Software and Systems Modeling
https://doi.org/10.1007/s10270-024-01167-z

REGULAR PAPER

Current trends in digital twin development, maintenance, and operation: an interview study

Hossain Muhammad Muctadir[1] · David A. Manrique Negrin[1] · Raghavendran Gunasekaran[2] · Loek Cleophas[1,3] · Mark van den Brand[1] · Boudewijn R. Haverkort[2]

Received: 12 May 2023 / Revised: 30 November 2023 / Accepted: 16 February 2024

Abstract

Digital twins (DTs) are often defined as a pairing of a physical entity and a corresponding virtual entity (VE), mimicking certain aspects of the former depending on the use-case. In recent years, this concept has facilitated numerous use-cases ranging from design to validation and predictive maintenance of large and small high-tech systems. Various heterogeneous cross-domain models are essential for such systems, and model-driven engineering plays a pivotal role in the design, development, and maintenance of these models. We believe models and model-driven engineering play a similarly crucial role in the context of a VE of a DT. Due to the rapidly growing popularity of DTs and their use in diverse domains and use-cases, the methodologies, tools, and practices for designing, developing, and maintaining the corresponding VEs differ vastly. To better understand these differences and similarities, we performed a semi-structured interview research with 19 professionals from industry and academia who are closely associated with different lifecycle stages of digital twins. In this paper, we present our analysis and findings from this study, which is based on seven research questions. In general, we identified an overall lack of uniformity in terms of the understanding of digital twins and used tools, techniques, and methodologies for the development and maintenance of the corresponding VEs. Furthermore, considering that digital twins are software intensive systems, we recognize a significant growth potential for adopting more software engineering practices, processes, and expertise in various stages of a digital twin's lifecycle.

Keywords Digital twin · Semi-structured interview · Digital twin trends

1 Introduction

Digital twins (DTs) have captured the interest of industry and academia in recent years, because of their promise to help to better understand, monitor and improve systems. The concept of DTs often encompasses the notion of a real-world entity and a digital counterpart that mimics certain aspects of the former (Fig. 1). We believe that both industry and academia are playing a crucial role in further developing the DT concept as well as its design, development, operation, and maintenance. Although growing in popularity, the concepts and practices used around DTs show significantly differences. According to Zhang et al. [1], there is no general consensus on the nature of the real-world entity, the required

Communicated by Andreas Wortmann.

✉ Hossain Muhammad Muctadir
h.m.muctadir@tue.nl

David A. Manrique Negrin
d.a.manrique.negrin@tue.nl

Raghavendran Gunasekaran
r.gunasekaran@tilburguniversity.edu

Loek Cleophas
l.g.w.a.cleophas@tue.nl

Mark van den Brand
m.g.j.v.d.brand@tue.nl

Boudewijn R. Haverkort
b.r.h.m.haverkort@tilburguniversity.edu

1 Software Engineering and Technology Cluster, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

2 Tilburg School of Humanities and Digital Sciences, Tilburg University, Tilburg, The Netherlands

3 Department of Information Science, Stellenbosch University, Stellenbosch, South Africa

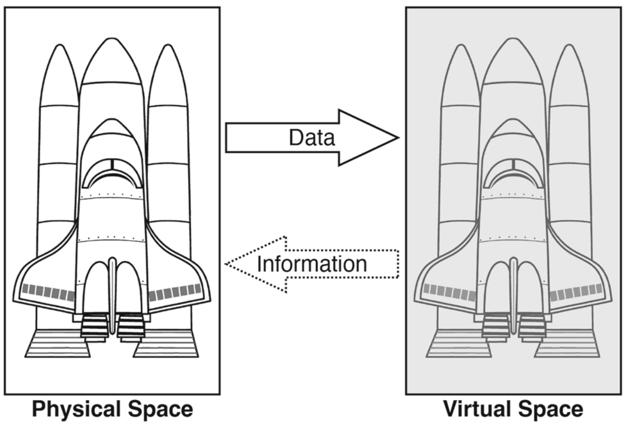

Fig. 1 Depiction of the concept of digital twin as introduced by Grieves [6] where he conceptualized a virtual space or virtual entity (VE) and synchronization with its physical counterpart. Our research focuses on the VE

fidelity of the digital counterpart, or the terms used to refer to these entities.

In recent times, an increasing number of domains are developing and/or using DTs to address use-cases spanning over a large spectrum of complexity. A significant number of these DTs are related to complex industrial systems that utilize cutting-edge technologies to execute complex tasks with an extremely high degree of precision and efficiency. Various cross-domain and heterogeneous models play a central role within these systems, and the use of model-driven engineering (MDE) techniques is paramount in the design, development, and maintenance of these models, which is clearly apparent from the current literature [2–5]. We believe that the role of models and MDE is equally crucial in the development of the digital counterpart or virtual entity (VE) of these physical industrial systems in the context of DTs. Our viewpoint is based on three facts: firstly, models are widely recognized as abstractions of real-world concepts, which aligns with the overarching objective of VEs and, therefore, integration of diverse cross-domain models into a VE is logical; secondly, reuse of existing models and related artifacts can significantly improve the development speed of the corresponding VE; finally, the general benefit of MDE, which makes it increasingly popular [4], also apply to the development of VEs.

As model-based systems, VEs consist of cross-domain and heterogeneous models, which must be (KC1) kept consistent, (KC2) properly orchestrated, and (KC3) validated for correct behavior. Although these activities are identified as key challenges in [7], they have not received sufficient attention in the current literature, as far as we are aware. We believe this shortfall can be attributed to the vast diversity of DTs and their rapid large-scale adoption both in industry and academia. In this research, we aimed to better understand the concepts, practices, and methods around the design, development, and maintenance of VEs. To achieve this, we interviewed 19 individuals from industry and academia who are all involved in the design, development, maintenance, and use of DTs of complex industrial systems. The interviews and subsequent analysis provided us with practical insights into diverse understanding of DTs and practices around the three earlier mentioned challenges. Our work was guided by the following seven research questions (RQ):

RQ 1: How are digital twins defined in practice?
RQ 2: How does reuse of existing (software) artifacts influence the lifecycle of DTs?
RQ 3: How is consistency maintained among cross-domain models used in a DT yet developed independently?
RQ 4: What technologies and methodologies are used to integrate models in a DT?
RQ 5: What practices are used to design and develop the orchestration and data exchange between models in DTs?
RQ 6: What techniques and tools are used to validate a DT and its overall dynamic behavior?
RQ 7: What properties need to be validated in a DT for its consistent dynamic behavior?

RQs 3–7 are directly related to the three key challenges (i.e., KC1–3) mentioned earlier. In contrast, RQs 1–2 intend to understand the perception of DTs in practice and the role of reusing various artifacts in the creation of VE and its overall impact on DT lifecycle. We believe insights gathered based on these two RQs are valuable to understand the insights from the latter RQs and interesting for not only the DT but also the MDE community at large.

We structure this paper as follows: Section 2 describes the background of this research; Sect. 3 elaborates further on our motivation for performing this exploratory research and points out the core contributions of this paper. The detailed explanation of the research methodology we followed is explained in Sect. 4. In Sect. 5, we present a short summary of the domains covered by the interviewees and corresponding DT applications. Based on the analysis of the information, we gathered during the interviews, in Sect. 6 we present our findings related to the above RQs; this section is subdivided into subsections, each one devoted to a single RQ. Additional findings that are not strictly related to the RQs but are relevant in the context of this paper are presented in Sect. 7. Section 8 explains the threats to validity of our research and how we attempted to minimize them. In Sect. 9, we present a discussion correlating findings from individual RQs, recent work that explores various technical aspects of DTs, and our observations. Section 10 concludes the paper.

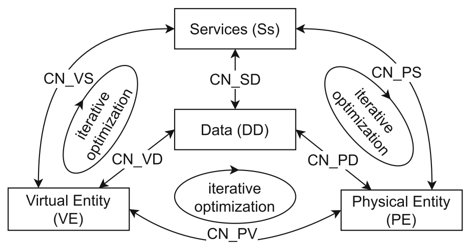

Fig. 2 Five-dimensional (5D) model of a DT proposed by Tao et al. [10]

2 Background

The concept of digital twins (DTs) was introduced by Grieves [6] in 2003; Grieves modeled DTs with three dimensions, i.e., the physical entity, virtual model and connection, which facilitates the physical–virtual interaction. Since then, researchers and practitioners have used DT as an umbrella term to refer to something from a simple simulation to a complex virtual entity closely mimicking a real-world counterpart. For example, Bielefeldt et al. [8] focus on ultra-realistic multi-physical computational models in their definition of DT, whereas El Saddik [9] defines a DT as a digital replica of a physical entity whether living or non-living.

Tao et al. [10] extended the original DT model by Grieves and proposed a five-dimensional (5-D) model depicted in Fig. 2, i.e., M_{DT} = (PE, VE, Ss, DD, CN), where PE refers to the physical (actual) real-world entity with various functional subsystems, VE is the corresponding high-fidelity digital model that reproduces certain abilities and properties of the PE, Ss represents the services for PE and VE, DD encapsulates the domain knowledge—data from both PE & VE and their fusion, and finally, CN is the connection among parts of the DT.

In order to avoid any ambiguity, in the following sections we use the terms actual entity (AE) and virtual entity (VE) to, respectively, refer to real-world entity, which can be an existing or foreseen engineered or naturally occurring physical system or process, and the corresponding digital counterpart.

Since the inception of DTs, research has focused on understanding the concept, the development of DT's applications, or exploration of different implementation technologies used. An example is the work of Liu et al. [11] that analyzes the concept of a DT, the technologies used, and DTs' main industrial applications. That research is based on systematic literature review (SLR) analyzing over 200 publications.

Tao et al. [12] analyze the state of the art in development and applications of DTs, aiming to outline key technologies enabling DT development, classify current and future applications, and lay out possible gaps and challenges. The research used an SLR, analyzing 50 papers and eight patents.

Sharma et al. [13], based on an SLR of over 80 papers, analyzed and proposed solutions for the research and implementation gaps of DT technology, such as IoT (internet of things), machine learning and data. This research concluded that regulation and security mechanisms are essential for the proper implementation of DTs due to its cross-domain nature. They also concluded that there are multiple technical and domain-specific challenges that require more research to be resolved.

Gürdür et al. [14] explore how DTs can help the infrastructure industry. The research methodology used semi-structured expert interviews with non-technical executives from industry in the UK. This approach allowed the researchers to collect their opinions, related to non-technical challenges, on the value of DTs.

Dalibor et al. [15] conducted an SLR on 356 papers, analyzing DTs with a bottom-up approach exploring different implementations to investigate expected DT properties and how DTs are deployed, operated and evaluated. In addition, the authors developed a DT feature model. They explored different implementation techniques, tooling and development processes.

The majority of the research shown above is based on SLRs focused on DT practices and understanding from a high-level systems perspective. The empirical research by Gürdür et al. also approached DTs from a business and high-level systems perspective.

3 Motivation and contribution

Although Grieves [6] introduced the concept of DT more than two decades ago, practitioners are discovering its potential only recently, resulting into an increase of activities and practices around DTs. An increasing number of DTs are being developed and used in domains such as automotive, healthcare, manufacturing, and construction. The application of these DTs extends across a wide spectrum of use-cases, ranging from basic monitoring to highly intricate control systems and predictive maintenance applications. Due to the diversity of use-cases and domains involved, the tools, methods, and practices around the design, development, and maintenance of DTs vary significantly. The central objective of this research is to understand and study this diversity. Our work concentrates on the design, development, and maintenance of the virtual part of a DT, which we refer to as virtual entity (VE) earlier.

VE, which is often described as digital replication of certain capabilities of a physical entity, is essentially a complex software- and data-intensive system that encompasses various heterogeneous and multi-domain models [12, 16]. To maintain the continued operation of a DT, the corresponding models must be kept consistent with each other, they need to be orchestrated properly, and finally, the dynamic behavior of both the individual models and the DT as a whole must be validated. Although these tasks are well-known and researched within the broader model-driven engineering (MDE) community, we are unaware of any work focusing on these topics in the context of DTs. This deficiency motivates our work to interview 19 DT practitioners from industry and academia, in order to acquire insight into current challenges, methods and practices related to the three previously mentioned activities (i.e., KC1–3) in the context of DTs.

Our research was guided by the seven research questions mentioned in Sect. 1. Among these research questions, RQs 1–2 delve into the wide range of understanding of the DT concept and impact of reusing various artifacts on the lifecycle of DTs, respectively. RQ 3 focuses on consistency challenges among heterogeneous and multi-domain models, RQs 4–5 are about the integration and orchestration of these models, and RQs 6–7 concentrate on validation techniques and properties of DTs. Through the interviews and the subsequent analysis of the corresponding transcripts, we identified various practices, methods, and tools related to the seven RQs, challenges faced by practitioners, and implemented measures addressing these challenges. Furthermore, we also discussed unresolved challenges, indicating possible research directions, and the similarities and differences between industrial and academic practices.

4 Research methodology

Considering the exploratory nature of our research questions, we opted for semi-structured interviews [17]. This provided sufficient flexibility for the participants to express themselves while allowing us to collect data on our topics of interest. In this section, we introduce and explain our research methodology, following Strandberg's interview lifecycle [18], with the steps depicted in Fig. 3. We expand on each step and explain the related activities in the following subsections.

In this exploratory study, interviews were conducted by the first three authors of this paper, all PhD candidates; the first with a Bachelor's degree in Information Technology, a Master's degree in Software Systems Engineering, and an Engineering Doctorate in Software Technology; the second with a Bachelor's degree in Chemical Engineering, and Master's degrees in Operational Research and in Automotive Technology; and the third with a Bachelor's degree in Electronics and Instrumentation Engineering and a Master's degree in Embedded Systems.

4.1 Planning

This phase entailed the regulatory activities that our universities required for collecting data from human participants, and preparing a questionnaire consistent with the RQs, serving as guideline during the interviews.

4.1.1 Ethical review and research data management

Ethical review is a process followed by our universities that enables researchers to perform research activities in accordance with accepted ethical standards and existing regulations. This process ensured that measures and infrastructure were in place for maintaining data security and confidentiality as we collected personally identifiable information for (prospective) interviewees and recorded the interviews which potentially contained sensitive information.

4.1.2 Questionnaire design

We prepared a questionnaire to act as a guideline to keep the discussion in our semi-structured interviews focused. It was developed in line with the Interview Protocol Refinement (IPR) framework [19], comprising four phases:

1. **Ensure interview questions are aligned with RQs:** We took an iterative approach. For the first iteration, we listed our initial RQs and from these derived the initial set of interview questions. We tagged the interview questions with corresponding RQs. This allowed us to identify under-represented research questions and adapt the interview questions accordingly. We repeated this step until the questionnaire stabilized.
2. **Constructing an inquiry-based conversation:** We categorized the interview questions into (1) background, (2) key, and (3) concluding ones. Based on this categorization and suggestions of Hove and Anda [20], we sorted them and rephrased some, enabling an inquiry-based conversation.
3. **Receiving feedback on the questionnaire:** We performed several review rounds among the authors of this paper and a pilot interview with a researcher working in the model-driven software engineering domain to check how well participants did understand the questionnaire. Wherever we identified significant difference between interviewee perception and our intention, we rephrased our questions for better understandability.
4. **Conduct pilot interview:** We performed mock interviews with colleagues, allowing us to pilot our ques-

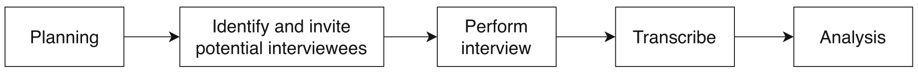

Fig. 3 An overview of our major research activities

Table 1 Interview questions and their relation to the RQs

	Research questions (RQs) →	1	2	3	4	5	6	7
Background questions								
1	What is the domain of your company and what kind of services does it provide?							
2	How long have you been working and what is your current role?							
Key questions								
3	What is your understanding of a DT?	x						
4	How are you involved in the development or usage of the DT?							
5	What problems are you solving with your DT?	x						
6	Is your DT for the entire process/system or a specific part?	x						
7	What are the main parts of your DT? Could you shortly describe their role?	x			x			
8	Is there a physical counterpart of your DT? Does it communicate with the digital world? If yes, how?	x		x				
9	Did you build your DT from scratch or you reused some of the things that already existed? What kind of issues did you face if you reused something or by building from scratch?		x					x
10	How is the data exchange between the models (and physical system) specified?			x	x	x		
11	How is the sequence of execution of the models in your DT?					x		
12	What issues, if any, did you face with the overall collective execution of models in your DT?							x
13	Which platforms/tools are you using to develop the digital twin?			x	x	x	x	
14	How often is the DT updated for bug fixes, improvement or similar reasons?			x				
15	After each update did you face integration issues? If yes, what kind?			x				x
16	How do you ensure various software elements can work together specially if multiple tools were used for development?		x	x	x	x	x	
17	What properties/parameters did you validate to ensure an overall consistent behavior of your DT?							x
18	What tools and techniques did you use to validate these properties and parameters?						x	x
19	What do you consider to be the general characteristics of your DT?	x						
Concluding questions								
20	How do you see the DT evolve in the future to solve additional problems?							
21	What is your opinion on the 5-dimensional DT model from Tao et al. [10]?	x						

tionnaire, receive feedback, and gather experience as interviewers. This helped to further mature the questionnaire.

Table 1 shows the resulting set of interview questions and their associations with the RQs. All the interview questions except for the first two are connected to one or more RQs. These two questions allowed us to start the conversation, get acquainted with the interviewee, and contributed to a conversation-like interview. Furthermore, with the final question we asked the interviewees' opinion on Tao et al.'s 5-D DT model [10] (see Sect. 2). While asking this question, we showed an image of the 5-D model and briefly explained it. To avoid influencing the interviewees during the rest of the interview, we intentionally asked this question at the end.

4.2 Finding interviewees

We aimed to interview practitioners and researchers actively involved in the design, development, maintenance, and use of DTs of complex industrial systems. We started with our own network, which consists of a large number of academics and industrial experts working with such systems, as our universities are located within the industrial hub of the Netherlands with many start-ups, high-tech, and manufacturing companies. We initially created a set of potential interviewees that matched our search criteria. Additionally, we verified the DT-related involvements of these individuals based on their Google Scholar or LinkedIn profile. Furthermore, as we conducted the interviews, we requested participants to propose potential interviewees from their network, which added two individuals to our list. In the end, we invited 25 persons. We received 22 responses, 20 of them positive; in the end, we conducted 19 interviews, (one respondent stopped responding). Among these interviewees, 10 were from industry and 9 from academia. In Sect. 5 we further discuss the demographics of the interviewees.

We emailed potential interviewees to introduce ourselves and explained the purpose of our research. With the emails, we also included a consent form in which we explained details of our research about data processing and our measures for ensuring data anonymization and privacy. This allowed us to create a level of trust with the interviewees. While we did not a priori share our questionnaire with the interviewees to avoid prepared and potentially biased answers, we provided them with a description of our topics of interest to allow them to prepare if they wanted to.

4.3 Performing interviews

As we conducted semi-structured interviews, we did not follow any concrete structure; the questionnaire in Table 1 acted as a guideline. Therefore, the variation in the quantity of interview questions linked to each RQ had no impact on the quantity of information we were able to gather. Our methodology allowed us to ask more questions if necessary.

We prepared standard texts that we read to the interviewee at the beginning and end of the interview. These introduced the interviewers, checked whether the interviewee had any questions regarding the consent they provided earlier, and at the end, thanked them for their participation and requested their feedback on the interview. We recorded video and audio of all 19 interviews. These interviews took place between September 2021 and February 2022.

4.4 Transcription

The interviews altogether accounted for just over 26 h of recorded video with audio. To generate word-to-word transcripts of these recordings, we used automated transcript generation followed by manual verification and revision. As most interviews were conducted and recorded using Microsoft Teams, we could use the generated transcript of the corresponding recording. We manually verified and revised each transcript twice to ensure correctness.

4.5 Data analysis

We further analyzed the transcripts based on the thematic qualitative analysis methodology [21], the activities related to which are explained in this section. Each of these activities utilize outcomes from one or more of the previous activities. Figure 4 depicts the relationships among the activities and the related outcomes.

We used LaMa [22], a web-based tool for collaborative labeling and thematic analysis, to collaborate on this analysis. To restrict access and ensure data privacy, we deployed this platform locally.

4.5.1 Generating and anonymizing artifacts

The aim of this step was to generate a set of *artifacts* from the transcripts of the interviews. We define an artifact as an independent piece of text that focuses on a specific subject and contains sufficient context information for understanding that subject. To generate them, we manually went through each transcript focusing on text spoken by the interviewee and separated text fragments whenever we identified different subjects being discussed. At this stage, we only tried to identify changes in subject, not subjects themselves. It was interesting to see that the change of subject occurred not only when new questions were asked but also while discussing one single question. As we generated these text fragments, we kept sufficient context information for them to be understandable. When this was not the case, we added a few keywords as context, marking such an addition with square brackets, e.g., to indicate that the word "they" (at that point) refers to a "[digital entity and its 3D visualization]". We also generated artifacts by splitting one artifact into two or more during the labeling step, which is explained in the next section. Typically, we split artifacts if we found more than one key message in it. At the end, we had 748 text artifacts of various sizes. Furthermore, we anonymized the transcripts during this manual artifact generation: all personally identifiable information was replaced by unique identifiers that we stored separately for traceability purposes. The anonymization was essential for performing unbiased analysis in later phases of our research.

4.5.2 Labeling of artifact and topic generation

After generating the artifacts for all transcripts, we labeled them using LaMa [22]. We define a label as a short text that sufficiently captures the core message of an artifact. To reduce bias in labeling, each artifact was labeled by two labelers. We resolved conflicting labels by agreeing on one label through discussion. During the labeling process, the labeler could use an existing label or create a new one. In LaMa, these labels were accompanied by a description explaining how and when a label should be used, which was crucial for reuse of existing labels. Moreover, while labeling we encountered artifacts that lacked sufficient information or context. We labeled these artifacts with two predefined labels: *No value* or *Not understandable*. An example is an artifact discussing an interviewee research policy in relation to their clients. This artifact was classified as *No value*, since it does not discuss any information related to DT development.

Once we had over 70% of the artifacts labeled, we started with topic creation in parallel. In this case, we define a *topic* as a clustering of labels that can collectively provide a complete message on a specific subject. We chose an iterative approach to create the topics. Based on our initial overview

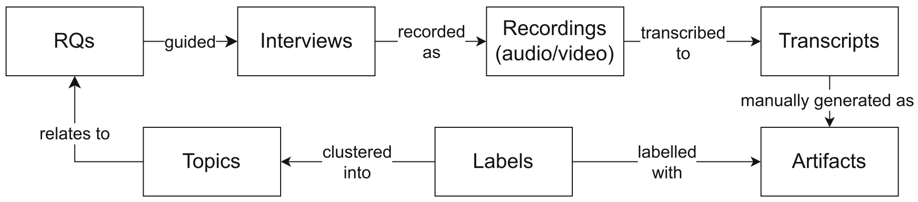

Fig. 4 Relationship among various research activities and related outcomes

of the existing labels, we created our first set of topics. The topic *Different understandings of DT* is one of the first topics we created. We created this topic because we noticed more than 50 labels related to DT definition. These topics and the list of labels were revisited at regular intervals, resulting in one or more of (1) creation of new topics, (2) redefining a topic at a higher level of abstraction, (3) breaking up a topic into multiple topics, and (4) moving labels from one topic to another. We repeated this until reaching convergence.

4.5.3 Relating topics to RQs and perform analysis

In this last step, we focused on answering the research questions. To do that, we created a matrix with the final set of topics and our RQs that enabled us to identify the correlations between the two. This matrix allowed us to aggregate and group relevant information from the topics, corresponding labels and artifacts. From this, we identified opinions, trends, and practices, which formed the foundation for answering the research questions. To reduce individual opinions and bias, we only considered information groups where three or more interviewees contributed their opinions. However, we occasionally deviated from this while discussing challenges mentioned by the interviewees as we were able to identify limited number of challenges related some of the RQs, which is expected considering the infancy of the DT concept. Excluding these outliers would also mean losing the corresponding valuable insights.

4.6 Replication package

The replication package for this interview study is available at [23]. Due to the agreement made with our interviewees, we must keep the interview transcripts confidential. Although this significantly constraints on the reproducibility and validation of the findings presented in this paper, we believe that the provided package can still significantly contribute to similar research endeavors. This package includes:

- **Ethical review form:** We used this form for the ethical review process, which is explained in Sect. 4.1.1.
- **Interview questions:** This document contains the set of interview questions that guided the semi-structured interviews. These are identical to the ones listed in Table 1 and explained in Sect. 4.1.2.
- **Participation consent form:** We used this form for collecting a written and formal consent from the potential interviewees. To do that, we introduced ourselves and explained about this interview study including the purpose of this study, interview procedure, nature of the data we intended to collect, format of the collected data (i.e., audio, video), usage of this data, and confidentiality and privacy measures we put in place to protect this data.
- **Labels and related topics:** This document contains the *labels* and *topics*, which are explained in Sect. 4.5.2, we used to annotate the interview data.

Table 2 Professional domains of the interviewees

Domain	# DTs
Manufacturing and chemical processing	10
High-tech products	9
Building and construction	3
Transport and logistics	3
Information systems	2
Healthcare	1

The number represents the number of DTs in that domain

5 Demographics of interview participants

As explained in Sect. 4, our aim is to interview individuals who are actively involved in the development, maintenance, and use of DTs. Out of the 19 interviewees, ten were primarily from industry and nine from academia. In this section, we provide an overview of their domains and DT applications.

We classified the professional domains of the interviewees into six categories. Seven interviewees claimed that their work involves two different domains and one mentioned being involved in three domains. Table 2 shows the distribution of domains among the 19 interviewees. As visible here there are two dominant categories: the manufacturing and chemical process industry category, and the high-tech products category.

All the interviewees mentioned that they have multiple applications for their DTs. We therefore analyzed the correlation between the domain and the DT's application. For the analysis, we classified the applications in eight categories, as follows (in alphabetical order).

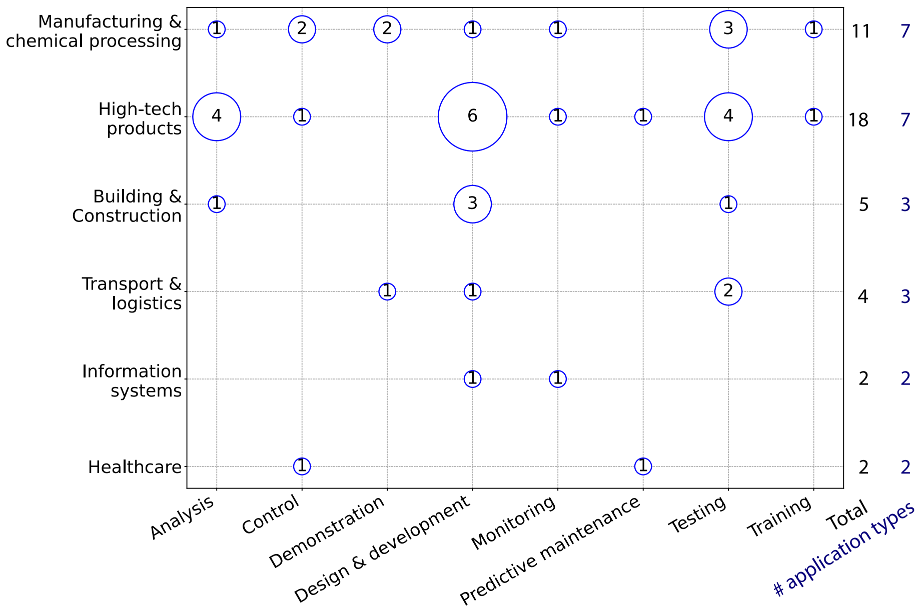

Fig. 5 The correlation among the identified domains (y-axis) of the interviewees and DT's application (x-axis). The numbers on the plot represent the number of DT's applications in that specific domain

- **Analysis** or improvement of the operation of a product or process. Examples provided by interviewees are bug detection, optimization, system behavior analysis and simulation for decision making or process configuration.
- **Control** of a process or a product. Such applications aim to take corrective actions from a monitored state toward a desired one.
- **Demonstration** of alternative solutions or configurations for a physical system. This tends to use visual tools, such as 3D modeling tools.
- **Design and development** of a product (hardware or software) or process. Examples presented by interviewees are prototyping, use of simulation for design and improvements.
- **Monitoring** a property within a process or a product, aiming to compare the current state against a planned one.
- **Predictive maintenance** concerns the supervision and prediction of an equipment of product condition. The aim of such a DT application is to determine the state of health of the supervised entity and anticipate its maintenance.
- **Testing** products or processes; examples shared by interviewees are virtual commissioning, verification and validation or experimentation.
- **Training** operators or users of a specific machine or product.

Figure 5 shows the correlation between the domains of the interviewees and the applications of their corresponding DTs. The *total* value, shown in black (pre-final column), represents the sum of all the applications in a specific domain. The *# application types*, shown in blue (final column), represents the number of application categories of a specific domain. The DTs in the high-tech products and manufacturing and chemical processing domains have the highest DT application diversity, each with seven application types. In these two domains, the most frequent DT applications are design and development, and testing. Furthermore, with the exception of the healthcare domain, all the other domains use DT for design and development. The other two most popular DT applications are testing—used by four domains—and analysis—used by three domains.

The findings presented in Fig. 5 indicate that the prevailing DT applications among the interviewees are design and development, testing, and analysis. It is noteworthy that these DT applications align closely with the backgrounds of a substantial portion of the interviewees, which predominantly have a computer science background.

6 Results and findings

We now present the answers to our seven research questions. These answers are based on the analysis of the data, following the methodology described in Sect. 4, collected during the interviews. Some of the discussions were not strictly related to the defined RQs but still yielded interesting insights; such additional findings are presented in Sect. 7.

6.1 Definitions and understanding of DT (RQ1)

As indicated it in Sect. 1, DT is used as an umbrella term: across different domains it can have many different definitions, interpretations, and understandings. With RQ1 we aimed to understand these, and the similarities and differences between them.

Table 3 Correlation between DT components from interviewees and 5D model [10] addressed by industrial (#I) and academic (#A) interviewees

Interviewees' components	5D model components	#I	#A
Models	Virtual entity	10	9
Data	Data	10	9
AE	Physical entity	4	8
Purpose	Services	8	5
Unidirectional communication: AE $\rightarrow$ VE	Connections	9	9
Bidirectional communication: AE $\leftrightarrow$ VE	Connections	5	6

A variety of definitions of DTs were discussed by the interviewees and there was no uniformity in these definitions. Some interviewees have defined DTs with certain boundaries at the start of the interview, yet over the course of the interview mentioned additional aspects of DT, which extended their initial description. In Sect. 6.1.1, we present views on DTs observed from the interview data.

The different components of a DT discussed during the interviews are presented in Table 3, where the numbers indicate the number of interviewees from academia and industry, respectively, who discussed those specific components of DTs. These numbers are out of the 10 participants from industry and 9 from academia as discussed in Sect. 5; the reader should note that these numbers pertain as maxima for this table as well as for all later tables that refer to '#I' (10) and '#A' (9).

6.1.1 Virtual representation of an entity

We intended to find out how many interviewees agreed to the fundamental understanding of a DT being a virtual representation of some entity. All the interviewees mentioned that a DT is a virtual representation of some entity, by using different terms such as 'digital counterpart', 'digital copy', 'virtual replica', 'virtual prototype' or 'model'. We believe that these terms mentioned by the interviewees refer to a model or a set of models, which we will define in Sect. 6.1.2. Five interviewees used terms such as 'accurate', 'precise' and 'high fidelity' to describe that a DT should be a high-fidelity representation of an entity. According to ten interviewees, the level of fidelity is determined by the DT's purpose and application. When further talking about DTs, interviewees discussed the type of entities the DT could virtually represent. These could be (1) a real-world object with physical dimensions; (2) a real-world process or organization or even a concept without physical dimensions, such as a human resource process, a logistics process in manufacturing, fuzzy concepts and others.

Eight interviewees implicitly or explicitly discussed a DT being a virtual representation of not necessarily just a physical object, but of both physical and non-physical ones, indirectly referring to a virtual representation of an AE. Four interviewees did not explicitly mention whether DTs should be a virtual representation of an AE, but discussed their DTs being a virtual representation of a real-world object with physical dimensions. Ten interviewees also mentioned that DTs need not necessarily be a virtual representation of an existing AE, but could also be of an AE at the design stage. The current confusion in the description of DTs on whether the AE is part of the DT itself or not came up during the interviews. Four interviewees expressed that it is not since the word 'digital' refers only to virtual objects and not physical objects. From the above, it can be understood that there is some level of alignment in the understanding of DT as a virtual representation of some entity which could be physical or non-physical, and which may or may not already exist.

6.1.2 Components of a DT

Through the interviews, we wanted to understand what components interviewees considered part of DTs; two questions were aimed toward this. The different components of a DT discussed during the interviews are listed below. All these components of DTs are represented in Table 3, where the numbers represent the number of interviewees from academia and industry, respectively, who discussed those specific components of DTs.

- **Models:** Here, we consider models as an abstraction of the system which represents a certain viewpoint or aspects of the system such as its behavior, structure, function, performance or others. Interviewees mentioned different types of models, e.g., those representing geometry, physics, behavior and interactions; design and simulation models; descriptive and predictive models; 3D models; mathematical models; mechanical models; building information models (BIMs); CAD models; and others. All interviewees agreed explicitly or implicitly that models are an important component of DTs.
- **Data:** Interviewees also discussed different types of data such as data from sensor measurements; data from system; design data; historical data; reused data from the relevant product line which is in operation; data acquired during DT operation; data from people who are part of the

process; data from subject matter experts; data acquired during the entire lifecycle of a DT; and others. It can be concluded that all interviewees agreed explicitly or implicitly that data is an important component of DTs.

- **Purpose:** Thirteen interviewees expressed that a DT should have a purpose and some mentioned that this purpose is the driving factor for DTs to be developed. On the contrary, one industrial interviewee explicitly mentioned that DTs should *not* have a specific purpose. This interviewee further discussed that DTs should not be developed with a specific purpose and they have a purposeless existence, which is in stark contrast to what the thirteen interviewees mentioned above. This interviewee clarified that once the DT is developed, it can be used for whichever purpose is needed. The purpose mentioned by the different interviewees can be correlated to the 'services' component of Tao et al. [10]'s 5D model of DTs (see Sect. 2).
- **Communication between AE and its virtual counterpart:**
As part of the interview, we intended to understand the level of communication between the AE and its virtual counterpart. All but one interviewee discussed the synchronization from the former to the latter—either automated or manually. Such synchronization implicitly conveys unidirectional communication from AE to its corresponding virtual counterpart. Eleven interviewees discussed that a DT should have bidirectional communication between the two entities. In addition, two interviewees also expressed that the virtual replica should not always be connected to its AE but only when this is needed. Five interviewees shared that the synchronization between AE and its virtual replica should be in real time. However, it was not explicitly discussed further by any of the interviewees what was meant by 'real time' which could possibly have different interpretations in different domains. We identified seven interviewees who mentioned that the frequency of synchronization depends on the purpose or application.
- **AE:** This component is already discussed in Sect. 6.1.1.

6.1.3 Relation to the 5D DT model

As explained in Sect. 4.1.2, we collected and analyzed the opinions of our interviewees on the 5D DT model by Tao et al. [10] (explained in Sect. 2). This 5D DT model was selected in this research because it is a widely used model for DT representation and has been adopted for discussions on DTs by many. So, we wanted to understand the opinions of interviewees on this highly used and adopted model. In this section, we present our observations.

We were able to map some of the DT components mentioned by the interviewees, as discussed in Sect. 6.1.2, to the 5D model. This mapping is shown in Table 3, where the number of interviewees from academia and industry who discussed the specific components of DTs is shown. As shown in that table, we identified considerable alignment between the components mentioned by the interviewees and the physical entity, virtual entity, data and connection components of the 5D model. As discussed earlier, the purpose of a DT mentioned by different interviewees can be correlated to the services component in this model.

Table 4 Agreement on 5D model of DT [10] by interviewees from academia (#A) and industry (#I)

Opinion on model	#I	#A	Total
Agree	5	6	11
Disagree	4	0	4
Neither	1	3	4

As depicted in Table 4, 11 interviewees mentioned that they could relate to the 5D DT model to a certain extent and agreed to this model albeit with some changes. Four interviewees from industry explicitly disagreed with this model. Another four interviewees neither explicitly agreed nor disagreed with this 5D DT model and discussed their perspectives on this model. It is also important to mention here that we were not able to obtain the opinion of one interviewee due to time constraints. Since we do not have data from this interviewee, we consider that this interviewee neither agrees nor disagrees to this 5D model.

Three interviewees suggested that some connections in the 5D model are not needed and might be removed such as the connection between the PE and other components. They further elucidated that the connection from the PE might not be needed in some cases such as when the PE may not exist or when the PE may not be capable of communicating. Two of those three further expressed that some of these connections need not be bidirectional, but can be unidirectional—such as the connection between data and PE, data and VE, and others. Some interviewees also mentioned that the nature of these connections were not clear enough. For example, one academic interviewee explicitly mentioned that this model should also clearly specify what data flows in each direction from one component to another. Some interviewees, albeit not significant numbers, also mentioned the role of humans in DTs and the prospect of DTs interacting through their services; these topics will be covered in Sect. 7.

Overall, it can be concluded that although eleven interviewees agree to the 5D model, there are several changes suggested by them to this model, and thus, they do not completely agree with it. While some of these changes do apply to DTs in general, some changes are also specific to the DT that the interviewee worked on.

6.1.4 Discussion

Based on our findings presented in this section, there is no common understanding of DTs across the nineteen interviewees. It is relevant to mention that this lack of common understanding has been discussed as one of the non-technical challenges in DT development by van den Brand et al. [7], and it is highly important to overcome this. Moreover, we aimed to understand the alignment of interviewees on the different parts that make up a DT, especially the highly relevant ones such as the relation and communication between AE and virtual counterpart. It can also be observed from our findings that interviewees did not agree on all the components of a DT.

Takeaway message: There is no uniformity in the definition of DTs nor in the understanding of the components that make up a DT. This lack of uniformity was also observed for the agreement on the 5D model of DT, even though it has been a widely used and adopted model of a DT. Despite this disparity, some agreement exists on certain components of a DT, specifically, models, data, and the synchronization between the AE and its virtual counterpart. Several changes were suggested to the 5D model of DT, such as removing some of the interconnections; having services interconnected with services from other DTs; adding humans as another dimension in DTs; and others.

6.2 Influence of reuse on DT lifecycle (RQ2)

As indicated by Walravens et al. [24], developing DTs is a cross-domain and resource-intensive task where reuse of existing artifacts can significantly reduce the development and maintenance costs of DTs. With RQ2 our objective was to gain insight into the existing practices of reusing various artifacts for DT development. We identified 15 out of 19 interviewees acknowledging some form of artifact reuse while working with DTs. Industrial participants mentioned reuse more frequently. Based on the interviews, we identified two kinds of artifact reuse: data reuse and software component reuse, which we discuss in Sect. 6.2.1 and summarize in Sect. 5. We also discuss several related challenges that restrict the possibilities of reuse in Sect. 6.2.2. Afterward, in Sect. 6.2.3 we discuss our interpretation of these findings and what insights can be derived from them.

6.2.1 Current practices of reusing artifacts

Software artifact reuse
We use this term to collectively identify the reuse of static and dynamic models and software components developed independently or extracted from one or more separate software intensive systems. Table 5 shows the types of reused software artifacts we identified through our analysis. We identified the following factors that encouraged or motivated software artifact reuse:

Table 5 Mentions of artifact reuse by interviewees from industry (#I) and academia (#A)

Type of reuse	Types of reused artifacts	#I	#A
Software	3D/CAD models	8	5
	Design model		
	Simulation model		
	Software from product line		
	Third-party (commercial) libraries		
	Existing digital twin		
Data	Data (historical) from AE	5	2
	Data from related product line		
	Ontology		
Knowledge	Experiences gathered at work	2	1
	Academic knowledge		

- **Reduced resources for development:** Four interviewees mentioned that reusing software artifacts lead to shorter delivery time and reduced development effort. According to them, specially the ones from industry, reuse is essential as it greatly affects time to market. We additionally identified two cases where DTs were developed based on one or more existing DTs, allowing the developers to leverage existing artifacts and reduce development effort significantly. These identified benefits confirm our assumptions on the benefits of software artifact reuse based on prior publications reporting similar benefits for more traditional software systems [25, 26].
- **Ease of use:** According to six interviewees, ease of use and built-in support for integration provided by corresponding tooling encouraged them to reuse software artifacts. Based on our analysis, we divided these tools and integrated environments into two categories: *commercial* and *in-house tools*. While the commercially available tools are used both in industry and academia, the tools built in-house are exclusively used by the corresponding companies. Among the commercially available tools, *Unity* [27], a well-known physics-based game engine, stood out as six interviewees mentioned to have used it or one of its commercial extension for developing geometry and physics models. Furthermore, our analysis suggests that in-house tools are often developed based on requirements defined by the organizations themselves and therefore are only suitable for their specific needs.

Data reuse
Our analysis suggests that data reuse is an integral part of DT development. Seven interviewees explicitly mentioned it, three did so implicitly. Here we focus primarily on the explicit mentions. We found that the motivation for data reuse is related to the understandability of the data gathered from physical systems and its suitability for modeling certain aspects of the corresponding system into one or more DTs. These DTs are later used for monitoring or enhancing the system. Although the nature, applications, and data sources for such reuse vary significantly, we identified two major types of data reuse:

- **Data from similar systems:** Four interviewees mentioned developing DTs reusing data gathered from similar systems. Three interviewees mentioned reusing data from a product line that is closely related to the DT under development. Our study suggests that this is particularly useful when the DT is being developed alongside a physical system that is not yet mature enough.
- **Ontologies:** We found that both in industry and academia, the use of ontologies facilitates data reuse, especially when the data is produced and consumed in different contexts. In cases like this, ontologies are used to represent knowledge and describe various data properties. Three interviewees claimed that existing ontologies play an important role in their development of DTs. These interviewees are from the high-tech products and the building & construction domain, which are known to be highly multidisciplinary.

Knowledge reuse
We define knowledge as skills and experiences interviewees gather through education, training, and professional activities. It can be argued whether knowledge forms an artifact since it is not tangible and hard to measure. However, we identified three interviewees who emphasized the importance of preserving the experiences and practical knowledge gathered over time.

6.2.2 Challenges in reusing artifacts

Although fifteen interviewees mentioned practicing some form of artifact reuse and acknowledged its positive effects, we often identified cases where such reuse was restricted, most frequently due to:

- **Legal issues:** We found that legal measures or clauses often restrict or prevent artifacts from being reused, especially when industrial stakeholders are involved. These measures include non-conflict of interest agreements (NDAs), intellectual property rights (IPR), and privacy concerns. Liability concerns can also have restrictive consequences, especially when multiple organizations are involved. With four interviewees explicitly mentioning it, we find this to be the most frequent challenge affecting both data and software reuse.
- **Lack of explanation:** Our study suggests that the lack of appropriate description or documentation can severely reduce the possibility of artifact reuse. We found that data reuse is more affected by this. Three interviewees mentioned that various data is often collected to be used for specific purposes. Due to a lack of meta-data and knowledge lost over time, such data becomes meaningless, rendering reuse practically impossible. We identified that poorly documented or undocumented software components suffer from similar problems.
- **Incompatibility and integration issues:** This affects both software and data reuse. For the latter, this is often related to data formats being incompatible with available tooling. We also identified cases where precision and fidelity of available data restrict reuse. Software reuse also suffers from incompatibility issues that restrict the integration of existing components into newer systems. Lack of configurability, lack of interoperability between legacy and newer systems, and interface inconsistency are some of the factors that contribute to this issue. Our study also suggests that lack of proper documentation of software components can lead to perceived lack of interoperability.
- **Lack of methodology or tool:** This issue was identified by four interviewees as the reason behind limited reusability of existing artifacts. According to these interviewees, identifying reusable components and determining the degree of reusability of the identified ones become very difficult due to inadequate consideration for future reusability during the development phase, and lack of appropriate methodology and mindset within the organization. We also found that most industrial DTs are developed using tools that are highly specialized and often built in-house. As a result, artifacts built using these tools are not easily reusable in a different context.
- **Additional effort:** We identified four interviewees acknowledging the need for additional development and validation efforts to facilitate reuse of artifacts. The necessity to adapt existing software components for a new purpose is a major reason for this. Furthermore, newer operating conditions can reveal undetected defects of reused software components warranting further investigation and fixing.

6.2.3 Discussion

Based on the findings presented in this section, we observe that reusing existing artifacts can positively influence the development of DTs. We discussed reuse of software artifacts and data above. Besides these, the DT lifecycle can be significantly influenced by reuse of knowledge. However, we noticed that reuse is not practiced widely in the organizations that we interviewed from. In addition to the discussed reasons and challenges related to this reluctance, lesser mentioned ones include lack of appropriate software engineering skills, loss of knowledge due to personnel changes, and exceptional system requirements. These challenges and the ones listed above were mentioned by the interviewees in the context of DTs. However, we believe all of these challenges are relevant in the larger MDE context. Although resolving these might not be trivial (e.g., due to legal issues), we believe it is possible, with moderate organizational effort, to address issues related to inadequate tooling, lack of reuse attitude, and insufficient software engineering skills. Furthermore, during the interviews we noticed that organizations are recognizing the potential of reuse in the context of DTs and gradually moving toward developing, maintaining, and using reusable software components. We noticed a trend of developing modular or configurable DTs, often using a component-based integrated development environments for developing DTs by combining reusable components.

Takeaway message: Reuse of existing data and software artifacts has the potential to significantly optimize the development lifecycle of a DT. However, except for some limited cases, it is not yet practiced widely due to challenges legal restrictions, inadequate tool support, lack of information, and lack of experience.

6.3 Consistency management of cross-domain models (RQ3)

Cross-domain collaboration is essential for the development and maintenance of DTs. Within a cross-domain environment, we identified maintaining consistency among cross-domain software models as a challenge [7]. With RQ3 we aimed to investigate this challenge. We wanted to understand how consistency is defined in practice and identify key causes for inconsistencies, and tools and techniques used to manage them. Twelve interviewees mentioned that they have encountered or put measures in place to handle inconsistencies. In the following sections, we present our findings based on the analysis of the data we collected from these interviewees.

6.3.1 Inconsistencies encountered in practice

Based on our analysis, we identified several types of inconsistencies encountered in practice. We discuss the four types that appeared most often during the interviews. Table 6 shows an overview with each of the inconsistency types and the number of interviewees mentioning them.

Table 6 Various inconsistencies encountered by interviewees from industry (#I) and academia (#A)

Encountered inconsistencies	#I	#A
Interface inconsistency	2	2
Model format inconsistency	3	3
Representation inconsistency	1	2
AE-VE inconsistency	2	1

- **Interface inconsistency:** Hisarciklilar et al. [28] defined interface inconsistency as mismatching values, terminologies, or schemes among connected interface elements. We identified this inconsistency mostly in cases where two or more models, often cross-domain, need to communicate and do not share any compatible interfaces. Our analysis suggests that this is one of the most frequently encountered inconsistencies, with four interviewees explicitly mentioning it.
- **Model format inconsistency:** Six interviewees recognized such inconsistencies as a challenge. Our analysis shows that the development of DTs is almost always a cross-domain effort. In projects such as these, the stakeholders are from a variety of domains and use domain-specific tooling to develop cross-domain models and artifacts. From the interviews, we identified situations where these tools are completely or partially incompatible. As a result, models developed in one tool cannot be imported to or used in another tool, primarily due to incompatible formats.
- **Design-implementation inconsistency:** As DTs often represent complex cyber-physical systems, various diagrams (i.e., UML [29], SysML [30]) are used to conceptually represent parts or the complete system, usually during the design phase. In our analysis, we identified cases where the actual implementation deviates from the design, which we identify as representation inconsistency.
- **AE-VE inconsistency:** As discussed in Sect. 6.1, the concept of a DT often encompasses a certain degree of synchronization between AE and the corresponding VE to facilitate mimicking certain behavior or features. We identified two types of inconsistency in this context. The first kind is about inconsistent AE-VE communication, which is often a special kind of interface inconsistency (discussed earlier). The other inconsistency occurs when the collective behavior of models within the VE and the behavior of the AE are too dissimilar to be fit for the services the DT needs to offer. As a result, identical opera-

tions performed on both AE and VE can provide different results, rendering the DT ineffective.

6.3.2 Challenges in maintaining consistency

Our analysis of the interviews demonstrates that maintaining consistency among various artifacts in the context of DTs is not a trivial task; interviewees mentioned various challenges they encounter while trying to do so. In this section, we discuss the most prominent challenges we identified. Although this discussion includes a limited number of challenges and existence of more is highly likely, our finding already shows the complexity of maintaining a consistent system.

With five interviewees explicitly mentioning lack of standardization, we identified this as one of the most frequent reasons for both model- and data-related inconsistencies. Two of them mentioned that they are not aware of any standardization within their project, resulting in significant overhead in terms of communication and development efforts. We also identified cases where data was collected, stored, and exchanged using a non-standard format despite the existence of established standards. It was unclear from the interviews why existing standards were not followed. Furthermore, one of the interviewees mentioned that developing and following a set of standards for exchanging or storing information is extremely challenging due to the sheer number of different domains involved in their DT project.

We also identified lack of proper collaboration, insufficient tooling or methodology, and reuse of artifacts as challenges. Although each of these were mentioned only by one or two interviewees, they are potentially broad in impact.

6.3.3 Inconsistency mitigation practices

During the interviews, we tried to understand how inconsistencies are being handled in practice in the context of DTs. Below we discuss the most prominent of the measures we identified:

- **Use of standards:** We found that the usage of standards is one of the key practices for avoiding inconsistencies and maintaining consistency. Our analysis suggests that these standards can be globally accepted or custom-built for an organization. Five interviewees mentioned using one or more of the following standards: Functional Mock-up Interface (FMI), Functional Mock-up Unit (FMU), European Union (EU) defined standard (no further details were provided), and organization-specific customized standard.
- **Use of external tool or technology:** Four interviewees mentioned using external tools for handling inconsistency issues. In this context, we found that semantic technologies play a key role in understanding of data and, in some cases, conversion between different data formats. These interviewees mentioned the usage of ontologies and related technologies, i.e., SHACL,[1] and graph databases, i.e., Neo4j.[2]
- **Formal or informal communication:** Earlier we mentioned general lack of tooling or methodology as one of the reasons for inconsistency. In fact, we believe a majority of inconsistencies are avoided by established practices within an organization, way of working, in-person communication, or personal knowledge. Three interviewees indicated that they often needed to resort to informal in-person communication to resolve inconsistency issues. Our analysis shows that such communication can take place within or across organizations and domain.
- **Use of in-house tooling:** We discussed software-related inconsistencies in the context of multi-tool environments in Sect. 6.3.1. Our analysis suggests that one of the practices for reducing the number of inter-tool inconsistencies is to avoid multiple tools and using a single one. Our analysis identified three individual cases where this approach was employed.
- **Testing:** In safety critical domains, e.g., aerospace and healthcare, early and frequent testing or benchmarking was mentioned as a strategy to identify potential problems including inconsistencies.

6.3.4 Discussion

Our analysis suggests that inconsistencies are actual issues in the context of DTs and directly or indirectly affected over 60% of our interviewees. The nature and source of these issues are highly diverse and depend on the actual DT implementation, involved methods and tooling, organizational and personal constraints. Consequently, we believe that it is fairly impossible to categorize these inconsistencies completely. Furthermore, we identified that inconsistency issues are often not categorized as such. Instead, they are treated as regular issues encountered during system development or maintenance. We think this might have contributed to the inadequacy of related tooling, as discussed before. Furthermore, we identified situations where the vast majority of inconsistency issues are being avoided by simply using a single tool for development, or by developing large monolithic models. Besides, we think there is a large gap between academic innovation and industrial practice in the context of inconsistency management. For example, Torres et al. [31] referred to several academically developed consistency management approaches in their systematic literature review.

[1] Shapes Constraint Language https://www.w3.org/TR/shacl.

[2] Neo4J - a graph data platform https://neo4j.com.

However, we could not identify any commonality between this list and the approaches discussed by our interviewees, which we present in Sect. 6.3.3. Therefore, we believe that there are opportunities for further research and development for consistency management tools in the context of DTs. However, we also identified that certain domains, i.e., aerospace, construction engineering, automotive, are more aware of and mature about handling inconsistency issues. This is largely facilitated by safety requirements and standards or conventions accepted across related organizational entities.

Takeaway message: In the context of DTs, inconsistency issues are common and can adversely affect development and maintenance activities. We identified appropriate communication and usage of standards as the most frequently practiced measures to avoid inconsistencies. Contrarily, we noticed that the absence of these measures are the major reasons behind the emergence of inconsistency issues. Furthermore, we observed a general lack of tooling and methodology for effectively handling consistency. Therefore, we believe that further research, development, and adoption are needed to understand these issues and develop tools and techniques to avoid or mitigate them.

6.4 Model integration (RQ4)

RQ4 explores the topic of model integration in DTs. Model integration is the process of bringing together models to create the DT virtual entity. These models will interact to mimic a desired behavior from the AE. We intended to understand the approaches and design decisions the interviewees take when designing a DT.

6.4.1 Model integration practices

Regarding model integration, an overview of the interviewees' approaches and design decisions are shown in Table 7. The second column shows our classification of the findings.
Integration approaches
From our analysis, we observed two main integration approaches, namely multi-tool (nine interviewees) and single-tool (five interviewees).

- **Multi-tool approach**. This approach is used when a heterogeneous modeling environment is present, in which distinct modeling tools are combined [32]. Each model needs encapsulation resulting in a defined interface to establish communication with other models. The nine interviewees using the multi-tool approach agree that this approach has advantages for cross-domain collaboration because it allows different tools to be used. Other reasons mentioned to use this approach is information hiding of model details also known as model masking, for IP protection and reduction of model complexity. Here, technologies are used so the model becomes a "black box" with only its input and output exposed, and hiding the model details. However, this approach has many challenges such as interface consistency (discussed in Sect. 6.3.1), data and model semantics, and relationship complexity between the models.
- **Single-tool approach**. This approach requires to generate or transform all the models for use with a single software tool. The single tool will perform the execution of all the models, i.e., serves as execution platform. Our analysis showed that the selected execution platform in such cases was MATLAB. In addition, our analysis shows two strategies from interviewees. The first strategy is to use the same tool which is used for model execution for creating the models as well, e.g., two interviewees use MATLAB as their modeling environment and execution platform. The second strategy is to transform the original models into a format which is supported by the execution platform. Interviewees using this strategy mentioned that this requires re-work from them. An example of this practice is the use of Python as an execution platform that can support the execution of models made in Python or MATLAB. If there is a model in Modelica then this model is transformed into MATLAB, the supported platform, requiring extra work from the modeler. Interviewees mentioned that this approach might limit cross-domain collaboration, but significantly reduces the integration effort.

Communication among models
As mentioned previously, an important ingredient of model integration is the communication between models. The communication between models refers to the action of transmitting data between models, particularly between models built in different modeling tools. We aimed to understand how the interviewees implemented such communication. We identified 15 interviewees who addressed this topic. Our analysis showed two key topics mentioned:

1. **Important factors that influence communication design**. We identified three factors influencing model communication, namely the DT's application, software dependencies, and stakeholders' involvement. According to nine interviewees, the DT's application dictates the required communication frequency or the implementation approach. However, they mentioned that stakeholders' involvement is crucial because it will determine the implementation feasibility by providing supporting knowledge, resources, and software. Moreover, they mentioned the importance of knowing the dependencies

Table 7 Overview of model integration discussed by industrial (#I) and academic (#A) interviewees

Topics discussed	Findings classification	#I	#A
Approach	Multi-tool	6	3
	Single-tool	4	1
Communication among models	Design considerations	5	4
	Implementation approach	5	1
Technology used	Functional Mock-up Interface (FMI)	3	0
	Domain Specific Language (DSL)	1	2
	Own-design	4	0
	Tool provided	2	2
Tooling type	In-house	7	1
	Commercial	2	4

and requirements of the software modeling tools or platforms involved, to avoid operational failure.

2. **Implementation approach**. We identified two main approaches to implement the communication, namely use of standardized communication protocols and in-house technology. Five interviewees shared that they use standardized communication protocols. Three of them use OPC [3] (Open Platform Communication). The second approach used by remaining is an in-house technology similar to a publish-and-subscribe pattern for data exchange.

In conclusion, our analysis shows that the most popular implementation approach is the use of communication protocols, used by five out of the six interviewees who addressed this topic.

Technology used

During the interviews, we aimed to understand the type of technologies used for model integration. Fourteen interviewees described different technologies that are used for this purpose. Table 7 shows a summary of this subsection, where the technologies used varied greatly, but can be clustered in four groups: Functional Mock-up Interface (FMI), DSL, own-design and tool-provided technologies.

- **FMI**: Three interviewees specifically mentioned the use of FMI technology to integrate their models. This technology has been used in DTs [33] and is supported by over 170 modeling tools.[4] According to three interviewees from industry, two reasons to use this technology are its maturity and compatibility with various modeling tools.
- **DSL**: Three interviewees mentioned the use of DSLs for model encapsulation and for establishing communication between these models. Interviewees also use this technology for other aspects such as DT architecture (see Sect. 7) and to unify data semantics among the models.
- **Own-design**: Four industrial interviewees indicated that they developed their own technology to integrate models in their DTs. Their technology is based on developing the interfaces for each modeling tool they have used, e.g., if the interviewees have models in the MATLAB and ANSYS modeling tools, they design an interface for each of them. According to these interviewees, this method gives them flexibility, and it can be expanded according to their needs. However, they admit that it requires effort and time every time a new modeling tool is added.
- **Tool provided**: Another four interviewees mentioned that they use what is supported by their execution platform. As with the monolithic approach explained in Sect. 6.4.1, these interviewees have two choices: either transform incompatible models or develop them in a format supported by their execution platform. According to these interviewees, the main reason to choose the technology is because of their experience with the execution platform.

Tooling type

The interviewees mentioned two distinct uses of tooling. First, tools to develop models for their DTs, for which they all mentioned using commercial software such as MATLAB. Second, tools to execute all the models, as discussed in Sect. 6.4. We divided the identified execution platforms into two categories: developed in-house and commercial. Table 7 shows a preference for the development of in-house tooling among the interviewees, particularly industrial interviewees. Six interviewees did not mention the tooling used for DT integration.

- **In-house**: The data collected suggest that interviewees from industry prefer to develop their own tools for DT development. The main driver to do so seems to be the DT's application and its domain requirements. Two inter-

[3] https://opcfoundation.org/.

[4] https://fmi-standard.org/.

viewees mentioned the need for specific execution and modeling requirements for their event-based DTs, driven by the DT application needs. Another two based their tooling on visual models, which required specific communication technologies and the integration with specific software tooling from their stakeholders. The final four interviewees mentioned that they use modeling tools (commercial and in-house) frequently used in their own domain and thus required to work together. This resulted in the need to create their own tool for integrating models. To summarize, industrial interviewees seem to prefer to develop their own tools to realize their specific integration requirements.
- **Commercial**: The use of commercial tools varies between modeling (e.g., MATLAB), CI/CD (Continuous Integration/Continuous Development), cloud frameworks, and system design tools. According to the interviewees, the main reason for their use is the efficiency of component integration. However, commercial tools have limited support for external software. As a consequence, these interviewees are forced to transform their models to a supported format; this might imply re-work or limit collaboration.

6.4.2 Challenges in model integration

Seven interviewees shared the challenges they face when integrating models for DTs. Based on our analysis, we decided to classify these challenges in three groups, namely challenges in model heterogeneity, data heterogeneity, and complexity.

- **Model Heterogeneity:** These challenges are related to the types of models that need to be integrated into the DT. We identified two challenges related to models: integration of legacy models, extracted from legacy code, and integration of models from different software platforms. Three interviewees mentioned that these challenges seem to be particularly difficult and require further research to address them.
- **Data Heterogeneity:** The data heterogeneity challenges refer to data format and semantics; both challenges are present due to cross-domain collaboration. An example of a semantics challenge is when two terms referring to the same concept are used in different domains, such as "pressure drop" and "pressure gradient". Different terminology can generate confusion which might impact the development. According to our analysis, interviewees seem to find ways to address this challenge. For the data format challenge, a solution is the development of a communication layer to homologize data formats between models. For the semantics challenge, an interviewee uses semantic web technology to standardize the semantics.
- **Complexity:**The complexity challenges are related to models and the DT as a whole. Two interviewees defined complexity of a model as the level of fidelity. These interviewees suggest considering the purpose of the DT as the key factor for design. Therefore, the critical aspect lies in choosing the appropriate level of model fidelity to prevent the need for integration rework. The second complexity challenge discussed is related to the system as a whole. Two interviewees stated that a DT can be composed of several components, increasing the complexity, and hence leading to difficulties in understanding the relation between the components. In addition, understanding those relations becomes critical to solve integration issues.

6.4.3 Discussion

Based on our analysis, the multi-tool approach for integration seems to be most popular among the interviewees. Our opinion is that the multi-tool approach offers better maintainability and cross-domain cooperation, due to the separation of entities. On the other hand, interviewees agreed that this approach requires more effort and knowledge of software engineering. Among the interviewees, the most popular approach for model communication is using communication protocols such as OPC. According to our analysis, the selection of communication seems to be influenced largely by the experience of the developer. Through the interviewees, we found diverse technologies used for integration. However, two technologies were mentioned by three interviewees each: DSLs and FMI. Regarding tooling for integration, seven out of nine industrial interviewees prefer in-house tooling. Our analysis of the challenges suggests that tooling and technologies to facilitate cross-platform integration are required.

Takeaway message: The preferred approach for the integration of models is a multi-tool approach which requires interface development. The preferred technology for such interfaces seems to be the use of standardized communication protocols. Although there is no clear preference for integration technology, two technologies seem to be frequently used, FMI and DSL. Finally, integration has three main challenges: model heterogeneity, data heterogeneity, and DT's complexity.

6.5 Model orchestration (RQ5)

We define DT model orchestration as the definition of the required communication actions and correct execution sequence of the models [34]. To achieve this, the orches-

Table 8 Overview of model orchestration discussed by industrial (#I) and academic (#A) interviewees

Topics discussed	Findings classification	#I	#A
Description	Definition	9	2
	Components	4	1
Implementation approach	Pragmatic	2	4
	DT's Application specific	6	1
Technology	Model based	3	2
	Own design	3	0
Tools	In-house	6	0
	Commercial	2	5

tration should include activities such as data compatibility checks and starting and ending model execution [35]. With RQ5 we aimed to understand interviewees' perceptions and practices related to such orchestration in DTs.

6.5.1 Model orchestration practices

We identified five main topics of discussion related to model orchestration, namely understanding, implementation, technology, tools, and challenges. An overview of those topics is shown in Table 8's first column. The second column shows our classification of the findings to facilitate reading the results. The third column shows the number of interviewees who discussed an item from a specific class.
Understanding
We divided the discussion into two topics: interviewees' explanation of what orchestration is, and of its components.

- **Definition**. Eleven interviewees shared their definitions of model orchestration. They all agree that orchestration is the scheduling of model execution in their DTs. Only four of them specifically mentioned that the method of data exchange is part of the orchestration. In addition, these eleven interviewees also expressed their opinion on the importance of orchestration. From that we concluded that model orchestration is highly important, as three interviewees explicitly expressed it and another six implicitly did so. Yet two interviewees argued that orchestration is not needed in their DTs, because the complexity of their current DTs is not high.
- **Components**. Five interviewees specifically shared the necessary components to design the orchestration of models. All other interviewees mentioned that their orchestration implementation is dependent on purpose and domain; thus, they did not define specific components for orchestration. Table 9 shows the components mentioned by the five aforementioned interviewees and the number of mentions for each component. Concerning the trigger, eight interviewees explicitly stated that it is a key component of orchestration. Nevertheless, the type of trigger depends on the DT's application and domain. We observed two distinctive trigger definitions as a function of the DT's application. Two interviewees working on control applications stated that the orchestration should be done based on a time schedule, where the data exchange between models and the execution of each model should be synchronized based on a global clock. Another interviewee, working on event-based applications, mentioned that the definition of the trigger for each event is the most important factor to schedule each model execution step. The remaining five interviewees explained that the trigger for model execution depends on DT's application and domain.
Regarding the scheduling approach, interviewees mentioned two types, namely sequential scheduling and concurrent execution. Regarding the data exchange method, interviewees defined it as the scheduling of data exchange between models, e.g., First In, First Out. Our analysis shows two different roles of time in orchestration. The first role is as a trigger for model execution in control applications, known as time-based scheduling. The second role is event recording in event-based applications, by using time stamps for each event (cf. Table 9).

Table 9 Components to define orchestration in DTs

Components	Mentions
Trigger	8
Scheduling approach	4
Data exchange method	4
Global time	3
Time-stamping	3

In conclusion, our analysis shows a general consensus on orchestration as all activities ensuring correct scheduling of model execution. The majority of the interviewees consider orchestration important for the development of DTs. These interviewees also agree that the orchestration design requires a definition of the scheduling and method for data exchange. In addition, they agree to define a global time for the DT application and labeling produced data with timestamps. Other components for the orchestration design seem to depend on the DT's applications and domain.
Implementation approach
Thirteen interviewees shared their specific implementation method. Six interviewees stated to use a pragmatic approach, while the other seven shared specific implementation approaches, depending on their domain.

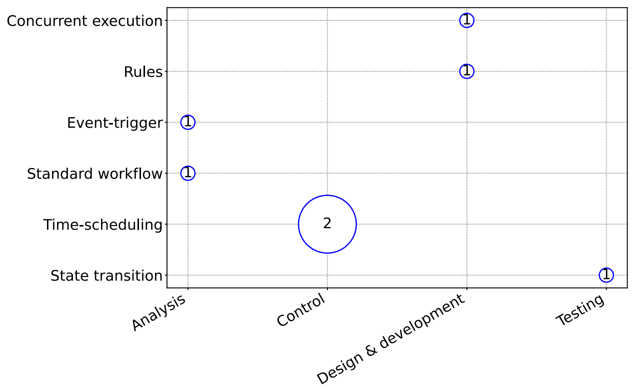

Fig. 6 DT's application specific approaches for model orchestration

- **Pragmatic**. Our analysis suggests that the pragmatic approach aims to design the scheduling of models by reproducing the AE behavior, e.g., if two tasks occur simultaneously, then concurrent execution is used. According to the interviewees, this approach requires iterative testing, design, and implementation. Our analysis shows that interviewees using the pragmatic approach design the orchestration by directly writing code to define the scheduling of models.
- **DT application specific**. We found seven interviewees who stated that the DT application determines the orchestration approach. During our interviews, we collected six different approaches, shown in Fig. 6. Each approach defines how to implement the models' scheduling. The only approach used by multiple interviewees, with control applications, is time-scheduling, in which time triggers execution for each model. In addition to the DT application, the orchestration approach seems to be related to the knowledge and domain of the interviewee. Figure 6 shows two examples. The first example is related to Design & development, which has two distinct approaches 1) rules, and 2) concurrent execution; these approaches are chosen based on their knowledge. The second example is on Analysis, which also has two approaches: 1) standard workflow and 2) event-trigger, chosen based on the domain.

In conclusion, the interviews indicate that the approaches are a function of the DT's application, and of interviewee knowledge and domain. In addition, around half of the interviewees seem to design the orchestration by attempting to pragmatically reproduce the AE behavior.

Technology used

Ten interviewees shared the specific technologies they used for orchestration. We have classified these into two categories: model-based and own design.

- **Model-based**. Five interviewees stated their preference in using technologies that are model-based to design the orchestration. The two technologies described by these interviewees were DSLs and ontologies. Four interviewees use DSLs to design the scheduling of the models. Three of them defined their own DSL, while another uses SysML. One of them also uses their DSL for system verification. Another interviewee uses an ontology to link data between models and thereby orchestrate data exchange.
- **Own design**. Three industrial interviewees explained that they designed their own technology for orchestration. The technology is based on their expertise and domain. None of them explained their technology in detail but rather shared how it works at a higher abstraction level. We identified two distinctive technologies based on the trigger type, i.e., time or event-based triggers.

In conclusion, our analysis suggests that the most popular technology is model-based, particularly DSLs. We observe that all orchestration technologies focus on the scheduling of the models, but not on how to exchange data among them.

Tool type

From 19 interviewees 13 shared the tools they use, in particular their execution platform, which performs the orchestration in a DT. Six interviewees from industry developed their own tool, while seven (five from academia and two from industry) use a commercial tool.

- **In-house**. Six interviewees explained that they have developed their own execution platform to schedule model execution. Based on our findings, we observe two main reasons for in-house tool development: (1) to minimize integration effort, for example use C# to create models and as an execution platform, and (2) to support specific execution requirements, such as specific execution time requirements, like simulate a model with specific time configurations.
- **Commercial**. We identified three types of commercial tools used as execution platforms. The first is to use modeling tools to orchestrate the models, e.g., MATLAB. This type of tool is used by two interviewees who also use the single-tool approach for the integration of models as explained in Sect. 6.4.1. The second type is the use of system design tools such as IBM Rhapsody [36] and HEEDS,[5] used by two interviewees. These tools facilitate the use of external software as long as they are supported by the vendor. The third type encompasses tools that support a DSL to sequence entities for execution; this tool type is used by two interviewees. The tools mentioned

[5] https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-heeds.html.

by interviewees are PDDL (Planning Domain Definition Language) and Dezyne from Verum.[6] The latter also supports formal verification.

Based on our findings, we observed that the use of commercial tools is slightly preferred over the development of in-house tools. Interviewees did not specifically mention why they preferred commercial tools, but three implied some reasons: previous tool knowledge, external software support, and facilitating DT system integration. The last reason was indicated by the interviewees who use a modeling tool for orchestration because they considered it easier to transform all models into a single modeling tool than to orchestrate cross-platform models.

6.5.2 Challenges in model orchestration

Through the interviews, different challenges were mentioned by interviewees. We have classified them in two groups: model fidelity and interoperability. From 19 interviewees, nine mentioned such challenges.

- **Model fidelity:** We found three interviewees addressing this challenge. The fidelity of a model is defined by interviewees as the level of accuracy between the AE and its model. This challenge particularly deals with conflicting requirements between real-time execution and high-fidelity models.
- **Interoperability:** The main challenge discussed by four interviewees is related to the cross-platform and heterogeneous nature of models for DTs, which yields semantic challenges.

6.5.3 Discussion

Our analysis suggests that the majority of interviewees agree that orchestration is to correctly schedule models' execution. In addition, there are five key components to implement the orchestration: a trigger for model execution, a scheduling approach, data exchange method, global time, and timestamps as shown in Table 9. We observed various scheduling approaches that are highly influenced by the DT's application and developers' knowledge. We believe that to facilitate orchestration design, more research should be done to create a general approach.

Technology selection for orchestration seems to favor model-based approaches, with DSLs as the most popular. Our findings suggest that influencing factors for tool selection are the interviewee's domain and previous knowledge of specific tools.

[6] https://www.verum.com/DiscoverDezyne.

Table 10 Different verification and validation techniques and strategies used by interviewees from industry (#I) and academia (#A)

Topic	Findings	#I	#A
Techniques	Comparing AE and VE behavior	8	5
	Formal mthods	2	1
Strategies	Increase complexity gradually	2	1
	Continuous validation	2	1

We believe that the challenges related to model fidelity and model understanding can be tackled by clearly defining a DT's purpose and developing or modifying the models accordingly. Regarding the interoperability challenge, we believe that research on tools to facilitate interoperability, particularly in cross-platform and model-type interoperability, can tackle this challenge.

Takeaway message: The main task of orchestration is to correctly schedule models' execution and a key component is the execution trigger. The orchestration design seems to require much domain knowledge and is highly influenced by the DT's application and the developers' knowledge. Technology and tool selection is also highly influenced by DT's application. Further research is needed on tools to facilitate the interoperability of cross-platform and cross-nature model types.

6.6 Validation and verification techniques and tools (RQ6)

RQ6 aims to understand what specific techniques and tools are used to verify and validate DTs and their overall dynamic behavior, that is, the behavior observed during the collective execution of the models and other components in VE. We identified 18 out of the 19 interviewees validating their DT. It was observed that when answering the related interview questions, interviewees tended to use 'system' to interchangeably describe either the DT or the VE or the AE. We found three different techniques used by the 18 interviewees for validating their DT, namely by comparing the AE and the VE behavior, use of formal methods, and testing.

These techniques are discussed in detail in Sect. 6.6.1. Moreover, interviewees discussed the different challenges involved with verification and validation of DTs which we cover in Sect. 6.6.2. Table 10 summarizes the different verification and validation techniques and strategies put forward by interviewees.

6.6.1 Verification and validation techniques

We believe validation of DTs to be highly important, as it reduces the possibility of errors while executing. Furthermore, our analysis suggests that validation is required in cases where a highly accurate DT is being used. Such highly accurate DTs of critical modules are then reused for different purposes across product lines, hence, across multiple DTs.

The observations from the interview may not necessarily encompass all aspects of DTs which need to be validated, but only those the interviewees stumbled upon in their DT or which they consider of highest importance from their standpoint. We observed that validation of DTs is highly dependent on their purpose, the DT's application domain, and the types of models and data used in the DT. For example, an interviewee from industry mentioned that when DTs are created as visualizations for marketing, validation is not required. However, he also mentioned that when high fidelity, consistent behavior, and reliability in DTs are required, validation is crucial. Interviewees discussed various techniques for verification and validation based on different cases as depicted in Table 10; we next present our findings on these.

- **Comparing AE and VE behavior** This technique concerns behavior comparison between VE and AE, with the aim to check differences. Thirteen interviewees discussed this informal validation technique. This is done in different ways at different abstraction levels. From the interviews, we observed that one way is using observational tests at high level, where the behavior of AE and VE are observed together using 3D visualization and checked for synchronicity and differences. At times, specific inputs or measurements from AEs are given to both AE and VE and behavior matching is checked. We further observed that in cases of DTs used for predictive maintenance, validation is done by initially observing the VE-based predictions and observing and comparing the output of the AE to those later on to check the accuracy of the predictions.
  Moreover, we identified that deeper observational tests are done by creating visual representations such as graphs or 3D visuals of the behavior of both AE and VE and superimposing them to observe the extent of overlapping and differences. In other cases, AE and VE behaviors are translated into events and actions in a Gantt chart [37]. The timing and sequence of actions are compared between AE and VE to check if there are any differences. When there is a combination of continuous and discrete behaviors, continuous signals are transformed into discrete ones and then the behaviors of AE and VE are compared to check for equivalence.
  There are some challenges with this type of validation. Being dependent on measured data from the AE, it is unreliable according to two interviewees, due to incorrect data stemming from measurement errors, faulty equipment, or incorrect interpretations. Moreover, we speculate that there could be other issues in a DT such as consistency issues at runtime which may not be discovered by the aforementioned methods.
- **Formal methods and tools:**
  We identified two interviewees from industry and one interviewee from academia who mentioned using formal methods in DT development. Formal verification techniques have been used to validate the behavior of DTs with the help of tools such as Verum's Dezyne [38] and Coco.[7] An interviewee from industry mentioned that DSLs have been used to specify the behavior of a system and transform such specifications into timed automata models in UPPAAL [39], allowing model checking. However, model checking is not always scalable enough, considering state space explosion [40]. The interviewee discussed that in order to address this challenge, recurring behavioral patterns were identified and validated using model checking, rather than the entire system. In this way, some level of correctness guarantees was provided. Another interviewee mentioned that formal methods have been used for consistent execution of models in DTs: formal semantics for such execution were defined and were helpful to understand differences in execution between models. He also shared that for their model interfaces, they formally proved that the components adhere to the interfaces to avoid interface violations caused by component changes. He mentioned that this ensured consistent behavior when integrating components. As witnessed by the above, three interviewees mentioned formal techniques for validating DTs; no others did. We speculate that the lower adoption of this method could be attributed to scalability issues.
  As mentioned before, formal techniques such as model checking are not scalable, due to state space explosion problems [40]. In addition, we believe that the complexity of formal methods and engineers' lack of background in them, could possibly contribute to the lower adoption of these techniques.

Testing and corresponding tools

Apart from these two categories of techniques identified and quantitatively analyzed by us, DTs also often undergo testing across their entire lifecycle, in order to check adherence to requirements. Interviewees used the term 'testing' to discuss two different items, namely, using scripts to test the system and comparing the AE and VE behavior using observational tests. Due to this lack of clarity, we did not perform quantitative analysis of interviewee responses for testing. As

[7] https://cocotec.io/.

mentioned earlier, in some domains such as the space domain, testing is highly time and resource intensive. In such cases, careful consideration is needed on when and what aspects to test, based on the DT and its context. Moreover, in these cases, testing effort is then balanced with effectiveness. For instance, after resolving an integration issue, only local tests are done. On the other hand, full regression testing would be executed when replacing an entire sub-component. An observation worth mentioning here is that this is not only specific to testing DTs, but generally used in the context of testing software systems. Different types of tests have been mentioned by interviewees such as model-based testing, integration testing, and unit testing. In some cases, static analysis is also performed to detect coding errors, thus helping gain confidence about the system. Some tools mentioned by interviewees are Axini's,[8] Matlab [41], and Unity [27] which are used for creating and testing models on the fly.

6.6.2 Challenges in validation of DTs

Some of the general validation challenges of DTs are presented here. While quite a few of these challenges were not mentioned by three or more interviewees, we still feel they were relevant to discuss here. Two industrial interviewees expressed that validating a DT is much harder than validating a real system and it is infeasible to validate every aspect of a DT due to its numerous degrees of freedom and myriad of parameters. Our study suggests that in some domains, such as the space domain, testing is more resource- and time-intensive than development. An academic interviewee speculated validating a DT to be challenging because the composition of multiple models and the resulting emergent behavior complicate matters. Furthermore, interviewees discussed challenges with validating multi-physics models owing to their complexity and thus not being able to allow real-time execution.

Apart from the challenges discussed in the interviews, we identified that the continuous evolution of VEs pose an additional challenge for validation of VEs. This is because of its continuous synchronization with AE, and others as discussed by Zhang et al. [1], which requires specific techniques for continuous validation of VE as it keeps evolving [7].

6.6.3 Strategies for DT validation and to facilitate validation

We list below the strategies which interviewees discussed for validating DTs and for facilitating such validation.

- **Validation by increasing complexity:** We found three interviewees who advocated an informal bottom-up DT validation, gradually increasing complexity. Even when comparing the behavior between AE and VE, it could be started with simple experiments, followed by more complex ones. The models in a DT could be validated initially and then the integration of models could be validated. A modular approach can also be adopted where instead of validating the entire DT at once, critical parts of the system are validated initially, followed by other parts and then the integration of all parts.
- **Continuous validation of DTs:** Three interviewees addressed continuous validation of DTs. They mentioned that models in a DT undergo updates due to data continuously being communicated from the AE in the field, feedback from subject matter experts or field service engineers, or bug fixes. When such updates occur, tests are run continuously to validate the DT and ensure that the same overall behavior is exhibited by DT before and after updating.

[8] https://www.axini.com/en/.

6.6.4 Discussion

All but one interviewee currently perform some form of validation of the complete DT or parts of it. In fact, we identified cases where it is necessary to validate the DT continuously as it undergoes changes across its entire lifecycle. From this we can infer that validation of DTs is highly important. Furthermore, we also identified several challenges involved in validating a DT: one major challenge is that most validation techniques can only cover certain aspects of a DT. Thus, our study suggests a multi-faceted approach, combining multiple techniques, is required to validate the different aspects of a DT.

Takeaway message: We identified 13 out of 19 interviewees who are currently validating their DT by comparing the behavior of AE and VE. In addition, we found three interviewees who are currently using formal methods to validate their DTs. Moreover, testing has been used as a technique for validating DTs. Our analysis suggests that the choice of validation method depends on the DT's purpose, domain, and application; and requires a multi-faceted approach, possibly combining multiple of the aforementioned techniques.

6.7 Properties for validation (RQ7)

The goal of this research question is to understand which validation properties are considered important and need to be validated in the context of DTs. During the interviews, ten interviewees explicitly mentioned one or more such properties in relation to their respective DTs. We discuss these properties and associated challenges based on the interview data analysis.

Table 11 Properties for validation of DTs discussed by industrial (#I) and academic (#A) interviewees

Properties for validation of DTs	#I	#A
Fidelity of VE	8	5
Time difference in execution between AE and VE	2	3
Related to DT's purpose, application, or domain	9	6
Related to dynamic consistency	5	2

6.7.1 Properties for validation of DTs

We intended to understand which DT aspects the properties for validation should cover. We believe the properties for validation of DTs could be quantitative or qualitative in nature. Two interviewees provided a high-level generic overview on this: one industrial interviewee mentioned that the properties to be validated in a DT lie on many levels; another interviewee expressed that the properties should enable the observation of critical things which might go wrong in DTs.

Interviewees used the terms ‘properties’ and ‘parameters’ while discussing this topic. We identified two important properties, namely, fidelity of VE and time difference in execution between AE and VE. Furthermore, we identified two main categories of properties, namely, properties related to DT's purpose, application, or domain; and properties related to dynamic consistency. These properties are discussed in detail below and are summarized in Table 11. Moreover, interviewees discussed the different challenges with properties relevant to verification and validation of DTs; we discuss this in Sect. 6.7.2.

- **Fidelity of VE:** From the validation method based on comparing AE and VE behavior—discussed in Sect. 6.6.1—it can be understood that VE fidelity is an important property that interviewees consider. Eight interviewees from industry and five interviewees from academia discussed fidelity of VE as an important property for validation.
- **Time difference in execution between AE and VE:** While discussing comparing AE and VE behavior, interviewees also discussed comparing the time difference in execution between AE and VE. Two interviewees from industry and three interviewees from academia explicitly mentioned that time difference in execution between AE and VE is an additional property for which validation is needed.
- **Related to DT's purpose, application, or domain:** We identified six interviewees from academia and nine from industry who conveyed explicitly or implicitly that the properties to validate depend on DT's purpose, application, or domain. Out of these 15 interviewees, five interviewees from industry and three academics explicitly specified properties which are specific to DT's purpose, application or domain. Some of these were behavioral properties specific to the DT's application such as whether the DT behaves correctly during specific scenarios and whether the DT has sufficient information from different sources for decision making. Several temporal properties related to DT's purpose, application, or domain were also explicitly specified by four interviewees from industry and three from academia. Some of these properties were related to the timing requirements of the domain; specific timeliness properties for certain applications; and real time properties such as software deadlines and activation time.
- **Related to dynamic consistency:** We identified five interviewees from industry and two academic ones who explicitly specified properties which are related to dynamic consistency of DTs. Some of the functional properties to be validated in DTs which were discussed are deadlocks and bottlenecks. Several temporal properties were also discussed by three interviewees, related to dynamic consistency issues, such as latency in communication between modeling tools; round-trip time and properties on how swiftly a tool sends and receives messages, and response times.

6.7.2 Challenges with properties for validation of DTs

During the interviews, the challenges with validating properties in DTs were discussed. A challenge mentioned by two interviewees from academia was how to measure the quality of DTs and which properties could be used for this. They further mentioned the challenge of quantifying the properties which could be used as a measure of a DT's quality. This challenge also entailed how these properties could be defined in order for them to be computable. We intended to identify such properties, which need to be validated in a DT.

6.7.3 Discussion

From our findings, we noticed that some interviewees expressed their concerns on how to measure the quality of a DT and how to quantify the relevant properties. We also observed that works such as Dalibor et al. [15] discuss this concern on quality assurance and requirements for DTs. Our interviewees, from a range of domains, uniformly agreed that the properties for DT validation depend on the DT's domain, purpose, and application.

Table 12 Overview of additional findings discussed by industrial (#I) and academic (#A) interviewees

Topic	#I	#A
Architecture	7	4
Process	3	4
Role of goals in design	6	5

Takeaway message: Fifteen interviewees discussed that DT properties to be validated depend on a DT's domain, purpose, and application. In addition, we identified thirteen interviewees agreeing to fidelity of VE; and five interviewees agreeing to time difference in execution between AE and VE as important properties for validation of DTs. Specific functional and temporal properties are of interest to seven interviewees as being key to address dynamic consistency issues in a DT.

7 Additional findings

As mentioned in Sect. 4, this research was conducted using semi-structured interviews, allowing the interviewee to discuss any topic. In this section, we present results that do not directly fit any of our research questions. Technical topics mentioned by at least seven interviewees are discussed below, and summarized in Table 12.

Furthermore, given the nature of Digital Twins, the role of humans in DTs is discussed briefly in Sect. 7.4. Finally, Sect. 7.5 briefly discusses future visions for DTs based on what the interviewees expressed.

7.1 Architecture

This section aims to explain the DT's architectural choices as shared by 11 interviewees. The main objective of the architecture is to aid rapid DT development, for which two key architectural properties were identified: re-usability of components and maintainability of the DT. We observed four architectures mentioned by the interviewees, but we only report on the one with at least three mentions, which is the block-based architecture mentioned by four industrial and two academic interviewees. For each interviewee the entity encapsulated in such a block is different. For two academic interviewees, the block is a model which must be configurable, that can exist at different levels of abstraction, e.g., a component of a machine, the complete machine, or the entire manufacturing system. Another two industrial interviewees explained that in order to facilitate VE maintainability, their notion of a block is a component of a machine or process, but never the entire system. According to our analysis, block-based architectures with their separation of concerns between entities of a different nature-such as components of the AE or machines cluster-aid rapid DT development, due to component re-usability and maintainability.

7.2 Process

Seven interviewees discussed the process to build a DT. Five of them follow a software development process adjusted to DT development, although they did not specify how exactly this adjustment was made. Another two described specific, domain-dependent, processes to design a DT. The software development processes that were mentioned are DevOps (Software Development and Operations) or Agile, which facilitate cross-domain cooperation. The other two interviewees used specific design processes and software development practices in different stages of their DT development. While the academic interviewee shared that he uses practices from the V-model to perform unit testing, the industrial interviewee discussed that he uses Continuous Integration-Continuous Deployment (CI/CD) tools.

7.3 Role of goals in design

This section discusses the role that a clear goal can play in the design of digital twins, according to 11 interviewees. All of them mentioned determining a goal (or purpose, or service) as the first step to developing a DT. We identified the influence of the goal in three entities, namely, the model, data, and other design choices such as tool selection and resource definition. Related to models, they mentioned that the goal defines the model's fidelity, level of abstraction, type (e.g., continuous time), and modeling approach (e.g., data or physics-based). Concerning data, the goal defines the data to collect, data processing methods, and the selection of sensors and actuators.

7.4 Role of humans in DTs

We identified two interviewees who explicitly mentioned that humans play an important role in a DT and should also be seen as a part of a DT. Several interviewees also implicitly discussed the importance of humans in a DT during its operation and this has been visualized in Fig. 7, with the numbers used to guide the reader in the text below. One common observation from our analysis is the role of humans in ① for monitoring, training, and other purposes. Three interviewees from academia also mentioned that continuous information input and knowledge use from humans as shown in ② help in updating and improving the models continuously while the DT is in operation. One industrial interviewee also mentioned that ③ which involves changing and tuning parameters in the model based on the data from the real world, may require

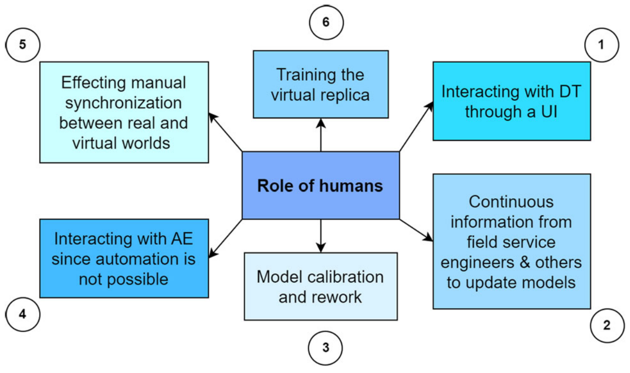
Role of humans
1
Interacting with DT through a UI
2
Continuous information from field service engineers & others to update models
3
Model calibration and rework
4
Interacting with AE since automation is not possible
5
Effecting manual synchronization between real and virtual worlds
6
Training the virtual replica

Fig. 7 Role of humans in the operation of a DT

human intervention. Our analysis further suggests that certain services provided by the DT cannot be automated and thus, a human is required to translate the information received through the user interface, into actions on the AE (see ④) and this was explicitly mentioned by two interviewees. Three mentioned that in some of the DTs they have worked on, there is no direct connection between the AE and its virtual counterpart and thus, humans act as the bridge connecting these two entities by manually transferring data from the AE to its corresponding VE for ⑤. Apart from these, one interviewee from academia mentioned that in one of the DTs he had worked with, humans played an important role in training ⑥ the AE for its specific purpose. Thus, it has been observed that humans, apart from contributing to the development of DTs, play an important role in the operation of DTs.

7.5 Future vision on DTs

One interview question was aimed to understand interviewees' perspective on the future evolution of DTs. During our analysis, we identified interesting and significant outcomes from this discussion. Two interviewees from industry expressed that the DTs could evolve in the future to become a tradeable asset which would be made available alongside its corresponding AE: when an AE is traded between two parties, its corresponding DT could also be traded or provided access to. Moreover, two interviewees shared that in the future, DTs from different domains around us could probably then interact with each other and exchange useful information for use-cases such as improving decision making, accurate diagnosis, etc. According to one interviewee from academia, machine learning and reinforcement learning could possibly be combined with DTs in the future, to help to learn about complex systems (i.e., safety critical systems) in a virtual environment, when this is difficult to do on the real-world system. In addition, two interviewees emphasized that in the future, AEs could possibly become more autonomous and self-adaptive with the help of their DTs. Furthermore, two interviewees predicted DT usage to enable increased automation of real-world processes and improving design support. Additionally, we identified two interviewees who predicted that there would be a shift in the way AEs are developed in the future, where a DT would always be completely part of the development of systems.

We also found interviewees discussing various visions on future improved DT development. Two industrial interviewees mentioned that they believe that standards would become available for, e.g., developing and maintaining DTs, and for managing and combining data in DTs. In addition, two interviewees predicted improvements in the ease-of-use and intuitiveness of DT development tools which would enable experts from different domains with minimal software knowledge and expertise to develop DTs with ease. Moreover, two interviewees explicitly mentioned that DTs should evolve over their entire lifecycle to serve new purposes, i.e., provide new services in addition to what they were initially developed for. Furthermore, one interviewee from industry mentioned that visualization in DTs would possibly improve in the future which would pave a way for better interpretations.

7.6 Discussion on additional findings

During the interview, the interviewees shared their thoughts and opinions on the development of DTs. The additional findings, as visualized in Table 12, are related to architecture, development process and the role of goals in DT design. In addition, the role of humans in DTs and the future vision on DTs are discussed.

Most interviewees who discussed DT architecture mentioned a block-based architecture as their preference. Each has a different interpretation of what a block entails, but all agree that the aim of this architecture is rapid DT development and maintenance.

From the interviewees who discussed their process for DT development, the majority stated using an adapted software development process, such as Dev-Ops or Agile. Others who use specific development processes seem to use some software development practices and tools, such as unit testing or CI/CD tools. Our analysis shows that the main driver to use software practices or processes is the software-centered nature of DT's.

The eleven interviewees who discussed the role of the goal in the design, agreed that it is crucial to define it before starting DT development. The goal has a big influence on many design decisions related to models, data, and tooling. Examples of this are the fidelity of the models or the data selection from the AE.

Furthermore, while several interviewees implicitly discussed the importance of humans' role in DTs, only two interviewees explicitly considered humans to be a part of DTs.

Future predictions were made on three different aspects of DTs, namely the trade aspect of DTs, the influence of DTs in engineering applications, and the evolution of DT development. We believe that some of the aforementioned future applications of DTs such as increasing AE's autonomy and adaptability, and improving automation and design support, are highly important. We further believe that the influence of DTs in engineering applications could substantially grow in the future.

Takeaway message: Our analysis shows that DT architectures should facilitate maintainability, cross-domain collaboration, and rapid development. It also shows that DT development processes contain ingredients of software development processes or adaptations thereof for DTs. A common recommendation for development is to define the goal of the DT before starting its development. Predictions on the business model of DTs have been made such as DTs and their related data becoming a tradeable asset whose ownership could be transferred or shared across the lifecycle. DTs have been predicted to improve the automation and self-adaptability of systems; and also to help in the design support for such systems. Current challenges in DT development such as lack of intuitiveness and standards are predicted to be resolved in the future.

8 Threats to validity

This research is an empirical study, which is never completely devoid of omissions or pitfalls. We discuss the threats to validity (TTV) as described by Easterbrook et al. [42] and the measures taken to mitigate these.

Construct validity pertains to quality of the measurement of the constructs for the experiment. One concern regarding this TTV is the level of expressiveness of information shared by the interviewees about the DT practices and technical DT issues at their organization. However, this was mitigated to a certain extent by emphasizing the research's approval by the universities' ethics review board and data stewards, assuring data privacy rights and complete anonymization of the collected data and its processing.

A second TTV concerns the **limited variability of interviewee opinions**. To mitigate, we included interviewees from both industry and academia with varied educational backgrounds, different levels of experience with DTs; working in diverse roles; from different industrial domains; and from companies of different sizes. As discussed in Sect. 4.2, most of the interviewees are from our own network comprising experts from start-ups to high-tech manufacturing companies from the 'Brainport region', which is one of Europe's biggest industrial hubs. However, we made sure to include interviewees from different backgrounds such as manufacturing and chemical processing, high-tech products, building and construction, transport and logistics, information systems and healthcare. In addition, we also included interviewees working on diverse applications of DTs, namely control, design and development, predictive maintenance, monitoring, analysis, testing, training and demonstration.

A third TTV concerns the **understandability of the interview questions**. To mitigate this threat, we conducted a pilot interview with an interviewee from industry working with DTs. Based on the feedback received, interview questions were reordered and rephrased for a seamless interview flow and ease of understanding, the process of which is explained in Sect. 4.1.2.

The fourth TTV is related to the **background of the interviewers** which may affect the outcome of interviews. This was mitigated by not having one interviewer but three. Moreover, these interviewers had diverse backgrounds in different domains such as software engineering, embedded systems, chemical, automotive, mathematics, electronics, and instrumentation engineering. In addition to diverse backgrounds, the interviewers had experience working in both industry and academia.

A last construct TTV is related to **interviewer bias**. This was mitigated by creating an interview guide which had instructions to drive the interview with an exhaustive list of pre-defined questions to be asked. Furthermore, each interview was conducted by two interviewers, one asking questions and one keeping track of questions and making notes.

Regarding **internal validity**, the dependency of results on just the interview data is a potential threat. In order to mitigate this, the data analysis was done meticulously following the analysis methodology described in Sect. 4.5. Data analysis involved three levels of transcription: firstly, automated transcription; secondly, manual transcription by student assistants; and thirdly, verification of the entire transcripts against the recorded interviews by three researchers. The coding process was scrupulously done by three researchers in an interpretative manner over two months. For any artifact, the process was deemed complete only when it was coded equally by at least two researchers. When the two researchers had different interpretations in coding a particular artifact, these were presented and discussed among the researchers to arrive at a common understanding on such codes. This way of working is described in Sect. 4.5.2.

Regarding **external validity**, the generalizability of results is a TTV. We tried to minimize this TTV by having a significant number of interviewees with diverse roles and varied educational backgrounds from both industry and academia; with different levels of experience with DTs; from different

Table 13 Summary of related work research related to the implementation of DTs

References	Summary	Approach
Perno et al. [43]	This paper conducts a study to identify categories of barriers hindering the development of DTs in the process industry. The paper proposes a model designed to recognize potential facilitators addressing the identified barriers and intends to offer possible solutions	SLR
Segovia et al. [44]	This paper analyzes the state-of-practice on DT development, with a specific focus on the methodological approaches found in scientific literature. The paper considers three primary phases: design, modeling, and implementation. It states the challenges that are encountered in each phase	SLR
Sharma et al. [13]	This study focuses on the conceptualization and adoption of DT in practical applications. The goal of this research is to define the key components of DTs and identify open research questions for their development. This study analyzes three case studies; two from the construction domain and one from the medical domain	SLR
Liu et al. [45]	This paper performs an analysis of the approaches to develop DT based on a classification. Specifically, its implementation classification criteria are based on a central component. For instance, if the primary component is a service, the DT is categorized as a "DT as a service	SLR
Trauer et al. [46]	The objective of this paper is to investigate the obstacles faced by 61 practitioners in implementing DTs. It particularly aims to understand the characteristics of DTs and the technical and non-technical challenges associated with their implementation	Interview
Mihai et al. [47]	This research aims to understand the technologies that ease the implementation of DT and its challenges. It identifies enabling technologies and outlines research challenges for DT implementation, including fidelity and synchronization of models and ongoing standardization efforts	SLR
Botin-Sanabria et al. [48]	This paper examines different barriers to the use and development of DTs, revealing high-level obstacles spanning social, technical, and business domains	SLR
Parnianifard et al. [49]	This research discusses limitations and future challenges, highlighting the complexity and interoperability issues arising from the heterogeneous nature of DT components	SLR
Ammar et al. [50]	This study aims to understand the usage and challenges of DTs by interviewing eight practitioners in the construction industry. Six categories of challenges are identified, with a focus on technology development, particularly in sensor technology	Interview

SLR Systematic Literature Review

industrial domains varying from health care to space exploration; from companies of different sizes; and with different levels of understanding of DTs, considering the lack of common understanding of DTs in both industry and academia. However, the sample size of nineteen in this research can still be seen as relatively small, possibly limiting external validity.

9 Related work

In this section, we explore the academic literature relevant to our study, focusing on current practices and challenges in the implementation of DTs. Table 13 contains a summary of the relevant papers we have explored and compared to our current research.

We have identified two main research topics from the papers presented in Table 13. The first domain is associated with methodologies employed in the implementation DTs [13, 44, 45, 50]. Research related with methodologies for implementing DTs has yielded valuable outputs by identifying potential open research questions and approaches embraced by practitioners. However, it is noteworthy that the focus of these investigations predominantly revolves around the primary approaches by practitioners. This differs from our research where we focus our effort on specific development areas, such as cross-domain model consistency, orchestration, and dynamic consistency.

The second topic revolves around the identification of challenges and enablers in the implementation of DTs [43, 46–49]. Each of these works has highlighted diverse technical and non-technical challenges confronted by practitioners in the course of DT implementation. Nevertheless, these studies avoid examining deeply the strategies employed by practitioners in addressing these challenges. Our research focused particularly on exploring the strategies used by practitioners to deal with different implementation challenges.

10 Conclusion

In this exploratory research, we studied the current landscape of DTs from a technical point of view, particularly its current state-of-practice on design, development, operation and maintenance. To do that we interviewed individuals from industry and academia. Our research focused on software aspects of DTs, specifically related to the interviewees' understanding of DTs, model consistency, integration, orchestration and validation. In addition, we also discussed our interviewees' opinions on the future impact of DTs.

The empirical findings show a lack of methods and tools designed to address inconsistencies between models. Furthermore, we found that distinct tools and approaches are used for the integration and orchestration of heterogeneous models, with their selection dependent on the specific domain or nature of the DT application. Regarding dynamic consistency, approaches primarily involved the comparative analysis between virtual and physical entities.

Our research indicates several research opportunities: (1) developing methods to deal with consistency and formulating guidelines for reuse components in DTs; (2) developing tools that facilitate integration and orchestration; and (3) determining the required properties and approaches for validating dynamic consistency of DT.

In response to the identified opportunities, we see the following future research directions. Firstly, to address the consistency challenges we are developing a framework and related tooling capable of processing heterogeneous DT models for identifying and (semi-)autonomously resolving any potential consistency issues. Secondly, to tackle the integration and orchestration challenges, we are designing a DSL aimed at easing the orchestration design. This DSL will be tailored to accommodate the concerns identified in this preliminary investigation. Thirdly, to address issues pertaining to dynamic consistency, we are working on automating continuous V&V of DTs. This entails combining V&V techniques such as model checking and model-based testing, with active and passive model learning.

Concluding, the field holds a lot of promise and need for applying and adapting the discussed concepts and results, requiring research on how best to adjust and apply these to Digital Twin development, maintenance, and operations. With such research, and with the vision of a DT-enriched future as expressed by the interviewees, the future definitely looks both bright and twinned.

Acknowledgements This research was funded by NWO (the Dutch national research council) under the NWO AES Perspectief Program, Project Code P18-03 P3.

References

1. Zhang, L., Zhou, L., Horn, B.K.P.: Building a right digital twin with model engineering. J. Manuf. Syst. **59**, 151–164 (2021). https://doi.org/10.1016/j.jmsy.2021.02.009
2. Baker, P., Loh, S., Weil, F.: Model-driven engineering in a large industrial context—motorola case study. In: Briand, L., Williams, C. (eds.) Model Driven Engineering Languages and Systems, pp. 476–491. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11557432_36
3. 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). https://doi.org/10.1016/j.cola.2020.101021
4. Davies, J., Gibbons, J., Welch, J., Crichton, E.: Model-driven engineering of information systems: 10 years and 1000 versions. Science of Computer Programming **89**, 88–104 (2014). https://doi.org/10.1016/j.scico.2013.02.002. Special issue on Success Stories in Model Driven Engineering
5. Rodrigues da Silva, A.: Model-driven engineering: a survey supported by the unified conceptual model. Comput. Lang. Syst. Struct. **43**, 139–155 (2015). https://doi.org/10.1016/j.cl.2015.06.001
6. Grieves, M.: Digital twin: manufacturing excellence through virtual factory replication. White Pap. **1**, 1–7 (2014). https://doi.org/10.13140/RG.2.2.26367.61609
7. van den Brand, M., Cleophas, L., Gunasekaran, R., Haverkort, B., Manrique-Negrin, D.A., Muctadir, H.M.: Models meet data: Challenges to create virtual entities for digital twins. In: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 225–228 (2021). https://doi.org/10.1109/MODELS-C53483.2021.00039
8. Bielefeldt, B., Hochhalter, J., Hartl, D.: Computationally efficient analysis of sma sensory particles embedded in complex aerostructures using a substructure approach. Smart materials, adaptive structures and intelligent systems, vol. 1: Development and Characterization of Multifunctional Materials; Mechanics and Behavior of Active Materials; Modeling, Simulation and Control of Adaptive Systems (2015). https://doi.org/10.1115/SMASIS2015-8975
9. El Saddik, A.: Digital twins: the convergence of multimedia technologies. IEEE Multimedia **25**(2), 87–92 (2018). https://doi.org/10.1109/MMUL.2018.023121167
10. Tao, F., Zhang, M., Liu, Y., Nee, A.Y.C.: Digital twin driven prognostics and health management for complex equipment. CIRP Ann. **67**(1), 169–172 (2018). https://doi.org/10.1016/j.cirp.2018.04.055
11. Liu, M., Fang, S., Dong, H., Xu, C.: Review of digital twin about concepts, technologies, and industrial applications. J. Manuf. Syst. **58**, 346–361 (2021). https://doi.org/10.1016/j.jmsy.2020.06.017. Digital Twin towards Smart Manufacturing and Industry 4.0
12. Tao, F., Zhang, H., Liu, A., Nee, A.Y.C.: Digital twin in industry: state-of-the-art. IEEE Trans. Ind. Inf. **15**(4), 2405–2415 (2019). https://doi.org/10.1109/TII.2018.2873186
13. Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., Calinescu, A.: Digital twins: state of the art theory and practice, challenges, and open research questions. J. Ind. Inf. Integr. **30**, 100383 (2022). https://doi.org/10.1016/j.jii.2022.100383
14. Broo, D.G., Schooling, J.: Digital twins in infrastructure: definitions, current practices, challenges and strategies. Int. J. Constr. Manag. **23**(7), 1254–1263 (2023). https://doi.org/10.1080/15623599.2021.1966980
15. 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). https://doi.org/10.1016/j.jss.2022.111361

16. Wright, L., Davidson, S.: How to tell the difference between a model and a digital twin. Adv. Model. Simul. Eng. Sci. **7**, 13 (2020). https://doi.org/10.1186/s40323-020-00147-4
17. Magaldi, D., Berler, M.: In: Zeigler-Hill, V., Shackelford, T.K. (eds.) Semi-structured Interviews, pp. 4825–4830. Springer, Cham (2020). https://doi.org/10.1007/978-3-319-24612-3_857
18. Strandberg, P.E.: Ethical interviews in software engineering. International Symposium on Empirical Software Engineering and Measurement 2019-September, 1–11 (2019). https://doi.org/10.1109/ESEM.2019.8870192
19. Castillo-Montoya, M.: Preparing for interview research: the interview protocol refinement framework. Qual. Rep. **21**(5), 811–831 (2016). https://doi.org/10.46743/2160-3715/2016.2337
20. Hove, S.E., Anda, B.: Experiences from conducting semi-structured interviews in empirical software engineering research. In: 11th IEEE International Software Metrics Symposium (METRICS'05), pp. 10–23 (2005). https://doi.org/10.1109/METRICS.2005.24
21. Kiger, M.E., Varpio, L.: Thematic analysis of qualitative data: Amee guide no. 131. Med. Teach. **42**, 846–854 (2020). https://doi.org/10.1080/0142159X.2020.1755030
22. Bogachenkova, V., Martins, E.C., Jansen, J., Olteniceanu, A.-M., Henkemans, B., Lavin, C., Nguyen, L., Bradley, T., Fürst, V., Muctadir, H.M., van den Brand, M., Cleophas, L., Serebrenik, A.: Lama: a thematic labelling web application. J. Open Source Softw. **8**(85), 5135 (2023). https://doi.org/10.21105/joss.05135
23. Muctadir, H.M., Manrique Negrin, D.A., Gunasekaran, R., Cleophas, L., van den Brand, M., Haverkort, B.R.: Replication package for the interview study on current trends in Digital Twin development, maintenance, and operation. Zenodo (2023). https://doi.org/10.5281/zenodo.10187933
24. Walravens, G., Muctadir, H.M., Cleophas, L.: Virtual soccer champions: a case study on artifact reuse in soccer robot digital twin construction. In: ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion) (2022). https://doi.org/10.1145/3550356.3561586
25. Barros-Justo, J.L., Pinciroli, F., Matalonga, S., Martínez-Araujo, N.: What software reuse benefits have been transferred to the industry? A systematic mapping study. Inf. Softw. Technol. **103**, 1–21 (2018). https://doi.org/10.1016/j.infsof.2018.06.003
26. Mohagheghi, P., Conradi, R.: Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empir. Softw. Eng. **12**(5), 471–516 (2007). https://doi.org/10.1007/s10664-007-9040-x
27. Unity Technologies: Unity User Manual (2019.3). https://docs.unity3d.com/Manual/index.html. Accessed 02 January 2023
28. Hisarciklilar, O., Rahmani, K., Thomson, V.: A conflict detection approach for collaborative management of product interfaces. In: Proceedings of the ASME Design Engineering Technical Conference **6**, 555–563 (2011). https://doi.org/10.1115/DETC2010-28464
29. Object Management Group: Unified Modeling Language Specification Version 2.5.1. https://www.omg.org/spec/UML/. Accessed 17 November 2022
30. Hause, M., et al.: The SysML modelling language. https://www.omgsysml.org/The_SysML_Modelling_Language.pdf. Accessed 12 May (2023)
31. Torres, W., van den Brand, M.G.J., Serebrenik, A.: A systematic literature review of cross-domain model consistency checking by model management tools. Softw. Syst. Model. (2020). https://doi.org/10.1007/s10270-020-00834-1
32. Ptolemaeus, C.: System design, modeling, and simulation: using Ptolemy II. https://ptolemy.berkeley.edu/books/Systems/. Accessed 02 January (2023)
33. Negri, E., Fumagalli, L., Cimino, C., Macchi, M.: Fmu-supported simulation for cps digital twin. Procedia Manufacturing 28, 201–206 (2019). https://doi.org/10.1016/j.promfg.2018.12.033. 7th International conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2018)
34. Barros, A., Dumas, M., Oaks, P.: Standards for web service choreography and orchestration: status and perspectives. In: Bussler, C.J., Haller, A. (eds.) Business Process Management Workshops, pp. 61–74. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11678564_7
35. Li, Q., Zhou, J., Peng, Q.-R., Li, C.-Q., Wang, C., Wu, J., Shao, B.-E.: Business processes oriented heterogeneous systems integration platform for networked enterprises. Comput. Ind. **61**(2), 127–144 (2010). https://doi.org/10.1016/j.compind.2009.10.009. Integration and Information in Networked Enterprises
36. IBM Corporation: Rational Rhapsody User Guide. https://public.dhe.ibm.com/software/rationalsdp/documentation/product_doc/Rhapsody/version_7-5/UserGuide.pdf. Accessed 02 January 2023
37. Wilson, J.M.: Gantt charts: A centenary appreciation. Eur. J. Oper. Res. **149**(2), 430–437 (2003). https://doi.org/10.1016/S0377-2217(02)00769-5. Sequencing and Scheduling
38. van Beusekom, R., de Jonge, B., Hoogendijk, P., Nieuwenhuizen, J.: Dezyne: paving the way to practical formal software engineering. Electron. Proc. Theor. Comput. Sci. **338**, 19–30 (2021). https://doi.org/10.4204/eptcs.338.4
39. Behrmann, G., David, A., Larsen, K.G.: In: Bernardo, M., Corradini, F. (eds.) A Tutorial on Uppaal, pp. 200–236. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7
40. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cmabridge (2008)
41. Higham, D.J., Higham, N.J.: MATLAB Guide, 3rd edn., p. 476. Society for Industrial and Applied Mathematics, Philadelphia (2017)
42. Easterbrook, S., Singer, J., Storey, M.-A., Damian, D.: Selecting empirical methods for software engineering research. Guide to advanced empirical software engineering, 285–311 (2008). https://doi.org/10.1007/978-1-84800-044-5
43. Perno, M., Hvam, L., Haug, A.: Implementation of digital twins in the process industry: a systematic literature review of enablers and barriers. Comput. Ind. **134**, 103558 (2022). https://doi.org/10.1016/j.compind.2021.103558
44. Segovia, M., Garcia-Alfaro, J.: Design, modeling and implementation of digital twins. Sensors **22**(14) (2022). https://doi.org/10.3390/s22145396
45. Liu, Y., Ong, S., Nee, A.: State-of-the-art survey on digital twin implementations. Adv. Manuf. **10**(1), 1–23 (2022). https://doi.org/10.1007/s40436-021-00375-w
46. Trauer, J., Mutschler, M., Mörtl, M., Zimmermann, M.: Challenges in Implementing Digital Twins - a Survey. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 2: 42nd Computers and Information in Engineering Conference (CIE), pp. 002–02055 (2022). https://doi.org/10.1115/DETC2022-88786
47. Mihai, S., Yaqoob, M., Hung, D.V., Davis, W., Towakel, P., Raza, M., Karamanoglu, M., Barn, B., Shetve, D., 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. Tutor. **24**(4), 2255–2291 (2022). https://doi.org/10.1109/COMST.2022.3208773
48. Botín-Sanabria, D.M., Mihaita, A.-S., Peimbert-García, R.E., Ramírez-Moreno, M.A., Ramírez-Mendoza, R.A., Lozoya-Santos, J.d.J.: Digital twin technology challenges and applications: a comprehensive review. Remote Sens. **14**(6) (2022). https://doi.org/10.3390/rs14061335
49. Parnianifard, A., Jearavongtakul, S., Sasithong, P., Sinpan, N., Poomrittigul, S., Bajpai, A., Vanichchanunt, P., Wuttisittikulkij, L.: Digital-twins towards cyber-physical systems: a brief survey.

Eng. J. **26**(9), 47–61 (2022). https://doi.org/10.4186/ej.2022.26.9.47
50. Ammar, A., Nassereddine, H., AbdulBaky, N., AbouKansour, A., Tannoury, J., Urban, H., Schranz, C.: Digital twins in the construction industry: a perspective of practitioners and building authority. Front. Built Environ. **8** (2022). https://doi.org/10.3389/fbuil.2022.834671

Hossain Muhammad Muctadir is a PhD candidate at Software Engineering and Technology cluster in the Department of Mathematics and Computer Science of Eindhoven University of Technology (TU/e), the Netherlands. His research is focused toward model-based software engineering, digital twins, and model consistency management. Previously, he received an Engineering Doctorate (EngD) in Software Technology from TU/e, master in Software Systems Engineering from RWTH Aachen in Germany, and bachelor in Information Technology from University of Dhaka in Bangladesh. Mr. Muctadir has also worked as a software engineer at various organizations in Germany and Bangladesh.

David A. Manrique Negrin is currently pursuing a PhD in the Software Engineering and Technology research group at Eindhoven University of Technology. He earned his Bachelor's degree in Chemical Engineering from the esteemed National Autonomous University of Mexico (UNAM). Subsequently, he obtained a Master's degree in Operational Research with a specialization on sustainable energy from the University of Edinburgh, and a further Master's degree in Automotive Technology from Eindhoven University of Technology. He also has a background industrial experience spanning 8 years in Manufacturing and Sustainability across diverse companies in Mexico. David's research pursuits are deeply rooted in his passion for innovation and problem-solving. His current interests center around Domain-Specific Languages (DSLs), Digital Twin design, and System Engineering, leveraging Model-Driven Software Engineering (MDSE) techniques, and Automotive Software Engineering, particularly focusing on autonomous driving and software development using MDSE methodologies.

Raghavendran Gunasekaran is currently pursuing his PhD. research in the topic of dynamic consistency in digital twins, in the Tilburg School of Humanities and Digital Sciences at Tilburg University. He is also a guest researcher in Software Engineering and Technology group at Eindhoven University of Technology. Prior to this, he obtained his Master's degree in Embedded Systems from Eindhoven University of Technology. He completed his bachelors in Electronics & Instrumentation Engineering at Shanmugha Arts, Science, Technology and Research Academy (SASTRA University). Furthermore, he has 3 years of experience working in the software industry. His research interests include model-based system engineering (MBSE), digital twin engineering, verification and validation (V&V) of software systems including digital twins, cyber-physical systems (CPS), automata learning, domain-specific languages (DSLs), and internet-of-things (IoT).

Loek Cleophas is an assistant professor in engineering of software-intensive systems in the Software Engineering Technology cluster at Eindhoven University of Technology (TU/e) and a research fellow at Stellenbosch University, South Africa, with industrial startup experience as well. His research often relates to digital (twin) engineering, including model-driven virtualization of high-tech systems; analysis and management of models and model collections, including dealing with variability; and consistency and orchestration of models in heterogeneous settings such as high-tech systems engineering.

Mark van den Brand is a full professor of Software Engineering and Technology in the Department of Mathematics and Computer Science, and a visiting professor at Royal Holloway, University of London. His current research activities are on model-driven engineering, domain-specific languages, metamodeling, model management, digital twins, and automotive software engineering. His research is industry inspired; he works with most of the high-tech companies in the Eindhoven (the Netherlands) region. He is project leader of a research project on Digital Twins, the focus in this project on the orchestration and management of models involved in the Digital Twins. He is scientific director of the Digital Twin Lab (part of EAISI) of the Eindhoven University of Technology (TU/e). He has been an invited lecturer and keynote speaker at various conferences, workshops, and doctoral schools. He was and is member of PCs on workshops and conferences related to software engineering, language engineering, rewriting, reverse engineering, and software maintenance. He initiated the special issues of Science of Computer Programming devoted to academic software development (Experimental Software and Toolkits), and since 2007 has been guest editor of six of these special issues. He is on the editorial board of the journal Computer Languages (COLA). He is deputy Editor-in-Chief of platinum open-access journal JOT. He is one of the Editors-in-Chief of the open-access Journal of Software Engineering for Autonomous Systems (JSEAS).

Boudewijn R. Haverkort is Dean of School of Humanities and Digital Sciences at Tilburg University in the Netherlands, since 2019. He received an engineering and a Ph.D. degree from the University of Twente, Enschede, the Netherlands, in 1986 and 1991, respectively. Before moving to Tilburg University, Tilburg, the Netherlands, he has been a full professor with the University of Twente since 2003 and was professor with RWTH Aachen University, Aachen, Germany, from 1995 to 2002. From 2009 to 2013, he also was the Scientific Director of the public-private Embedded Systems Institute, Eindhoven, the Netherlands, an applied research institute focusing on high-tech cyber-physical systems design. His field of interest is very wide, encompassing internet technology, cyber-physical systems, smart energy systems, energy management in data centers, computer performance and reliability evaluation, stochastic model checking, AI and data science, and, more recently, also digital twins. He has published over 200 papers about his scientific work in the above fields, as well as two textbooks (on computer system performance and dependability evaluation; Wiley), and chaired many international conferences. He has been a Fellow of IEEE since 2007 and received, together with co-authors, the Jean-Claude Laprie Award for Dependable Computing in 2023, for the 2003-paper “Model-checking algorithms for continuous time Markov chains.”