model Manager component
	Model interpretation	Behavioral model of the PT	Model Synchronizer + Model Manager component
[S14]	Code generation	BPMN	Python Drone control code
[S58]	Model interpretation	Causal Directed Acyclic Graphs (DAG)	Inference Engine
[S30]	Model interpretation	OWL Ontology + OWL Individuals	Capability matchmaker
	Model interpretation	AAS	User interface created by the authors
	Model-to-model transformation	AAS	OWL Individuals (MaRCO)
	Model-to-model transformation	OML	UML Stereotypes
[S36]	Code generation	SysML (CSV representation)	SysML (Graph Database entries)
[S60]	Code generation	BPMN Model	Java template code for microservices
	Model-to-model transformation	BPMN Model	DTDL Model
[S26]	Model interpretation	Knowledge base model	Scheduling Engine
	Model interpretation	Knowledge base model	Simulator
-	Model-to-model transformation	Knowledge Base Model	Knowledge Base Model (updated with runtime data)
[S33]	Model interpretation	BPMN	OLIVE platform
[S49]	Model-to-model transformation	BPMN	Activity Diagram
	Model-to-model transformation	Activity Diagram	UML CD
[S20]	Code generation	UML CD	Java Code
	Code generation	MontiArc	System architecture
	Code generation	MontiGem GUI model	DT cockpits
[S22]	Model interpretation	Henshin model	Digital Twin
[S39]	Code generation	Twin Process Model	Numeric Control (NC) Code
[S31]	Code generation	AADL	FMI device mock-ups
	Code generation	AADL	Azure code that is used to execute the system and perform monitoring
	Model-to-model transformation	AADL	Linear No-Threshold (LNT) model
	Model-to-model transformation	SysML	AADL
[S34]	Code generation	MontiThings	IoT controller code in C++
	Model-to-model transformation	MontiThings	MontiThings
[S47]	Model interpretation	Snapshot of PT and DT, represented as UML OD	Trace Alignment Algorithm
	Model interpretation	UML OD (Snapshot); UML CD (snapshot metamodel)	Type Checking Algorithm

Table 4 continued

Paper	Model processing technique	Source	Target
[S46]	Model interpretation	UML CD	UML cd adapter that interprets CD and adapts a data lake accordingly (data lake stores data from the PT)
[S62]	Model interpretation	DT physical model	DT Simulation
	Model interpretation	Behavior model	DT Simulation
	Model interpretation	Geometric model	DT Simulation
[S6]	Model-to-model transformation	BPMN	IoTBPMN
	Code generation	IoTBPMN	eBPMN (Java-based implementation of a BP digital twin)
	Code generation	SysML	Application-specific sensor driver code
[S59]	Model interpretation	Application-Specific Behavior Model	Digital Twin Information System
	Code generation	OPCUA address space model, machine interface model, mapping between OPCUA and machine interface	Kafka/OPCUA client implementation
	Code generation	Class Diagrams	DT Information System
	Code generation	MontiArc models	DT architecture
[S48]	Model interpretation	ADOxx-based DSL for air conditioning facilities	API Gateway
	Model-to-model transformation	DSML for air conditioning	JSON
[S13]	Model-to-model transformation	Goal Structure Notation (GSN) Model	ROS Launch Files
	Model interpretation	GSN Model	Analysis Component of a MAPE-K System
[S28]	Model interpretation	Protocol State Machine	Vulnerability Analysis and Exploitation Component
[S41]	Model-to-model transformation	Data	Configuration model, process rule model base, control logic model base, simulation model base → DT model of welding production line
	Model interpretation	DTD-WPL (DT data of welding production line) model (comprising of production line configuration model, process rule model, control logic model, sub-model base and simulation model)	Simulation Engine
[S8]	Model-to-model transformation	AAS Model	Abstract Syntax Tree (AST)
	Code generation	Intermediate Representation of Abstract Syntax Tree	C# + Python + TypeScript SDK
[S44]	Code generation	Domain Model	Digital Twin Cockpit
	Code generation	Architecture Model	Digital Twin Cockpit
	Code generation	GUI Model	Digital Twin Cockpit
	Code generation	OCL Model	Digital Twin Cockpit
	Model interpretation	Reasoning Model	Evaluator component
	Model interpretation	Process Model	Evaluator component
	Model-to-model transformation	System Description	Explanation Model

Table 4 continued

Paper	Model processing technique	Source	Target
	Model-to-model transformation	Process Model	Explanation Model
	Model-to-model transformation	Reasoning Model	Explanation Model
[S10]	Model-to-model transformation	STEP + ODP Model	OWL Model
	Model-to-model transformation	OWL Model	Feature Model in XML
[S9]	Code generation	AAS Metamodel in Domain-Specific Language	XSD Schema
	Code generation	AAS Metamodel in Domain-Specific Language	JSON Schema
	Code generation	AAS Metamodel in Domain-Specific Language	RDFS Schema
[S55]	Code generation	Device Domain Model	API for communicating with DT/PT
	Model-to-model transformation	Medicine dispenser structure metamodel	JSON
	Model-to-model transformation	JSON	Medicine dispenser structure instance model
[S45]	Model interpretation	UML models	Validation in USE tool
[S17]	Code generation	Object Event Table	Web API, Business Logic and Data Store of DT
	Code generation	Class Diagram	Web API, Business Logic and Data Store of DT
	Code generation	Finite State Machine	Web API, Business Logic and Data Store of DT
[S24]	Model-to-model transformation	Building model, nuclear waste model, sensor network model	Bayesian network model for risk analysis
[S40]	Model-to-model transformation	Physics-based model	Data-driven model
[S32]	Model interpretation	Speech Interaction Model	Speech IO Framework
[S25]	Code generation	Simulation model	Digital Twin Cockpit
	Model interpretation	Simulation model	MoSTHealth Framework
[S63]	Model interpretation	Simulation model	Maintenance platform

Acknowledgements The authors of the University of Stuttgart were partly supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) [grant number 441207927], and the Ministry of Science, Research and Arts of the Federal State of Baden-Württemberg within the Innovations Campus Future Mobility (ICM). The authors of JKU Linz were partly supported by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG).

Funding Open access funding provided by Johannes Kepler University Linz.

References

[1] de Araújo Silva, E., Valentin, E., Carvalho, J.R.H., da Silva Barreto, R.: A survey of model driven engineering in robotics. J. Comput. Lang. **62**, 101021 (2021)
[2] Arora, R., Arora, N.: Analysis of SDLC models. Int. J. Curr. Eng. Technol. **6**(1), 268–272 (2016)
[3] Baheti, R., Gill, H.: Cyber-physical systems. Impact Control Technol. **12**(1), 161–166 (2011)
[4] Barat, S., Kulkarni, V., Clark, T., Barn, B.: Digital twin as risk-free experimentation aid for techno-socio-economic systems. In: E. Syriani, H.A. Sahraoui, N. Bencomo, M. Wimmer (eds.) Proc. of MODELS, pp. 66–75. ACM (2022)
[5] Barn, B.S., Clark, T., Barat, S., Kulkarni, V.: Towards the essence of specifying sociotechnical digital twins. In: Proceedings of the 16th Innovations in Software Engineering Conference, pp. 1–5 (2023)
[6] Barricelli, B.R., Casiraghi, E., Gliozzo, J., Petrini, A., Valtolina, S.: Human digital twin for fitness management. IEEE Access **8**, 26637–26664 (2020)
[7] Bencomo, N., Götz, S., Song, H.: Models@run.time a guided tour of the state of the art and research challenges. Softw. Syst. Model. **18**(5), 3049–3082 (2019)
[8] Birken, K.: Building Code Generators for DSLs Using a Partial Evaluator for the Xtend Language. In: T. Margaria, B. Steffen (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 8802, pp. 407–424. Springer (2014)
[9] Boje, C., Guerriero, A., Kubicki, S., Rezgui, Y.: Towards a semantic construction digital twin: directions for future research. Autom. Constr. **114**, 103179 (2020)
[10] Bordeleau, F., Combemale, B., Eramo, R., van den Brand, M., Wimmer, M.: Towards model-driven digital twin engineering: Current opportunities and future challenges. In: Systems Modelling and Management: First International Conference, ICSMM 2020, Bergen, Norway, June 25–26, 2020, Proceedings, pp. 43–54. Springer (2020)
[11] Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice. Morgan & Claypool Publishers (2017)
[12] Brito, T.B., Trevisan, E.F.C., Botter, R.C.: A conceptual comparison between discrete and continuous simulation to motivate the hybrid simulation methodology. In: Proceedings of the Winter Simulation Conference (WSC), pp. 3910–3922. IEEE (2011)
[13] Brockhoff, T., Heithoff, M., Koren, I., Michael, J., Pfeiffer, J., Rumpe, B., Uysal, M.S., van der Aalst, W.M.P., Wortmann, A.: Process prediction with digital twins. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS, Fukuoka, Japan, October 10-15, 2021, pp. 182–187. IEEE (2021)
[14] Burgueño, L., Ciccozzi, F., Famelis, M., Kappel, G., Lambers, L., Mosser, S., Paige, R.F., Pierantonio, A., Rensink, A., Salay, R., et al.: Contents for a model-based software engineering body of knowledge. Softw. Syst. Model. **18**, 3193–3205 (2019)
[15] Cabot, J., Vallecillo, A.: Modeling should be an independent scientific discipline. Softw. Syst. Model. **21**(6), 2101–2107 (2022)
[16] Casalaro, G.L., Cattivera, G., Ciccozzi, F., Malavolta, I., Wortmann, A., Pelliccione, P.: Model-driven engineering for mobile robotic systems: a systematic mapping study. Softw. Syst. Model. **21**(1), 19–49 (2022)
[17] Cimino, C., Negri, E., Fumagalli, L.: Review of digital twin applications in manufacturing. Comput. Ind. **113**, 103130 (2019)
[18] Cleophas, L., Godfrey, T., Khelladi, D.E., Lehner, D., Combemale, B., Rumpe, B., Zschaler, S.: Model-Driven Engineering of Digital Twins (Dagstuhl Seminar 22362). In: Dagstuhl Reports, vol. 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2023)
[19] Dalibor, M., Jansen, N., Rumpe, B., Schmalzing, D., Wachtmeister, L., Wimmer, M., Wortmann, A.: A cross-domain systematic mapping study on software engineering for digital twins. J. Syst. Softw. **193**, 111361 (2022)
[20] David, I., Archambault, P., Wolak, Q., Vu, C.V., Lalonde, T., Riaz, K., Syriani, E., Sahraoui, H.: Digital twins for cyber-biophysical systems: challenges and lessons learned. In: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 1–12. IEEE (2023)
[21] van Dinter, R., Tekinerdogan, B., Catal, C.: Predictive maintenance using digital twins: A systematic literature review. Inf. Softw. Technol. **151**, 107008 (2022)
[22] Durão, L.F.C.S., Haag, S., Anderl, R., Schützer, K., de Senzi Zancul, E.: Digital Twin Requirements in the Context of Industry 4.0. In: P. Chiabert, A. Bouras, F. Noël, J. Ríos (eds.) Product Lifecycle Management to Support Industry 4.0 - 15th IFIP WG 5.1 International Conference, PLM 2018, Turin, Italy, July 2-4, 2018, Proceedings, *IFIP Advances in Information and Communication Technology*, vol. 540, pp. 204–214. Springer (2018)
[23] Erazo-Garzón, L., Román, A., Moyano-Dután, J., Cedillo, P.: Models@ runtime and internet of things: A systematic literature review. In: 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST), pp. 128–134. IEEE (2021)
[24] Errandonea, I., Beltrán, S., Arrizabalaga, S.: Digital Twin for maintenance: a literature review. Comput. Ind. **123**, 103316 (2020)
[25] European Commission and Directorate-General for Research and Innovation: ERA industrial technologies roadmap on human-centric research and innovation for the manufacturing sector. Publications Office of the European Union (2024). https://doi.org/10.2777/0266

[26] European Parliament: Regulation (EC) No 1893/2006 of the European Parliament and of the Council of 20 December 2006 establishing the statistical classification of economic activities NACE Revision 2 and amending Council Regulation (EEC) No 3037/90 as well as certain EC Regulations on specific statistical domains (2006). https://eur-lex.europa.eu/eli/reg/2006/1893/2019-07-26
[27] Garousi, V., Felderer, M., Mäntylä, M.V.: Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf. Softw. Technol. **106**, 101–121 (2019)
[28] Ghenai, C., Husein, L.A., Al Nahlawi, M., Hamid, A.K., Bettayeb, M.: Recent trends of digital twin technologies in the energy sector: a comprehensive review. Sustai. Energy Technol. Assess. **54**, 102837 (2022)
[29] Goldacre, B., DeVito, N.: Catalogue of bias: publication bias. BMJ Evidence-Based Medicine **24**(2) (2018)
[30] Grieves, M.W.: Product lifecycle management: the new paradigm for enterprises. Int. J. Prod. Dev. **2**(1–2), 71–84 (2005)
[31] Gusenbauer, M., Neal, H.: Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. Research synthesis methods **11**(2), 181–217 (2020)
[32] ISO: ISO 23247 automation systems and integration-digital twin framework for manufacturing. Tech. rep., International Standardization Organization (ISO) (2020)
[33] Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B.: Characterising the Digital Twin: A systematic literature review. CIRP J. Manuf. Sci. Technol. **29**, 36–52 (2020)
[34] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008)
[35] Kan, C., Anumba, C.: Digital twins as the next phase of cyber-physical systems in construction. In: ASCE International Conference on Computing in Civil Engineering 2019, pp. 256–264. American Society of Civil Engineers Reston, VA (2019)
[36] Kirchhof, J.C., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 90–101 (2020)
[37] Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering-a systematic literature review. Inf. Softw. Technol. **51**(1), 7–15 (2009)
[38] Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.: Digital Twin in manufacturing: A categorical literature review and classification. Ifac-PapersOnline **51**(11), 1016–1022 (2018)
[39] Lehner, D., Gil, S., Mikkelsen, P.H., Larsen, P.G., Wimmer, M.: An architectural extension for digital twin platforms to leverage behavioral models. In: 19th IEEE International Conference on Automation Science and Engineering, CASE 2023, Auckland, New Zealand, pp. 1–8. IEEE (2023)
[40] Lehner, D., Pfeiffer, J., Tinsel, E.F., Strljic, M.M., Sint, S., Vierhauser, M., Wortmann, A., Wimmer, M.: Digital twin platforms: requirements, capabilities, and future prospects. IEEE Softw. **39**(2), 53–61 (2021)
[41] Liu, R., Li, H., Lv, Z.: Modeling methods of 3D model in digital twins. CMES-Comput. Modeling Eng. Sci. **136**(2), 985–1022 (2023)
[42] Lo, C.K., Chen, C., Zhong, R.Y.: A review of digital twin in product design and development. Adv. Eng. Informatics **48**, 101297 (2021)
[43] Malavolta, I., Muccini, H., Pelliccione, P., Tamburri, D.: Providing architectural languages and tools interoperability through model transformation technologies. IEEE Trans. Software Eng. **36**(1), 119–140 (2010)
[44] Mehta, P., Rao, P., Wu, Z., Jovanović, V., Wodo, O., Kuttolamadom, M.: Smart manufacturing: state-of-the-art review in context of conventional and modern manufacturing process modeling, monitoring and control. In: International Manufacturing Science and Engineering Conference, vol. 51371, p. V003T02A008. American Society of Mechanical Engineers (2018)
[45] Michael, J., Pfeiffer, J., Rumpe, B., Wortmann, A.: Integration challenges for digital twin systems-of-systems. In: Proceedings of the 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems, pp. 9–12 (2022)
[46] Mihai, S., Yaqoob, M., Dang, H.V., Davis, W., Towakel, P., Raza, M., Karamanoglu, M., Barn, B., Shetve, D.S., Prasad, R.V., Venkataraman, H., Trestian, R., Nguyen, H.X.: Digital twins: a survey on enabling technologies, challenges, trends and future prospects. IEEE Commun. Surv. Tutorials **24**(4), 2255–2291 (2022)
[47] Minerva, R., Lee, G.M., Crespi, N.: Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. Proc. IEEE **108**(10), 1785–1824 (2020)
[48] Modoni, G.E., Caldarola, E.G., Sacco, M., Terkaj, W.: Synchronizing physical and digital factory: benefits and technical challenges. Procedia Cirp **79**, 472–477 (2019)
[49] Mohamed, M.A., Kardas, G., Challenger, M.: Model-Driven Engineering Tools and Languages for Cyber-Physical Systems-A Systematic Literature Review. IEEE Access **9** (2021)
[50] Neethirajan, S., Kemp, B.: Digital twins in livestock farming. Animals **11**(4), 1008 (2021)
[51] Negri, E., Fumagalli, L., Macchi, M.: A review of the roles of digital twin in CPS-based production systems. Procedia manufacturing **11**, 939–948 (2017)
[52] Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: G. Visaggio, M.T. Baldassarre, S.G. Linkman, M. Turner (eds.) 12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008, Workshops in Computing. BCS (2008)
[53] Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. **64**, 1–18 (2015)
[54] Pfeiffer, J., Lehner, D., Wortmann, A., Wimmer, M.: Modeling capabilities of digital twin platforms–old wine in new bottles? J. Object Technol. **21**(3), 1–14 (2022)
[55] Pfeiffer, J., Lehner, D., Wortmann, A., Wimmer, M.: Towards a product line architecture for digital twins. In: 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C), pp. 187–190. IEEE (2023)
[56] Pozsgai, G., Lövei, G.L., Vasseur, L., Gurr, G., Batáry, P., Korponai, J., Littlewood, N.A., Liu, J., Móra, A., Obrycki, J., et al.: Irreproducibility in searches of scientific literature: a comparative analysis. Ecol. Evol. **11**(21), 14658–14668 (2021)
[57] Psarommatis, F., May, G.: A literature review and design methodology for digital twins in the era of zero defect manufacturing. Int. J. Prod. Res. **61**(16), 5723–5743 (2023)
[58] Pylianidis, C., Osinga, S., Athanasiadis, I.N.: Introducing digital twins to agriculture. Comput. Electron. Agric. **184**, 105942 (2021)
[59] Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., Nee, A.: Enabling technologies and tools for digital twin. J. Manuf. Syst. **58**, 3–21 (2021)
[60] Santiago, I., Jiménez, Á., Vara, J.M., de Castro, V., Bollati, V.A., Marcos, E.: Model-Driven Engineering as a new landscape for

traceability management: A systematic literature review. Inf. Softw. Technol. **54**(12), 1340–1356 (2012)
[61] Schluse, M., Priggemeyer, M., Atorf, L., Rossmann, J.: Experimentable digital twins-streamlining simulation-based systems engineering for industry 4.0. IEEE Trans. Industrial Inform. **14**(4), 1722–1731 (2018)
[62] Singh, M., Srivastava, R., Fuenmayor, E., Kuts, V., Qiao, Y., Murray, N., Devine, D.: Applications of digital twin across industries: a review. Appl. Sci. **12**(11), 5727 (2022)
[63] Song, Y., Xia, M., Chen, Q., Chen, F.: A data-model fusion dispatch strategy for the building energy flexibility based on the digital twin. Appl. Energy **332**, 120496 (2023)
[64] Stan, M., Borangiu, T., Raileanu, S.: Data- and model-driven digital twins for design and logistics control of product distribution. In: 23rd International Conference on Control Systems and Computer Science, CSCS 2021, Bucharest, Romania, May 26-28, 2021, pp. 33–40. IEEE (2021)
[65] Szvetits, M., Zdun, U.: Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. Softw. Syst. Model. **15**(1), 31–69 (2016)
[66] Tao, F., Xiao, B., Qi, Q., Cheng, J., Ji, P.: Digital twin modeling. J. Manuf. Syst. **64**, 372–389 (2022)
[67] Tao, F., Zhang, H., Liu, A., Nee, A.Y.: Digital twin in industry: State-of-the-art. IEEE Trans. Industr. Inf. **15**(4), 2405–2415 (2018)
[68] Troya, J., Moreno, N., Bertoa, M.F., Vallecillo, A.: Uncertainty representation in software models: a survey. Softw. Syst. Model. **20**(4), 1183–1213 (2021)
[69] Utting, M., Legeard, B.: Practical model-based testing: a tools approach. Elsevier (2010)
[70] Van Deursen, A., Visser, E., Warmer, J.: Model-driven software evolution: A research agenda. In: Proceedings 1st International Workshop on Model-Driven Software Evolution, pp. 41–49 (2007)
[71] Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: M.J. Shepperd, T. Hall, I. Myrtveit (eds.) 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, London, England, United Kingdom, May 13-14, 2014, pp. 38:1–38:10. ACM (2014)
[72] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
[73] Wortmann, A., Barais, O., Combemale, B., Wimmer, M.: Modeling languages in Industry 4.0: an extended systematic mapping study. Softw. Syst. Model. **19**(1), 67–94 (2020)
[74] Xu, Q., Ali, S., Yue, T.: Digital twin-based anomaly detection with curriculum learning in cyber-physical systems. ACM Transactions on Software Engineering and Methodology (2023)
[75] Yu, W., Patros, P., Young, B., Klinac, E., Walmsley, T.G.: Energy digital twin technology for industrial energy management: Classification, challenges and future. Renew. Sustain. Energy Rev. **161**, 112407 (2022)
[76] Zheng, X., Lu, J., Kiritsis, D.: The emergence of cognitive digital twin: vision, challenges and opportunities. Int. J. Prod. Res. **60**(24), 7610–7632 (2022)
[S1] Banerjee, A., Chaudhuri, S.R., Basak, B., Dhakshinamoorthy, R., Annadurai, S., Subramani, N.: Knowledge driven rapid development of white box digital twins for industrial plant systems. In: IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, October 13-16, 2021, pp. 1–6. IEEE (2021)
[S2] Barat, S., Kulkarni, V., Clark, T., Barn, B.: Digital twin as risk-free experimentation aid for techno-socio-economic systems. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, pp. 66–75 (2022)
[S3] Barosan, I., van der Heijden, J.: Integration of SysML models in a 3D environment for Virtual Testing and Validation. In: Proceedings of the Federated Africa and Middle East Conference on Software Engineering, pp. 39–45 (2022)
[S4] Bibow, P., Dalibor, M., Hopmann, C., Mainz, B., Rumpe, B., Schmalzing, D., Schmitz, M., Wortmann, A.: Model-Driven Development of a Digital Twin for Injection Molding. In: S. Dustdar, E. Yu, C. Salinesi, D. Rieu, V. Pant (eds.) Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings, *Lecture Notes in Computer Science*, vol. 12127, pp. 85–100. Springer (2020)
[S5] Binder, C., Calà, A., Vollmar, J., Neureiter, C., Lüder, A.: Automated Model Transformation in modeling Digital Twins of Industrial Internet-of-Things Applications utilizing AutomationML. In: 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021, pp. 1–6. IEEE (2021)
[S6] Bocciarelli, P., D'Ambrogio, A., Panetti, T.: A model based framework for IoT-aware business process management. Future Internet **15**(2), 50 (2023)
[S7] Braunisch, N., Ristin-Kaufmann, M., Lehmann, R., van de Venn, H.W.: Generative and model-driven sdk development for the industrie 4.0 digital twin. In: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–4. IEEE (2021)
[S8] Braunisch, N., Ristin-Kaufmann, M., Lehmann, R., Wollschlaeger, M., van de Venn, H.W.: Empowering Industry 4.0 with Generative and Model-Driven SDK Development. In: 49th Annual Conference of the IEEE Industrial Electronics Society, IECON 2023, Singapore, October 16-19, 2023, pp. 1–6. IEEE (2023)
[S9] Braunisch, N., Ristin-Kaufmann, M., Lehmann, R., Wollschlaeger, M., van de Venn, H.W.: Generation of Digital Twins for Information Exchange Between Partners in the Industrie 4.0 Value Chain. In: 21st IEEE International Conference on Industrial Informatics, INDIN 2023, Lemgo, Germany, July 18-20, 2023, pp. 1–6. IEEE (2023)
[S10] Caesar, B., Jansen, N., Weigand, M., Fay, A., Rumpe, B.: Extracting hardware reconfiguration models based on knowledge synthesis from STEP files. In: Companion Proceedings of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Västerås, Sweden, October 1-6, 2023, pp. 434–443. IEEE (2023)
[S11] Chaudhary, H.A.A., Guevara, I., John, J., Singh, A., Ghosal, A., Pesch, D., Margaria, T.: Model-Driven Engineering in Digital Thread Platforms: A Practical Use Case and Future Challenges. In: T. Margaria, B. Steffen (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Practice - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part IV, *Lecture Notes in Computer Science*, vol. 13704, pp. 195–207. Springer (2022)
[S12] Chen, X., Kang, E., Shiraishi, S., Preciado, V.M., Jiang, Z.: Digital behavioral twins for safe connected cars. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 144–153 (2018)
[S13] Cheng, B.H.C., Clark, R.J., Fleck, J.E., Langford, M.A., McKinley, P.K.: AC-ROS: assurance case driven adaptation for the robot operating system. In: E. Syriani, H.A. Sahraoui, J. de Lara, S. Abrahão (eds.) MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, pp. 102–113. ACM (2020)
[S14] Chis, A., Ghiran, A., Buchmann, R.A.: Proof of Concept for a Roundtrip Engineering IS for the New Enterprise in the Indus-

try 4.0 Era. In: B. Buchberger, M. Marin, V. Negru, D. Zaharie (eds.) 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2022, Hagenberg / Linz, Austria, September 12-15, 2022, pp. 212–215. IEEE (2022)

[S15] Christofi, N., Pucel, X.: A novel methodology to construct digital twin models for spacecraft operations using fault and behaviour trees. In: Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 473–480 (2022)

[S16] Cimino, C., Leva, A., Ferretti, G.: Ensuring consistency in scalable-detail models for DT-based control. IFAC-PapersOnLine **54**(1), 313–318 (2021)

[S17] Compagnucci, I., Snoeck, M., Asensio, E.S.: Supporting digital twins systems integrating the MERODE approach. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Västerås, Sweden, October 1-6, 2023, pp. 449–458. IEEE (2023)

[S18] Dai, Y., Shi, Y., Zhang, Z., Tao, R., Fang, F.: Digital twins driving model based on petri net in industrial pipeline. In: 2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA), pp. 283–291. IEEE (2020)

[S19] Dalibor, M., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Varga, S., Wortmann, A.: Generating customized low-code development platforms for digital twins. J. Comput. Lang. **70**, 101–117 (2022)

[S20] Dalibor, M., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Towards a Model-Driven Architecture for Interactive Digital Twin Cockpits. In: G. Dobbie, U. Frank, G. Kappel, S.W. Liddle, H.C. Mayr (eds.) Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings, *Lecture Notes in Computer Science*, vol. 12400, pp. 377–387. Springer (2020)

[S21] Di Maio, M., Kapos, G.D., Klusmann, N., Atorf, L., Dahmen, U., Schluse, M., Rossmann, J.: Closed-loop systems engineering (close): Integrating experimentable digital twins with the model-driven engineering process. In: IEEE International Systems Engineering Symposium (ISSE), pp. 1–8. IEEE (2018)

[S22] Eisenberg, M., Lehner, D., Sindelár, R., Wimmer, M.: Towards Reactive Planning with Digital Twins and Model-Driven Optimization. In: T. Margaria, B. Steffen (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Practice - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part IV, *Lecture Notes in Computer Science*, vol. 13704, pp. 54–70. Springer (2022)

[S23] Fend, A., Bork, D.: Cpsa_{ml}: a language and code generation framework for digital twin based monitoring of mobile cyber-physical systems. In: T. Kühn, V. Sousa (eds.) Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Montreal, Quebec, Canada, October 23-28, 2022, pp. 649–658. ACM (2022)

[S24] Giovanni, M.D., Campanile, L., D'Onofrio, A., Marrone, S., Marulli, F., Romoli, M., Sabbarese, C., Verde, L.: Supporting the development of digital twins in nuclear waste monitoring systems. In: G.A. Tsihrintzis, C. Toro, S.A. Ríos, R.J. Howlett, L.C. Jain (eds.) Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 27th International Conference KES-2023, Athens, Greece, 6-8 September 2023, *Procedia Computer Science*, vol. 225, pp. 3133–3142. Elsevier (2023)

[S25] Gorelova, A., Meliá, S., Gadzhimusieva, D., Parichenko, A.: Applying discrete event simulation on patient flow scenarios with health monitoring systems. In: International Conference on Web Engineering, pp. 101–108. Springer (2023)

[S26] Gorodetsky, V., Kozhevnikov, S.S., Novichkov, D., Skobelev, P.O.: The Framework for Designing Autonomous Cyber-Physical Multi-agent Systems for Adaptive Resource Management. In: V. Marík, P. Kadera, G. Rzevski, A. Zoitl, G. Anderst-Kotsis, A.M. Tjoa, I. Khalil (eds.) Industrial Applications of Holonic and Multi-Agent Systems - 9th International Conference, HoloMAS 2019, Linz, Austria, August 26-29, 2019, Proceedings, *Lecture Notes in Computer Science*, vol. 11710, pp. 52–64. Springer (2019)

[S27] Grimmeisen, P., Wortmann, A., Morozov, A.: Case study on automated and continuous reliability assessment of software-defined manufacturing based on digital twins. In: T. Kühn, V. Sousa (eds.) Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Montreal, Quebec, Canada, October 23-28, 2022, pp. 511–518. ACM (2022)

[S28] Guo, J., Zhao, D., Gu, C., Chen, X., Zhang, X., Ju, M.: An enhanced state-aware model learning approach for security analysis in lightweight protocol implementations. J. Cloud Comput. **13**(1), 28 (2024)

[S29] Huang, Y., Dhouib, S., Medinacelli, L.P., Malenfant, J.: Enabling semantic interoperability of asset administration shells through an ontology-based modeling method. In: T. Kühn, V. Sousa (eds.) Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Montreal, Quebec, Canada, October 23-28, 2022, pp. 497–502. ACM (2022)

[S30] Huang, Y., Dhouib, S., Medinacelli, L.P., Malenfant, J.: Semantic Interoperability of Digital Twins: Ontology-based Capability Checking in AAS Modeling Framework. In: 6th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS 2023, Wuhan, China, May 8-11, 2023, pp. 1–8. IEEE (2023)

[S31] Hugues, J., Hristosov, A., Hudak, J.J., Yankel, J.: TwinOps - DevOps meets model-based engineering and digital twins for the engineering of CPS. In: E. Guerra, L. Iovino (eds.) Companion Proceedings of MODELS'20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, pp. 94:1–94:5. ACM (2020)

[S32] Jayaraman, R., Lehner, D., Klikovits, S., Wimmer, M.: Towards generating model-driven speech interfaces for digital twins. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Västerås, Sweden, October 1-6, 2023, pp. 459–464. IEEE (2023)

[S33] Karagiannis, D., Buchmann, R.A., Utz, W.: The OMiLAB Digital Innovation environment: Agile conceptual models to bridge business value with Digital and Physical Twins for Product-Service Systems development. Comput. Ind. **138**, 103631 (2022)

[S34] Kirchhof, J.C., Malcher, L., Rumpe, B.: Understanding and improving model-driven IoT systems through accompanying digital twins. In: E. Tilevich, C.D. Roover (eds.) GPCE '21: Concepts and Experiences, Chicago, IL, USA, October 17 - 18, 2021, pp. 197–209. ACM (2021)

[S35] Kirchhof, J.C., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: E. Syriani, H.A. Sahraoui, J. de Lara, S. Abrahão (eds.) MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, pp. 90–101. ACM (2020)

[S36] Laukotka, F.N., Krause, D.: Supporting Digital Twins for the Retrofit in Aviation by a Model-Driven Data Handling. Syst. **11**(3), 142 (2023)

[S37] Lehner, D., Sint, S., Eisenberg, M., Wimmer, M.: A pattern catalog for augmenting digital twin models with behavior. Autom. **71**(6), 423–442 (2023)

[S38] Lehner, D., Sint, S., Vierhauser, M., Narzt, W., Wimmer, M.: AML4DT: A Model-Driven Framework for Developing and Maintaining Digital Twins with AutomationML. In: 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021, pp. 1–8. IEEE (2021)

[S39] Li, J., Zhou, G., Zhang, C.: Twin Process Model Driven Machining Quality Optimization for Complex Parts. In: 2021 International Conference on Digital Society and Intelligent Systems (DSInS), pp. 218–222. IEEE (2021)

[S40] Li, W., Li, B., Gao, H., Peng, C., Zhang, W., Li, Y., Ji, X.: Survey and practice on architecture and deployment method of digital twin system for intelligent substation. In: 7th International Symposium on Computer Science and Intelligent Control, ISCSIC 2023, Nanjing, China, October 27-29, 2023, pp. 183–190. IEEE (2023)

[S41] Liu, J., Ji, Q., Zhang, X., Chen, Y., Zhang, Y., Liu, X., Tang, M.: Digital twin model-driven capacity evaluation and scheduling optimization for ship welding production line. Journal of Intelligent Manufacturing pp. 1–23 (2023)

[S42] Lyan, G., Jézéquel, J.M., Gross-Amblard, D., Combemale, B.: Datatime: a framework to smoothly integrate past, present and future into models. In: 2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 134–144. IEEE (2021)

[S43] Michael, J., Nachmann, I., Netz, L., Rumpe, B., Stüber, S.: Generating digital twin cockpits for parameter management in the engineering of wind turbines. In: M. Riebisch, M. Tropmann-Frick (eds.) Modellierung 2022, 27. Juni - 01. Juli 2022, Hamburg, Deutschland, *LNI*, vol. P-324, pp. 33–48. Gesellschaft für Informatik e.V. (2022)

[S44] Michael, J., Schwammberger, M., Wortmann, A.: Explaining cyberphysical system behavior with digital twins. IEEE Softw. **41**(1), 55–63 (2024)

[S45] Moreno, N., Toro-Gálvez, L., Troya, J., Canal, C.: Modeling urban digital twins over the cloud-to-thing continuum. In: A. Boronat, A. García-Domínguez, G. Hinkel, K. Lano, M.D. Sanctis, N. Ferry, S. Mosser, M. Wimmer, H. Alfraihi, S.K. Rahimi, J. Troya (eds.) Post Proceedings of the STAF 2023 Workshops TTC 2023, MeSS 2023 and AgileMDE 2023, Leicester, United Kingdom, July 18, 2023 and June 21, 2023, *CEUR Workshop Proceedings*, vol. 3620. CEUR-WS.org (2023)

[S46] Muñoz, P., Troya, J., Vallecillo, A.: Using UML and OCL Models to Realize High-Level Digital Twins. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Fukuoka, Japan, October 10-15, 2021, pp. 212–220. IEEE (2021)

[S47] Muñoz, P., Wimmer, M., Troya, J., Vallecillo, A.: Using trace alignments for measuring the similarity between a physical and its digital twin. In: T. Kühn, V. Sousa (eds.) Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Montreal, Quebec, Canada, October 23-28, 2022, pp. 503–510. ACM (2022)

[S48] Nast, B., Reiz, A., Ivanovic, N., Sandkuhl, K.: A modeling approach supporting digital twin engineering: Optimizing the energy consumption of air conditioning facilities. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Västerås, Sweden, October 1-6, 2023, pp. 479–483. IEEE (2023)

[S49] Niati, A., Selma, C., Tamzalit, D., Bruneliere, H., Mebarki, N., Cardin, O.: Towards a digital twin for cyber-physical production systems: a multi-paradigm modeling approach in the postal industry. In: E. Guerra, L. Iovino (eds.) Companion Proceedings of MODELS'20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, pp. 89:1–89:7. ACM (2020)

[S50] i Palomés, X.P., Tuset-Peiró, P., i Casas, P.F.: Combining low-code programming and sdl-based modeling with snap! in the industry 4.0 context. In: Companion Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, Fukuoka, Japan, October 10-15, 2021, pp. 741–750. IEEE (2021)

[S51] Parri, J., Patara, F., Sampietro, S., Vicario, E.: A framework for model-driven engineering of resilient software-controlled systems. Computing **103**(4), 589–612 (2021)

[S52] Rassõlkin, A., Rjabtsikov, V., Kuts, V., Vaimann, T., Kallaste, A., Asad, B., Partyshev, A.: Interface Development for Digital Twin of an Electric Motor Based on Empirical Performance Model. IEEE Access **10**, 15635–15643 (2022)

[S53] Rassõlkin, A., Rjabtšikov, V., Vaimann, T., Kallaste, A., Kuts, V., Partyshev, A.: Digital twin of an electrical motor based on empirical performance model. In: 2020 XI International Conference on Electrical Power Drive Systems (ICEPDS), pp. 1–4. IEEE (2020)

[S54] Rivera, L.F., Müller, H.A., Villegas, N.M., Tamura, G., Jiménez, M.: On the Engineering of IoT-Intensive Digital Twin Software Systems. In: ICSE '20: 42nd International Conference on Software Engineering, Workshops, Seoul, Republic of Korea, 27 June - 19 July, 2020, pp. 631–638. ACM (2020)

[S55] Sartaj, H., Ali, S., Yue, T., Moberg, K.: Model-based digital twins of medicine dispensers for healthcare IoT applications. Software: Practice and Experience **54**(6), 1172–1192 (2024)

[S56] Schroeder, G.N., Steinmetz, C., Rodrigues, R.N., Henriques, R.V.B., Rettberg, A., Pereira, C.E.: A methodology for digital twin modeling and deployment for industry 4.0. Proc. IEEE **109**(4), 556–567 (2021)

[S57] Shokooh, S., Nordvik, G.: A model-driven approach for situational intelligence & operational awareness. In: 2019 Petroleum and Chemical Industry Conference Europe (PCIC EUROPE), pp. 1–8. IEEE (2019)

[S58] Somers, R.J., Clark, A.G., Walkinshaw, N., Hierons, R.M.: Reliable counterparts: efficiently testing causal relationships in digital twins. In: T. Kühn, V. Sousa (eds.) Companion Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Montreal, Quebec, Canada, October 23-28, 2022, pp. 468–472. ACM (2022)

[S59] Spaney, P., Becker, S., Ströbel, R., Fleischer, J., Zenhari, S., Möhring, H., Splettstößser, A., Wortmann, A.: A model-driven digital twin for manufacturing process adaptation. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 Companion, Västerås, Sweden, October 1-6, 2023, pp. 465–469. IEEE (2023)

[S60] Valderas, P.: Supporting the Implementation of Digital Twins for IoT-Enhanced BPs. In: S. Nurcan, A.L. Opdahl, H. Mouratidis, A. Tsohou (eds.) Research Challenges in Information Science: Information Science and the Connected World - 17th International Conference, RCIS 2023, Corfu, Greece, May 23-26, 2023, Proceedings, *Lecture Notes in Business Information Processing*, vol. 476, pp. 222–238. Springer (2023)

[S61] Visconti, E., Tsigkanos, C., Hu, Z., Ghezzi, C.: Model-driven engineering city spaces via bidirectional model transformations. Softw. Syst. Model. **20**, 2003–2022 (2021)

[S62] Wang, G., Bing, Z., Hou, Z., Guan, Y., Qi, X., Liu, M.: Workshop Management and Control System Based on Digital Twin. In: 2022 8th International Conference on Control, Automation and Robotics (ICCAR), pp. 16–21. IEEE (2022)

[S63] Wang, L., Li, B., Zhang, K., Wang, K., Huimin, H., Huang, J.: Research on digital twin technology of small hydraulic sys-

tem. In: 2023 9th International Conference on Fluid Power and Mechatronics (FPM), pp. 1–9. IEEE (2023)
[S64] Wilking, F., Behringer, M., Fett, M., Goetz, S., Kirchner, E., Wartzack, S.: Concept of a modular and system model driven digital twin for engineering education. In: 2023 18th Annual System of Systems Engineering Conference (SoSe), pp. 1–7. IEEE (2023)
[S65] Yang, X., Liu, X., Zhang, H., Fu, L., Yu, Y.: Meta-model-based shop-floor digital twin architecture, modeling and application. Robotics Comput. Integr. Manuf. **84**, 102595 (2023)
[S66] Yang, X., Ran, Y., Zhang, G., Wang, H., Mu, Z., Zhi, S.: A digital twin-driven hybrid approach for the prediction of performance degradation in transmission unit of CNC machine tool. Robotics Comput. Integr. Manuf. **73**, 102230 (2022)

Daniel Lehner is a PhD candidate at the Institute of Business Informatics - Software Engineering at the Johannes Kepler University Linz, and also working as a self-employed consultant and trainer at TwinTech. His research interests include the application of model-driven engineering techniques and practices to digital twins. For more information, please visit: https://derlehner.at

Jingxi Zhang is a research assistant and PhD candidate at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. His research interests are systems engineering, digital twins, and artificial intelligence. For more information, please visit: https://www.isw.uni-stuttgart.de/institut/team/Zhang-00028/

Jérôme Pfeiffer is a research assistant and PhD candidate at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. His research interests are software language engineering, digital twins, and artificial intelligence. For more information, please visit: https://www.isw.uni-stuttgart.de/en/institute/team/Pfeiffer-00005/

Sabine Sint is a university assistant at the research unit of Building Physics at TU Wien and PhD candidate at the Institute of Business Informatics - Software Engineering at the Johannes Kepler University Linz. Her research interests include SysML-based modeling, execution/simulation of production systems, reverse engineering, data integration, and back-propagation of runtime information. For more information, please visit: https://se.jku.at/sabine-sint

Ann-Kathrin Splettstößer is a research assistant and PhD candidate at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. Her research interests are systems engineering and digital twins. For more information, please visit: https://www.isw.uni-stuttgart.de/institut/team/Splettstoesser/

Manuel Wimmer is full professor and head of the Institute of Business Informatics - Software Engineering at the Johannes Kepler University Linz. His research interests comprise foundations of software engineering as well as their application in domains such as software interoperability, software modernization, and cyber-physical systems. For more information, please visit: https://se.jku.at/manuel-wimmer

Andreas Wortmann is head of the chair "Model-Driven Engineering for Manufacturing Automation" at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. He conducts research on model-driven engineering, software language engineering, and systems engineering with a focus on Industry 4.0 and digital twins. For more information, please visit: www.wortmann.ac or https://www.isw.uni-stuttgart.de/en/