Current Trends in Digital Twin Development, Maintenance, and Operation

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], and Boudewijn R. Haverkort‖[2]

[1]Software Eng. & Technology clusterDepartment 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

June 22, 2023

Abstract

Digital twins (DT) are often defined as a pairing of a physical entity and a corresponding virtual entity 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. Although growing in popularity in both industry and academia, digital twins and the methodologies for developing and maintaining them differ vastly. To better understand these differences and similarities, we performed a semi-structured interview research study with 19 professionals from industry and academia who are closely associated with different lifecycle stages of the corresponding digital twins. In this paper, we present our analysis and findings from this study, which is based on eight research questions (RQ). We present our findings per research question. In general, we identified an overall lack of uniformity in terms of the understanding of digital twins and used tools, techniques, and methodologies for their development and maintenance. 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.

1 Introduction

Digital Twins (DTs) have captured the interest of industry and academia in recent years, because of their promise 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. We believe that both industry and academia are playing a crucial role in further developing this concept and DT's design, development, operation, and maintenance. Although growing in popularity, the concepts and practices around DTs are significantly different between instances. According to Zhang et al. [30] there is no general consensus on the nature of the real-world en-

*h.m.muctadir@tue.nl
†d.a.manrique.negrin@tue.nl
‡r.gunasekaran@tilburguniversity.edu
§l.g.w.a.cleophas@tue.nl
¶m.g.j.v.d.brand@tue.nl
‖b.r.h.m.haverkort@tilburguniversity.edu

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

In this exploratory research, we aimed to better understand the concepts around DTs, their differences and similarities. To achieve this, we interviewed 19 individuals from industry and academia who are involved in DT development, maintenance, and usage. During these interviews, within a DT system we primarily focused on the development, evolution, and maintenance of the digital counterpart and associated data. To guide this research we initially identified research questions RQ1–7 listed below. We defined the eighth research question (RQ8) during the analysis phase when we identified interesting insights on the future vision of DTs.

RQ1: How are digital twins defined in practice?

RQ2: How does reuse of existing (software) artifacts influence the lifecycle of DTs?

RQ3: How is consistency maintained among the cross-domain models that are developed independently?

RQ4: What technologies and methodologies are used to integrate models in a DT?

RQ5: What practices are used to design and develop the orchestration and data exchange between models in DTs?

RQ6: What techniques and tools are used to validate a DT and its overall dynamic behavior?

RQ7: What properties need to be validated in a DT for its consistent dynamic behavior?

RQ8: How will DTs and their development evolve in the future?

These RQs aims to investigate the practices and understandings of DTs from a software perspective, thus exploring DT specific software challenges discussed in [28]. The motivation for this study is to explore how current interviewees have been using DTs, what tools, frameworks and methodologies they apply to develop, maintain and operate DTs in different domains; as well as interviewees' understanding of DT. Specifically, we are interested in understanding the practices in three main areas known to be challenges in digital twin development and maintenance:

- Consistency of models used in DTs created across engineering domains;
- Orchestration of such cross-domain models in DTs;
- Validation of DT dynamic behavior.

To present our work, we structure this paper in different sections. Section 2 describes the background of this research, Section 3 the research method we followed. In Section 4, we present a short summary of the domains covered by the interviewees and corresponding DT's applications. We present our findings related to the eight RQs in Section 5, based on the analysis of the information we gathered during the interviews. Section 6 explains the threats to validity of our research and how we attempted to minimize them. Finally, Section 7 discusses our observations and concludes the paper with recommendations for future research directions.

2 Background

The concept of Digital Twins (DTs) was introduced by Grieves [11] in 2003; Grieves modelled 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. [12] focus on ultra-realistic multi-physical computational models in their definition of DT, whereas El Saddik [16] defines a DT as a digital replica of a physical entity whether living or non-living.

Tao et al. [17] extended the original DT model by Grieves and proposed a five dimensional (5D) model depicted in Figure 1, 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.

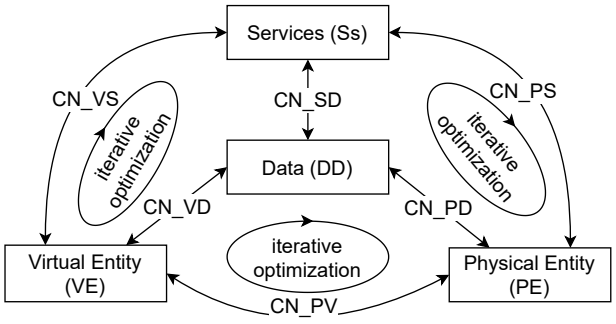


Figure 1: Five dimensional (5D) model of a DT proposed by Tao et al. [17]

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. [29] that analyses the concept of a DT, the technologies used, and DTs' main industrial applications. That research is based on systematic literature review (SLR) analysing over 200 publications.

Tao et al. [21] analyses 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. [34], based on an SLR of over 80 papers, analysed 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. [37] explores 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. [31] conducted an SLR on 356 papers. This paper analyses 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 Research Methodology

Considering the exploratory nature of our research questions we opted for semi-structured interviews [23]. 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. We followed Strandberg's interview lifecycle [20], with our steps depicted in Figure 2. Next, we expand on each step and explain the related activities.

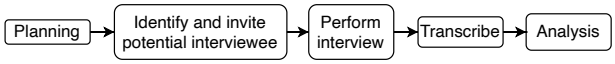


Figure 2: The key steps of the interview research

3.1 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.

3.1.1 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.

3.1.2 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 [13], 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 [3], 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 might understand the questionnaire. Wherever we identified significant difference between interviewee perception and our intention, we rephrased for better understandability.

4. **Conduct pilot interview:** We performed mock interviews with colleagues, allowing us to try our questionnaire, 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 introduced 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 5D DT model [17] (see Section 2). While asking this question, we showed an image of the 5D model and briefly explained it. To avoid influencing the interviewees during the rest of the interview, we intentionally asked this question at the end.

3.2 Finding interviewees

We aimed to interview practitioners and researchers who are actively involved in the development, maintenance and use of DTs. We started with our own network and created an initial 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 since one respondent stopped responding.

Table 1: Interview questions and their relation with the RQs

Research questions (RQs) →	1	2	3	4	5	6	7	8
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?								x
21. What is your opinion on the 5-dimensional DT model from Tao et al. [17]?	x							

We used emails to the potential interviewees to introduce ourselves and explained the purpose of our research. With the emails, we also included a consent form where we explained details of our research about data processing and our measures for ensuring data anonymization and privacy, which allowed us to

create a level of trust with the interviewees. While we did not 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.

3.3 Performing interviews

As we conducted semi-structured interviews, we did not follow any concrete structure and the questionnaire in Table 1 acted as a guideline. However, 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 or confusion 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.

3.4 Transcription

The interviews altogether accounted for just over 26 hours of recorded video with audio. To generate a 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.

3.5 Data analysis

We used the transcripts for further analysis based on a thematic analysis methodology for qualitative analysis [22]. We utilized LaMa [36], 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.

3.5.1 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 splitted artifacts if we found more than one key messages 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.

3.5.2 Labeling of artifact and topic generation

After generating the artifacts for all transcripts, we labeled them using LaMa [36]. 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 ar-

tifacts with two predefined labels: *No value* or *Not understandable*. An example is an artefact discussing an interviewee research policy in relation with their clients. This artefact 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 while creating the topics. Based on our initial overview 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.

3.5.3 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. To answer the research questions, we consulted this matrix to identify related topics. Subsequently, we carefully went through the topics of interest and corresponding labels and artifacts to answer the research questions.

4 Demographics of interview participants

As explained in Section 3.2, our aim was 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 DTs' 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, manufacturing and chemical process industry and high-tech products are the two most dominant categories.

Table 2: Professional domains of the interviewees. The number represents the number of DT's in that domain.

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

As for the DTs' applications, all the interviewees mentioned that they have multiple applications for theirs. We therefore analysed the correlation between the domain and the DT's application. Figure 3 shows this correlation. For the analysis, we classified the applications in eight categories as follows in alphabetical order:

- **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 towards a desired one.
- **Demonstration** of alternative solutions or configurations for a physical system. This tends to use visual tools, such as 3D modelling tools.
- **Design and development** of a product (hardware or software) or process. Examples presented by interviewees are prototyping, use of simulation for design or design 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 3 shows the correlation between the domains of the interviewees and the applications of their corresponding DTs. The *total* value, shown in black, represents the sum of all the applications in a specific domain. The *# application types*, shown in blue, 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.

5 Results and Findings

This section presents the answers to our eight research questions. These answers are based on analyzing the data collected during the interviews. Some of the discussions were not strictly related to the defined RQs but still yielded interesting insights. We present such additional findings in Section 5.9.

5.1 Definitions and understanding of DT (RQ1)

As indicated in Section 1, DT is used as an umbrella term and across different domains can have many different definitions, interpretations, and understanding. With RQ1 we aimed to understand these, and the similarities and differences between them, based on the data collected from the interviews.

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 interview yet over the course of interview mentioned additional aspects of DT, which extend their initial description. In the following text, we present views on DTs observed from the interview data.

5.1.1 Virtual representation of an entity

We intended to find 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'. 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.

One interviewee mentioned that in EU standardization, there is an umbrella term called 'entity of interest' (EOI) to describe entities for which virtual representations are created, encompassing both types of entities mentioned above. EOI as a term matches what we call AE (Actual Entity) in this paper. Eight interviewees implicitly or explicitly dis-

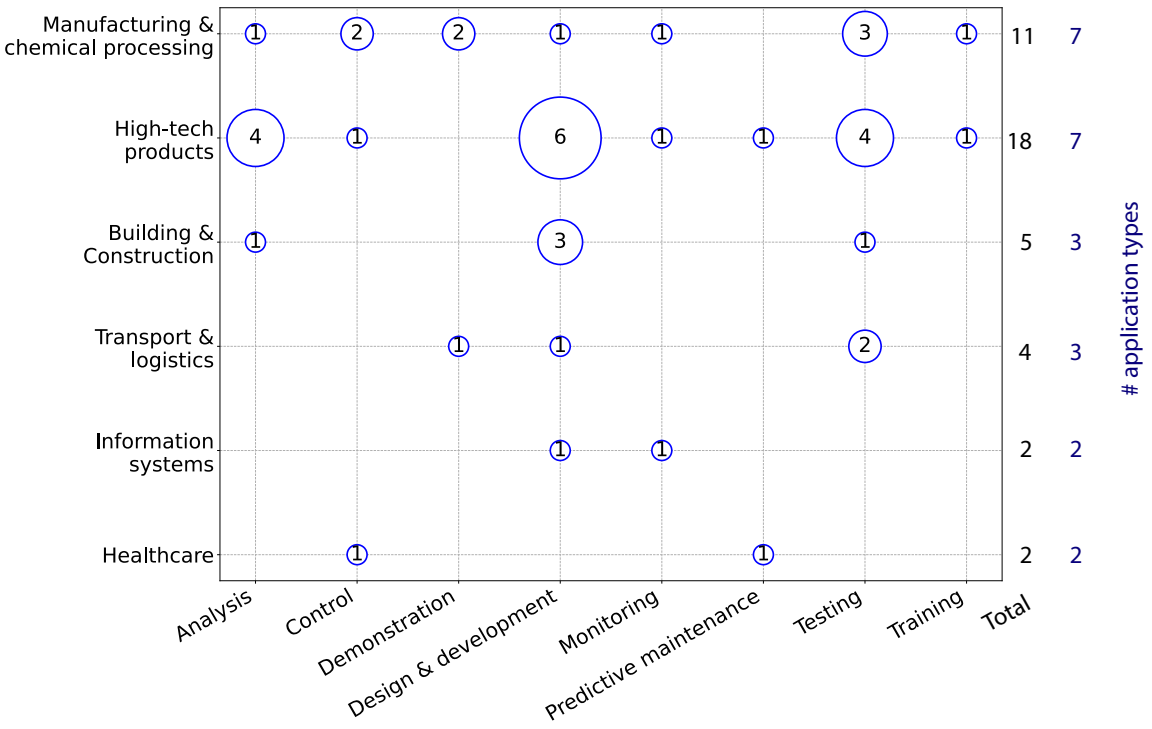


Figure 3: The correlation among the identified domains (y-axis) of the interviewees and DT's application (x-axis). The numbers on the plot represent the number of DT's applications in that specific domain.

cussed a DT being a virtual representation of not necessarily 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.

5.1.2 Components of a DT

Through the interviews, we wanted to understand what components interviewees considered part of DTs and two questions were aimed towards this. Some interesting varying responses were observed such as one academic interviewee describing the components of a discrete event simulation (DES) library as components of their DT. Another academic interviewee discussed the different types of data specific to their DT's application as components of their DT. One industrial interviewee mentioned Computer Aided Design (CAD) models and a software component used for creating model elements as the main components of a DT. However, during the course of

the interview they did implicitly communicate other components or aspects of a DT. The different components of a DT discussed during the interviews are listed below. All these components of DTs as described by interviewees have been represented in Figure 4, where the numbers represent the number of interviewees from academia and industry respectively, who discussed those specific components of DTs.

- **Models:** 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 measured data from sensors; 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 further 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 purpose. This interviewee further discussed that DTs should not be developed with a purpose and they have a purposeless existence, which is in stark contrast to what the thirteen interviewees mentioned above. However, 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 in the 5D model of DTs proposed by Tao et al. [17] (discussed in Section 2).

- **AE:** Already discussed in Section 5.1.1.

- **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 a 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. One interviewee explicitly mentioned that the connections between the AE and its virtual counterpart cannot be considered to be part of the DT since this interviewee's perspective was that it is only an infrastructure needed to create and to operate DTs and thus, it cannot be considered to be a part of DTs.

- **Other components of a DT:** Apart from the components described above, we found other interesting components mentioned by interviewees which could be a part of the DT, which are discussed below. An academic interviewee emphasized algorithms, including AI (Artificial Intelligence) based ones, which could be used for control, decision making, monitoring, or other purposes. Another academic interviewee claimed

that knowledge graphs which capture and describe knowledge about DT components should also be a component of a DT. Two interviewees explicitly mentioned that humans play an important role in a DT and thus, they should also be a component in a DT. One industrial interviewee considers that the main component in a DT at the lowest abstraction level is time. He further elucidated that time is an important component which can be sped up or slowed down, move backward or forward and also, diverge from a specific point in time.

5.1.3 Relation to the 5D DT model

As explained in Section 3.1.2, we collected and analyzed the opinions of our interviewees on the 5D DT model by Tao et al. [17] (explained in Section 2). In this section, we present our observations from this analysis.

We were able to map some of the DT components mentioned by the interviewees, as discussed in Section 5.1.2, to the 5D model. This mapping is shown in Figure 4, where the number of interviewees from industry and academia agreeing to the components is shown, respectively. As shown in the figure, we identified considerable number of mappings between the components mentioned by the interviewees and the physical entity, virtual entity, and data 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.

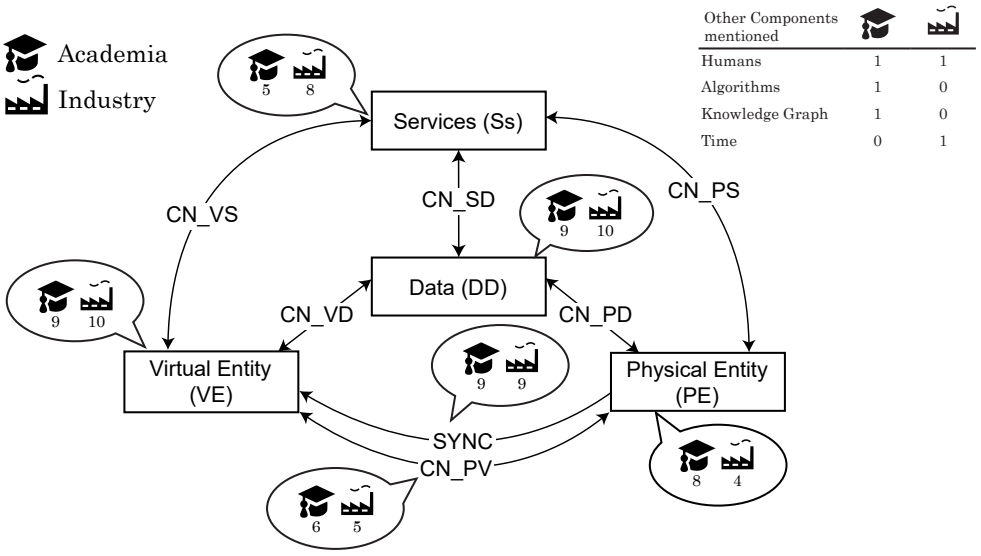
As depicted in Table 3, 11 interviewees mentioned that they could relate to the 5D DT model to a certain extent and agreed to this model albeit with some changes. Four interviewees from industry explicitly disagreed with this model. Whereas another four interviewees neither explicitly agreed nor disagreed 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.

Three interviewees suggested that some connections in the 5D model are not needed and might be removed such as the connection between PE and other components. They further elucidated that the connection from PE might not be needed in some cases such as when PE may not exist or when PE may not be capable to communicate. Two of those three further expressed that some of these connections need not be bi-directional, but can be uni-directional—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 flows in each direction from one component to another. According to two interviewees, humans play an important role in DTs and suggested to include them as another dimension. Two interviewees specifically mentioned the services part of the model to be highly important, and that a service could be interconnected with services from other DTs, thus, enabling service level interaction.

Table 3: Agreement on 5D model of DT [17] by interviewees from academia (#A) and industry (#I). The total number of interviewees are 10 from industry and 9 from academia.

5D DT-model	#A	#I	Total
Agree	6	5	11
Disagree	0	4	4
Neither agree nor disagree	3	1	4

We identified two interviewees who mentioned that they see data and VE combined as a single component. One of these interviewees clarified that they consider VE to represent data. One interviewee from industry emphasized using percentages in this model to represent the relative importance of each of the components. He further elucidated that the importance of each component could vary based on the DT application or based on how the DT is developed: in certain DT applications he had worked on, in his opinion the data was more important than models and this should be represented in the 5D model of that DT. We identified other interviewees who mentioned that data is an important part of the DT, while a few others mentioned that services are the most important part of a DT. One industrial interviewee specifically indicated that he does not see a single



Other Components mentioned		
Humans	1	1
Algorithms	1	0
Knowledge Graph	1	0
Time	0	1

Figure 4: 5D DT model components [17] and interviewees' agreement. The number indicates interviewees' agreement to the corresponding component as part of DTs. SYNC component is defined as the unidirectional synchronization from PE to VE.

VE for a DT, but several VEs which could possibly be inter-connected or separated. Overall, it can be understood that while 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 a generic DT, some changes are also specific to the DT that the interviewee worked on.

5.1.4 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. [28] 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: Our inference from the interview data is that there is no uniformity in the definition of DTs nor in the understanding of the components that make up a DT. Despite this disparity, some agreement existed on certain components of a DT, specifically, models, data, and the synchronization between the AE and its virtual counterpart.

5.2 Influence of reuse on DT lifecycle (RQ2)

As indicated by Walravens et al. [35], developing DTs is a cross-domain and resource intensive task. We believe reuse of existing artifacts can significantly reduce the development and maintenance costs of DTs. Answering RQ2 validated this belief and improved understanding of artifact reuse in the context of DTs.

We identified fifteen out of 19 interviewees ac-

knowledging 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. We further identified knowledge reuse, which we believe is worth discussing despite not being about artifacts. In this section, we discuss the aforementioned reuse scenarios, which we summarize in Table 4, and challenges we identified during our analysis.

5.2.1 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 4 shows the types of reused software artifacts we identified through our analysis. We identified the following factors that encouraged or motivated software artifact reuse:

- **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, which was based on prior publications reporting similar benefits for more traditional software systems [15, 5].

- **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 integrated tooling 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. Four interviewees mentioned *Unity* [42], a well-known game engine, as a tool they use for developing geometry and physics models. *Prespective*[1], a Unity-based 3D design and simulation platform, was mentioned by two industrial interviewees. 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.

- **Transfer of knowledge captured within models:** Two interviewees expressed their concern about personnel changes and loss of gained domain knowledge as a result. In both cases, models or software artifacts are being used as a way to encapsulate, preserve, and transfer such knowledge. This suggests that such use of software artifacts allows distributing this knowledge within the organization and increases the possibility of reuse.

5.2.2 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 . Unlike software artifact reuse, we found the motivation for data reuse to be mostly similar, related to the understanding of a natural or engineered system based on the data gathered from it, and modelling certain aspects of the system in one or more DTs. These DTs are later used for monitoring or enhancing the system. Although the nature, DT's applications, and data sources for such reuse vary significantly, we identified three major types of data reuse:

- **Data related to AE:** According to our analysis, this form of reuse is often related to historical data collected from the corresponding AE. Two interviewees mentioned collecting historical data from across the product lifecycle including design, manufacturing and operation. This data is often used to create a more accurate virtual

[1] `https://prespective-software.com/`

Table 4: Mentions of artifact reuse by interviewees from industry (#I) and academia (#A). We interviewed 10 individuals from industry and 9 from academia.

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

representation of the physical system. Depending on the followed DT concept, this representation can be purely data-based or a combination of software and data where the later is used to enhance the former. We analysed various DT concepts discussed during the interviews in Section 5.1.

- **Data from similar systems:** Four interviewees mentioned developing DTs reusing data gathered from similar systems, namely related *product line* and similar *digital shadows*. 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 mature enough. One interviewee mentioned that they construct a digital shadow first to learn configuration parameters that are later used during DT development.

- **Ontology:** We found that both in industry and academia, 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 high-tech products and building & construction domain, often known to be highly multidisciplinary.

5.2.3 Reuse of knowledge

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 key topics with the potential to significantly influence DT lifecycles. One interviewee estimated around 80% of their work to involve reuse of experiences and knowledge. This estimation provides a good notion of the influence of knowledge reuse. Another interviewee emphasized preserving knowledge, specially considering that people might leave an organization.

5.2.4 Challenges in reusing artifact

Although about 80% of the interviewees mentioned practicing some form of artifact reuse and acknowledged positive effects of it, we often identified cases where reuse was restricted to various degrees; we list the most frequent restrictions identified by our analysis.

- **Legal issues:** We found that legal measures or clauses often restrict or prevent artifacts from

being reused, especially when industrial stakeholder are involved. These measures include non-disclosure agreements (NDAs), intellectual property (IP) rights, 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 issue. Three interviewees mentioned that various data is often collected to be used for specific purposes. Due to lack of metadata, explanation, 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 consequences.

- **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 software engineering skills:** We believe that domain expertise plays a crucial role in the development of DTs, which has been indicated in several publications [27, 35, 18]. As a result, domain experts are often closely involved in the development activities. However, according to one interviewee, developing reusable software components requires advanced software engineering skills that are not common among the involved domain experts. As a result, the developed artifacts may lack reusability.

- **Lack of methodology or tool:** This issue was identified by four interviewees as the reason behind limited reusability of existing artifacts. One of them claimed that while developing software artifacts, future reusability is often disregarded due to the lack of an appropriate development methodology within the organization. Consequently, identifying reusable components and determining the degree of reusability of the identified ones becomes very difficult. 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 reusable in a different context. Furthermore, one interviewee from industry mentioned that organizations often promote the uniqueness of their products, reducing the organizational mindset for reuse of existing artifacts.

- **Additional effort:** Our analysis suggests that reuse of artifacts often takes effort. Four interviewees mentioned that it required additional development and validation efforts. The necessity to adapt existing software components for a new purpose is a major reason for this. Furthermore, previously undetected defects are another reason for this. One interviewee mentioned that a defect within a software component reused from another vendor caused a simulation to break at certain conditions and it took them significant effort to identify and fix the defect.

- **Reuse not possible:** Although generally beneficial, we did identify situations where reuse is impossible or might even have adverse effects. One interviewee shared that not reusing allowed them to be more independent, flexible, and avoid vendor lock-in. Another interviewee emphasized that an NDA signed for a project prevented them from reusing artifacts produced in that project in another one. Furthermore, we identified the

following dominant factors preventing artifact reuse: lack of appropriate tooling; artifacts developed without considering reuse; unexpected unavailability of artifacts; and exceptional system requirements. Although some such factors are similar to the points discussed earlier in this section as reasons for restricted reuseability, we also identified cases where they prevented reuse.

5.2.5 Discussion

Based on the findings presented in this section, we observe that reusing existing artifacts can positively influence the development of DTs. However, we noticed that this is not practiced widely in the organizations that develop and use DTs. We already discussed several reasons and challenges related to this. Although resolving some of these might not be trivial (e.g. legal issues), we believe it is possible to address issues related to inadequate tooling, lack of reuse attitude, and insufficient software engineering skills with moderate organizational effort. Furthermore, during the interviews we noticed that organizations are recognizing the potential of reuse in the context of DTs and gradually moving towards 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 practiced widely due to challenges such as legal restrictions, inadequate tool support, lack of information, and experience.

5.3 Consistent cross-domain models (RQ3)

Inter-domain collaboration is essential for the development and maintenance of DTs. Within a cross-domain environment, we identified the maintenance of consistency among cross-domain software models as a challenge [28]. With RQ3 we aimed to investigate this challenge further. We wanted to understand how consistency is defined in practice, 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.

5.3.1 Types of inconsistencies

Based on our analysis, we divided the identified inconsistencies into the following five categories.

- **Interface inconsistency:** Hisarciklilar et al. [9] defined interface inconsistency as mismatching values, terminologies, or schemes among connected interface elements. We identified this inconsistency mostly in cases where two or more entities, often cross-domain, communicate and do not share any compatible interfaces (i.e., parameters). Our analysis suggests that this is the most frequently encountered inconsistency, with four interviewees explicitly mentioning it.
- **Tool and data format inconsistency:** We identified six interviewees who 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 artifacts. From the interviews, we identified situations where these tools are completely or partially incompatible. As a result, artifacts developed in one tool can not be imported or used in another tool, primarily due to incompatible data formats.
- **Representation inconsistency:** As DTs are often complex cyber-physical systems, various diagrams (i.e., UML [40], SysML [38]) 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. One interviewee provided an example of such inconsistency where the Simulink [32] implementation was done differently compared to the design done using UML sequence diagram.

- **Configuration inconsistency:** From the interviews, we identified two highly dynamic software systems offering many configuration parameters that can be used to greatly change the system behavior. Configuration inconsistency can occur when dependencies exist between configuration parameters and changes are made to them without considering the dependencies. One of the interviewees mentioned that they usually do not change their software, but update configuration parameters to adjust the system, which at times introduced inconsistencies.

- **AE-VE inconsistency:** As analysed in Section 5.1, the concept of a DT often encompasses a certain degree of synchronization between AE and the corresponding VE to facilitate replication of certain behavior or features. We identified two types of inconsistency in this context. The first kind is about inconsistent VE-AE communication, which is often a special kind of interface inconsistency (discussed earlier). The other inconsistency occurs when the virtual and actual entities exhibit differences in certain behavior that is expected to be similar. As a result, identical operations performed on both AE and VE can provide different results rendering the DT ineffective.

5.3.2 Major reasons for inconsistency

Our analysis of the interviews suggests that inconsistencies can emerge for numerous reasons, which largely depends on the nature of the corresponding DT. Therefore, we believe extracting a complete list of reasons is not possible. However, during our analysis, we identified the following four most frequent causes of inconsistencies.

- **Lack of standardisation:** With five interviewees explicitly mentioning it, 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 standardisation 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.

- **Lack of proper collaboration:** DTs, specially the industrial ones, are typically highly multi-domain multi-team projects. Within such environments, various tools can be used for developing artifacts, often with tool-specific semantics. Our analysis suggests that lack of proper collaboration in such diverse environments can lead to inconsistent artifacts. Two interviewees identified lack of communication among different domains or teams working on dependent components as a key reason for inconsistency.

- **Insufficient tooling or methodology:** During our analysis, we identified a general lack of inconsistency detection and mitigation methodology and corresponding tooling. Such methodology or tooling either did not exist or was claimed to be ineffective or inefficient. Although we expected this shortcoming to be more widespread, surprisingly we found only two interviewees explicitly mentioning facing difficulty related to it.

- **Reusing existing artifacts:** We discussed artifact reuse, its benefits and challenges in Section 5.2. Additionally, we found that reuse can also lead to inconsistencies. Our analysis suggests that this is often a consequence of lack of

appropriate testing before reuse or the reused artifact lacking sufficient documentation. One interviewee mentioned that they had reused a software component from a previous project without importing all the related tests. As a result, they were not able to detect inconsistencies that lead to catastrophic failure of the new project.

5.3.3 Inconsistency mitigation practices

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

- **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. One interviewee validated this belief by mentioning the ineffectiveness of a well-defined workflow at their company and how they often needed to resort to informal in-person communication to resolve problems. An interviewee from the healthcare domain mentioned that they consult with their colleagues from automotive and aerospace domain where, according to the interviewee, inconsistency issues are better understood.

- **Use of standards:** We found that the usage of various standards is key for avoiding inconsistencies and maintaining consistencies. Our analysis suggests that these standards can be globally accepted or custom-built for an organization. International standards like Functional Mock-up Interface (FMI) and Functional Mock-up Unit (FMU) were mentioned by two interviewees. One interviewee mentioned using standards defined by the European Union (EU) without mentioning specific ones. Furthermore, usage of custom-built standards was mentioned by two interviewees.

- **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. Two of the four interviewees emphasized the usage of ontologies and related technologies (i.e., SHACL[2]). One interviewee mentioned semantic labeling of data and use of graph databases (i.e., Neo4j[3]) which allows mapping multiple ontologies and automatic data format conversion.

- **Use of in-house tooling:** We discussed software related inconsistencies in the context of multi-tool environments in Section 5.3.1. Our analysis suggests that one way to reduce the number of inter-tool inconsistencies is to avoid multiple tools and using a single one. One interviewee explicitly mentioned developing in-house tooling and using it for their DT development allowing them greater flexibility and avoiding inconsistencies. We found similar strategies used within two other organizations.

- **Testing:** In safety critical domains (e.g., aerospace, healthcare), early and frequent testing or benchmarking is one strategy to identify potential problems including inconsistencies.

- **Use predefined checklist:** Two interviewees explicitly mentioned the presence of a predefined checklist or workflow for identifying inconsistency issues.

5.3.4 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 methodologies and tooling, organizational and personal constraints. Consequently, we believe that it is highly challenging to categorise these inconsistencies

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

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

where this categorisation is also complete. Furthermore, we identified that inconsistency issues are often not categorized as such. Instead, they are treated as regular issues encountered typically during system development or maintenance. Therefore, we believe there is a lack of awareness of the need for specialized methodologies or techniques for identifying and mitigating inconsistencies. We think this might have contributed to the inadequacy of related tooling as discussed earlier. Furthermore, we identified situations where a vast majority of inconsistency issues are being avoided by simply using a single tool for development or 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. referred to several academically developed consistency management approaches in their systematic literature review [24]. However, we could not identify any commonality between this list and the approaches discussed by our interviewees, which we present in Section 5.3.3. Therefore, we believe that there are opportunities for further research and development for inconsistency management tools in the context of DTs. However, we also identified that certain domains, i.e., aerospace, construction engineering, automotive, are more aware and mature about inconsistency issues. This is largely facilitated by 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 various standards as the most frequently practiced measures to avoid such issues. 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 and development is needed to understand these issues and develop tools and techniques to avoid or mitigate them.

5.4 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's 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. An overview of the interviewees approaches and design decisions is shown in Table 5. The second column shows our classification of the findings. The third and fourth column show the number of interviewees who discussed a topic related to a specific class of findings. Not all interviewees discussed all the topics listed in Table 5; hence within each topic classification the number of interviewees does not add up to 19 (total number of interviewees).

5.4.1 Integration approaches

From our analysis, we observed two main integration approaches, namely multi-tool and single-tool. Fourteen interviewees explicitly mentioned the approaches used for model integration in DTs. As shown in Table 5, nine interviewees use a multi-tool approach, while five use the single-tool approach.

Multi-tool approach. This approach is used when a heterogeneous modeling environment is present, in which distinct modeling tools are combined [41]. Each model, in this approach, needs encapsulation which has a defined interface to be able 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 (Intellectual Property) protection and reduction of model complexity. Here, technologies are used so the model becomes a "black box" with only its input and output exposed, but the model details are hidden. However, this approach has many challenges such as data format consistency (discussed in Section 5.3.1), data and model semantics, and relationship complexity between the models.

Single-tool approach. This approach requires to

Table 5: Overview of model integration discussion. #I and #A indicate the number of industrial and academic interviewees who mention that specific topic. We interviewed 10 individuals from industry and 9 from academia.

Topics discussed	Findings classification	#I	#A
Approach	Multi-tool	6	3
	Single-tool	4	1
Communication among models	Influencing design factors	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
Tool type	In-house	7	1
	Commercial	2	4
Challenges	Heterogeneity	2	1
	Complexity	3	1

generate or transform all the models into a single software tool. The single tool will perform the execution of all the models as a whole; hence we refer to it 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, in the single-tool approach. 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 modelling 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, which is a supported platform, requiring extra work from the modeller. During the interviews, interviewees mentioned that this approach might limit cross-domain collaboration, but reduces the integration effort significantly.

5.4.2 Communication among models

As mentioned earlier in this section, an important ingredient of model integration is the communication between models. We aimed to understand how the interviewees implement the communication between models. We identified 15 interviewees who discussed this topic. Our analysis showed two key topics mentioned by interviewees: (1) important factors that influence communication design, and (2) implementation approach.

Influencing communication design factors. We identified three factors influencing model communication, namely DT 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 and requirements of the software modelling tools or platforms involved, to avoid operational failure or additional work.

Implementation approach. We identified two main approaches to implement the communication, namely use of standardised communication protocols and in-

house technology. Five interviewees shared that they use standardised communication protocols. Most mentioned—i.e. by three interviewees—was OPC (Open Platform Communication). Among these five interviewees, three mentioned that they chose a specific communication protocol because the execution platforms support it. The second implementation approach is the used in-house technology for model communication. This approach was mentioned by two interviewees. Both interviewees described their technology as an entity (e.g., communication layer) that centralized the data exchange among the models, these data can be accessed by any model. A model needs to specify which data to extract. We consider this technology to be 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 that discussed this topic.

5.4.3 Technology used

During the interviews, we intended to understand the type of technologies used for model integration. We found that fourteen interviewees described different technologies that are used for this purpose. Table 5 shows a summary of this subsection, where the technologies used varied greatly, but they 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 [19] and is supported by over 170 modelling tools[4]. According to three interviewees from industry, two reasons to use this technology are its maturity and compatibility with various modelling 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, which is discussed in Section 5.9, 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 modelling tool they have used, e.g., if the interviewees have models in MATLAB and ANSYS modelling 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 there is a new modelling tool.
- **Tool provided**: Another four interviewees mentioned that they use what is supported by their execution platform. As with the monolithic approach explained in Section 5.4.1, these interviewees have two choices: either to transform incompatible models or to 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.

5.4.4 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 Section 5.4. We divided the identified execution platforms into two categories: developed in-house and commercial. Table 5 shows a preference for the development of in-house tooling among the interviewees, particularly industrial interviewees. One of the interviewees shared that they used a combination of in-house and commercial tooling, which was therefore also considered in the statistics shown in Table 5. Six interviewees did not mention the tooling used for DT integration.

- **In-house**: The data collected suggests that interviewees from industry prefer to develop their

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

own tools for DT development. The main driver to do so seems to be the DT's application and its domain requirements. Two interviewees mentioned the need for specific execution and modelling 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 modelling tools (commercial and in-house) frequently used in their own domain and thus required to work together. Hence, they decided to create their own tools. To summarize, industrial interviewees seems to prefer to develop their own tools to realize their specific integration requirements.

- **Commercial**: The use of commercial tools varies between modelling (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 mean re-work or limit collaboration.

One interviewee mentioned combining commercial and in-house tooling as helping the collaboration between domains, but having the challenge of tools' semantic uniformity.

5.4.5 Challenges

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

Model Heterogeneity. These challenges are related to the type of models that needs to be integrated into the DT. We identified two challenges related to models: integration of legacy code models and integration of models from different software platforms. Three interviewees mentioned that these challenges seem to be particularly difficult and required 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". Multiple terminology can generate confusion which might delay 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. In addition, they also stated that high fidelity does not necessarily translate to better models. Thus, these interviewees suggest to consider the purpose of the DT as the key factor for design, to define the fidelity of the models. Another factor related to model complexity is model constrains, such as numerical constraints of a model. Two interviewees mentioned that understanding the model limitations is critical because model constraints issues can be confused with integration issues. An example is when a model is constrained to positive numbers as input, and subsequently is integrated with another model which can yield negative output values. If the constraints of the former model are not known by the person who is integrating, then a test might yield an error. Particularly, if the system is tested in out-of-bound conditions of the former model. This error might be confused with a software integration error, rather than a limitation of the system.

The second complexity challenge discussed is related to the system as a whole. Two interviewees stated that a DT can be composed from several components increasing the complexity, hence, the difficulty of understanding the relation between the components. In addition, understanding those relations

becomes critical to solve integration issues.

5.4.6 Discussion

Based on our analysis the multi-tool approach for integration seems to be more 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 to 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, industrial interviewees seem to prefer in-house tooling (seven out of nine industrial interviewees). Our analysis of the challenges suggests that tooling and technologies to facilitate cross-platform integration are required, in addition to consistency management methods discussed in Section 5.3.

Takeaway message: The preferred approach for the integration of models is a multi-tool approach which requires interface development. The preferred technology for such interface 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, the main challenges of integration are three. First its model heterogeneity because models can be of different types of develop in different platforms. Second, its data heterogeneity, which relates to difference in data type and semantics. Third, DT's complexity which is related to two issues. First, the fidelity of the models. Second, the several components that a DT can have.

5.5 Model orchestration (RQ5)

We define DT model orchestration as the definition of the communication actions and execution sequence of the models [4]. To achieve these actions the orchestration should do interfacing evaluation activities such as data compatibility checks and indicate the beginning and end of a model execution [8]. With RQ5 we aimed to understand interviewees' perceptions and practices related to such orchestration in DTs. Based on our analysis, 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 6'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.

5.5.1 Understanding

In this section, we cover the interviewees' discussion on model orchestration understanding. We divided the discussion into two topics: interviewees' explanation of what orchestration is, and 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 DTs, because the complexity of their current DTs is not high. Moreover, one interviewee expressed that the orchestration, if defined in a formal mathematical manner, can be used to reason about a physical system, not only to define the execution of the models.

Components. Five interviewees specifically shared, in their view, the necessary components to design the orchestration of models. All other interviewees men-

Table 6: Overview of model orchestration discussion. #I and #A indicate the number of industrial and academic interviewees discussing that topic. We interviewed 10 individuals from industry and 9 from academia.

Topics discussed	Findings classification	#I	#A
Understanding	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
	Tool provided	0	2
Tools type	In-house	6	0
	Commercial	2	5
Challenges	Model fidelity	2	1
	Model understanding	1	0
	Cross-domain interoperability	3	3

tioned that their orchestration implementation is dependent on purpose and domain, thus they did not define specific components for orchestration. Table 7 shows the components mentioned by the five aforementioned interviewees and the number of mentions for each component.

Table 7: Components to define orchestration in DTs

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

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 made 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. In regards to the data exchange method, interviewees defined it as the scheduling of data exchange between models, e.g., First In, First Out (FIFO).

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 scheduling. The second role is event record in event-based applications, by using time stamps for each event, which is also shown as an important component shown in table 7.

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. Moreover, these interviewees agree that the orchestration design requires a definition on the scheduling and method for data exchange, in addition to defining a global time for the DT application and labeling produced data with time-stamps. Other components for the orchestration design seems to depend on the DT's applications and domain.

5.5.2 Implementation approach

This subsection discusses the implementation approaches shared by the interviewees for model orchestration. 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.

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 that are shown in Table 8. Each approach defines how to implement the models' scheduling. The only approach used by more than one interviewee, 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. There are two examples shown in Table 8. The first example is related to the DT's application of design & application, where the orchestration approaches are two. The first is based on defined rules that activate the models. The second uses concurrent execution of all the models. According to these two interviewees, they chose their orchestration approach because of their knowledge in software and co-simulation, respectively. The second example is on the DT's application of analysis, where the approaches for orchestration are standard workflow and event-trigger. These two interviewees explicitly chose their approach based on their 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.

5.5.3 Technology

This subsection explains the technologies for orchestration that interviewees discussed during the interview. Ten interviewees shared the specific technologies they used. We have classified these into three categories: model-based, own design, and tool provided.

- **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 in 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 component, explained in Subsection 5.5.1. First, the technology that supports time as a trigger. Second, the technology that supports events as triggers.

- **Tool provided**. Two interviewees explained that the orchestration is performed by their execution platform, thus they do not know what technology is used for the orchestration. Their execution platform is a modeling tool like MATLAB, Anylogic[5], and SymPy[6].

[5] `https://www.anylogic.com/`
[6] `https://www.sympy.org/en/index.html`

Table 8: DT's application specific approaches for model orchestration

Approach	# Interviewees	DT's application
Time-scheduling	2	Control
Standard workflow	1	Analysis
Rules	1	Design & development
Event-trigger	1	Analysis
State transition	1	Testing
Concurrent execution	1	Design & development

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.

5.5.4 Tool type

In this subsection, we explain how the interviewees select the tools to develop a DT. 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. One interviewee is classified in both classifications since he uses a combination of in-house and commercial tools.

- **In-house**. Six interviewees explained that they have developed their own execution platform to schedule model execution. Our analysis suggests that the main reason for doing so is to minimize the integration effort to deploy their DTs in existing information systems. For example, one interviewee mentioned that he developed his orchestration tool using C# because their company used C# in their information system. However, from our analysis, we also observed that these interviewees aim to develop specific requirements for their DT that required distinct execution functionalities. An example from one interviewee, is the need to control the global time of execution, which he described as being able to execute a model from the present to the future, or from the present to the past. Based on our findings, we observe two main reasons for in-house tool development: (1) to minimize employment integration effort; and (2) to develop specific execution requirements.

- **Commercial**. We identified three types of commercial tools used as execution platforms. The first is to use tools to orchestrate the models. This type of tool is used by two interviewees who also use the single-tool approach for the integration of models as explained in Section 5.4.1. The second type is the use of system design tools such as IBM Rhapsody [39] and HEEDS[7], which is 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 by interviewees are PDDL (Planning Domain Definition Language) and Dezyne from Verum[8]. The former 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 mentioned 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 orchestrate cross-platform models.

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

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

5.5.5 Challenges

Through the interviews, different challenges were mentioned by interviewees. We have classified them in three groups: model fidelity, model understanding, and interoperability. From 19 interviewees 10 (53%) mentioned such challenges.

Model fidelity. The model fidelity challenge refers to issues related to the performance because of the accuracy of models. We found three interviewees who discussed 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. The interviewees mentioned that solutions to this challenge such as increased computational power are not always available or could increase execution complexity by adding distributed execution.

Model understanding. Model understanding encompasses two types of issues. The first is related to an unclear understanding of the function of the model in the system , which can lead to misuse of the model. An example is having numerical issues mentioned in subsection 5.4.5. The second issue is a lack of understanding of model relationships, e.g., unit consistency and input/output data structure formats, as mentioned in Section 5.3.1.

Interoperability. We identified three main challenges mentioned in this category. The first challenge, discussed by three interviewees, is related to the cross-platform nature of models for DTs, which yields semantic challenges. Another challenge is the design of model scheduling, discussed by two interviewees, particularly when the complexity of the system increases. They mentioned that the scheduling of models should be designed based on the desired purpose; hence, a different purpose requires different orchestration. Finally, another interviewee mentioned that the biggest challenge is related to the interaction of different types of models, e.g., combining continuous and discrete time models due to their different nature and solver strategies.

5.5.6 Discussion

Our analysis suggests that the majority agrees 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 time-stamps as shown in Table 7. During our analysis, we observed various scheduling approaches which are highly influenced by the DT's application and the 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.

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. In addition, the execution trigger is a key component for orchestration. The definition approach for the trigger is highly dependant on its domain and DT application.Moreover, the orchestration design seems to require much domain knowledge and is highly influenced by the DT's application and the developer's knowledge. Technology and tool selection is also highly influenced by their DT's application. Further research is needed on tools to facilitate the interoperability of cross-platform and cross-nature model types.

5.6 Validation and verification techniques and tools (RQ6)

RQ6 was aimed at understanding what specific techniques and tools are used to verify and validate DTs, and their overall dynamic behavior. Here we refer to dynamic behavior as the behavior observed during the collective execution of the models and other components in VE. 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. In the rest of this subsection, we discuss the techniques used for verification and validation of systems, as brought forward in the interviews.

5.6.1 Importance of validation

We believe validation of DTs to be highly important, as it reduces the possibility of errors in their functioning and in their behavior. The importance or need for validation was explicitly and implicitly addressed during the interviews, which are presented below. An industrial interviewee explicitly mentioned the impossibility of designing systems free of errors and thus, the need for validation. The need to validate a DT in order to make it reliable was described by one interviewee as relating to reflecting the AE with sufficient fidelity and to trust in calculation and by necessity consider boundary conditions. He further described that the DT needs to be open to changes and subsequent retraining, recalibration, and revalidation. 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.

5.6.2 Challenges in validation

Some of the general validation challenges of DTs as mentioned during the interviews are discussed 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 possibilities and degrees of freedom, myriad of parameters and corresponding calculations. 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. Measuring the quality of DTs has been voiced as a concern in the interviews, considering the lack of a standard methodology to do so.

5.6.3 Verification and Validation techniques

The different verification and validation techniques and strategies put forward by interviewees have been depicted in Figure 5. 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 or from literature. According to one interviewee, validation of a DT could be done by understanding how well it has served its purpose, e.g., optimization, decision making, or predictive maintenance. Another interviewee shared that validation is part of the process of updating models based on using the continuous data from the AE. It has been 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; we next present our findings on these.

- **Comparing AE & VE behavior** This technique concerns behavior comparison between VE and AE, with the aim to check differences. We identified thirteen interviewees who discussed this informal validation technique. This is done in different ways at different abstraction levels. One way is using observational tests at

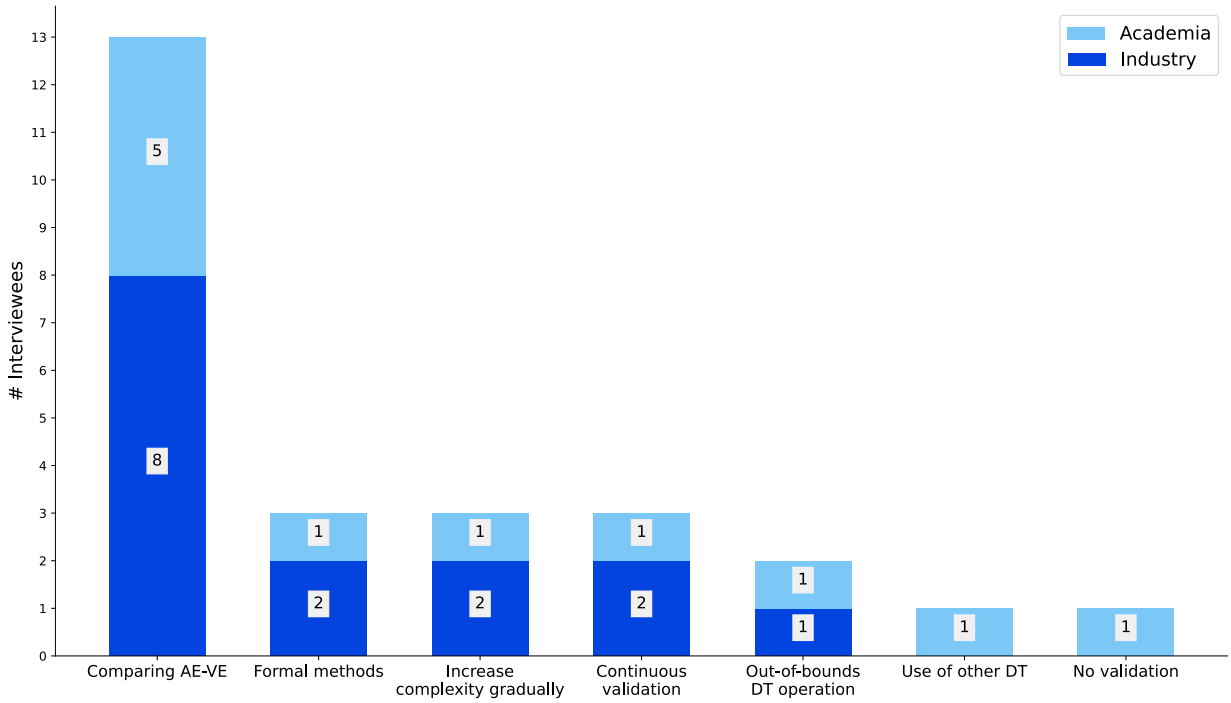


Figure 5: Different verification and validation techniques and strategies used by interviewees

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. 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.

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 [1]. The timing and sequence of actions are compared between AE and VE to check if there are any differences. Furthermore, in these cases, the system dependencies from both AE and VE are also compared, by creating respective dependency graphs and comparing these. Validation is a challenge when there is a combination of continuous and discrete behaviors in AE and VE. In such cases, continuous signals are transformed into discrete ones and then the behaviors of AE and VE are compared to check for equivalence. In some cases, to compare distributions using statistics, the amount of deviation is quantified using Kullback-Leibler divergence tests [10].

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 & 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 [26] and Coco[9]. 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 [2], allowing model checking. However, model checking is not quite scalable, considering state space explosion [6]. He 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 were 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. He further discussed that in such cases, model-based testing is also done, to ensure that relationships between provided and required interfaces are not violated. 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 [6]. Thus, formal methods have disadvantages related to scalability and broad applicability across domains.

- **Testing and corresponding tools:** DTs also often undergo testing across their entire lifecycle, in order to check adherence to requirements of a DT and its components. At times, some interviewees used the term 'testing' to discuss two items, namely, using scripts to test the system and comparing the AE and VE behavior using observational tests. Due to the lack of clarity in this term being used for different approaches, we did not perform quantitative analysis of interviewee responses for testing. As 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[10], Matlab [14], and Unity [42] which are used for creating and testing models on the fly.

  As a related challenge, one interviewee explicitly mentioned that testing requires more effort than development, estimating a factor of three to four difference, in DTs in the space domain. As mentioned earlier, testing is a very resource- and time-intensive one requiring the right facilities and people to be available.

[9] `https://cocotec.io/`

[10] `https://www.axini.com/en/`

5.6.4 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 after model reduction:** Detailed multi-physics models are complex in nature and might not allow real-time execution, hence, behavior computation at global scale is not possible. The complexity of such models is therefore reduced to obtain simplified models which work in real time (using neural networks, for instance). However, the complete detailed methodology for model reduction in this context was not discussed during the interview. This model reduction helps in facilitating the VE's behavior validation.

- **Early validation of assumptions in the case of uncertainties:** We identified two interviewees who discussed using assumptions about the VE's intended behavior during design, related to uncertainties, unforeseeable events, or unpredictable environments. Such assumptions need earliest possible validation, with one interviewee suggesting to have a shorter design loop. The behavioral assumptions could be validated against the AE, if possible. However, in domains such as the space domain, where it is not, and where no relevant stakeholders are available for validation either, the interviewee mentioned that an assessment of risks of unforeseeable events and uncertainties is done, in order to reduce the possibilities of failure to acceptable levels.

- **Validation by increasing complexity:** We found three interviewees who advocated 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.

- **Validation by operating DT at out-of-bound conditions:** We identified one industrial interviewee and one academic interviewee who mentioned validating a DT by operating it at out-of-bound conditions. Such pushing beyond design space helps in determining whether the scientific principles based on which the DT was designed, still hold. This technique is used to test functional issues in DTs and especially, whether the design of a DT encompasses more than just the data from the field, and to check for issues in cases of extrapolation. This technique also helps to find defects and to test whether behavior associated with every DT parameter is well defined, e.g., when operating at out-of-bound conditions whether it gives the right error messages in unrealistic scenarios. This method of testing helps to increase confidence and reliability of DTs. According to one interviewee, this technique has also been used as a test of quality of DTs in the print industry.

- **Continuous validation of DTs:** Three interviewees discussed about continuous validation of DTs. They mentioned that models in a DT undergoes updates due to data continuously communicated from the AE in the field, feedback from subject matter experts or field service engineers, or bug fixes. When such updates occur, benchmarks are run continuously to validate the DT and ensure that the same overall behavior is exhibited by DT before and after updating. One interviewee also advocated validation after reuse of artifacts during DT development.

- **Other cases of DT validation:** Validating a DT by using another DT, by simulating the behavior of the DT before deploying, was also discussed by an interviewee, referring to Ahlgren et al. [25]. In the chemical domain, a DT is validated by comparing it with the AE design information in terms of mass and heat balance equations. One interviewee from academia en-

visioned that over time validation of their DTs would be done by having a human-in-the-loop.

5.6.5 Not validating the DT

Eighteen interviewees discussed validating DTs by using the above mentioned methods, with just one interviewee indicating not validating their DT. He shared that there are currently no tests in place to check if the DT is functioning as intended. Moreover, any issues in the DT behavior can only be observed when the simulation is running. He elucidated that since he is already aware of how the DT should behave and deviation in behavior can be observed during the execution, no validation is performed on his DT.

5.6.6 Discussion

Based on the observations presented above, 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 and it is performed widely across the engineering spectrum in both academia and industry. 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 and not all. 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 also been used as a technique for validating DTs. We also found one interviewee who does not validate their DT currently. 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 aforementioned techniques.

5.7 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.

5.7.1 Introduction and challenges

We intended to understand which DT aspects the properties for validation should cover. 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. During the interviews, the challenges with validation 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.

5.7.2 Properties for validation of DTs

Different types of properties to be validated in a DT were discussed by the interviewees. Interviewees used the terms 'properties' and 'parameters' while discussing this topic. We classify the properties at high level into (1) behavioral properties, and (2) qualitative properties. We discuss these classes below.

Behavioral properties: From the validation method based on comparing AE and VE behavior—discussed in Section 5.6.3—it can be understood that VE fidelity is an important property that interviewees consider. In addition, five interviewees explicitly mentioned that time difference in execution between AE and VE is an additional property for which validation is needed. 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. These properties related to the domain, DT's application or purpose will be discussed below. In addition, properties related to dynamic consistency of DTs will also be discussed below.

- **Temporal properties related to domain, DT's application or purpose:** Several temporal properties were discussed by the interviewees. An interviewee from the space domain emphasized the importance of timing requirements in this domain and thus, of validating these properties in the DT. One property discussed was the availability of sufficient margin in timing budgets to meet software deadlines. He further mentioned properties specific to space missions where multiple computers are used to avoid failures which could lead to loss of onboard human life. These properties are whether the secondary computer switches and takes over control when the primary computer fails within the time margin available; and the retrieval of stored data during the mission. One academic interviewee mentioned query execution time as an important property for connected and autonomous vehicles. He further expressed that in semantic web use for DTs, low latency is an important property. Another academic interviewee shared that the maximum time taken for the DT to perform an action is an important property to consider. One academic interviewee mentioned the property of communication time between nodes, and an industrial one mentioned validating certain parameters in the communication layer, though which properties this pertained to was not made explicit. One interviewee from industry also expressed that timeliness is an important property to be considered in a DT. Real time properties, in the form of activation time and software deadlines, were discussed by two interviewees from industry.

- **Other properties related to domain, DT's application or purpose:** Other properties specific to domains were also discussed by interviewees. In the case of the lighting domain, e.g., a DT comprising several lamps in a room, these can be simple properties such as whether both lamps are only ON or OFF together. One interviewee from academia discussed properties for temperature control in a building such as how much data is required and needs to be stored for decision making, and how much redundant information is present in the stored information.

- **Properties related to dynamic consistency:** Some of the functional properties to be validated in DTs which were discussed are deadlocks and bottlenecks. One interviewee mentioned different types of deadlocks such as various software deadlocks: circular reference deadlocks; deadlocks related to data permutation; behavioral deadlocks resulting from the interplay of different behaviors—for instance, interplay of behavior related to kinematics, geometry, and time; and other types of deadlocks. Several temporal properties were also discussed, related to dynamic consistency issues, such as latency in communication between modelling tools; round-trip time and properties on how swiftly a tool sends and receives messages, and response times.

Qualitative properties: Some qualitative properties were also specified by an academic interviewee for a conglomerate of DTs such as modularity and composability.

5.7.3 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. [31] also discuss this concern on quality assurance and requirements for DTs. An academic interviewee mentioned using acceptance tests based on the Kano model [33] to measure the quality of requirements in a DT. However, the effectiveness of this model and its applicability for DTs across different domains was not discussed. Our interviewees, from a range of domains, uniformly agreed that the properties for DT validation depend on the DT's domain, purpose, or application.

Takeaway message: Fifteen interviewees discussed that DT properties to be validated depend on a DT's domain, purpose, or application. In addition, behavioral and qualitative properties have been discussed in the interviews. Specific functional and temporal properties are of interest to some of these interviewees as being key to address dynamic consistency issues in a DT.

5.8 Future vision of Digital Twins (RQ8)

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. In order to accommodate and coherently present these results, we formulated RQ8. Please note that unlike all other RQs, this RQ was defined after conducting the interviews.

5.8.1 DT as a tradeable asset

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. Furthermore, at times, organizations outsource their projects to a third party for developing the AE. During its development, the third parties might possibly create a DT of that AE for improving the AE's design. In such cases, the organization outsourcing its project would not only expect the third party to develop the AE, but would also want access to or ownership of the corresponding DT. This transferring or sharing of DTs across the entire AE lifecycle has been predicted by these two interviewees to be a trend in the future of DTs. This could be helpful in two ways. Firstly, third party organizations working with this AE–such as for maintenance, operations, or other collaborations—might require use of the corresponding DT, in order to efficiently provide their services and make them effective. These third parties could be provided access to the DT then since the organization owning the AE also owns or has access to the corresponding DT. Secondly, when organizations gain ownership of or access to DTs along with the relevant data, they have the option to experiment with this DT to see how they can make the best use of the newly traded AE for their organization. This experimentation with DTs in a way helps to realize the complete capabilities of an AE and thus to make the best use of this AE. One of the aforementioned two interviewees further specified that the ownership of data in a DT becomes a challenge when providing services based on such data. This interviewee speculates that when a service is provided to one party, using data owned by another party, then the data could be used to generate revenue and thus also become a tradeable asset. However, the other interviewee mentioned that shared ownership of a DT also brings in an additional complexity in terms of reliability of the shared DT and who is responsible when any impactful incident occurs.

5.8.2 Future evolution of DTs in real world applications

Below we present the predictions made by the interviewees on how the role of DTs would evolve in real world applications.

- **A world of DTs interacting with each other:** During our analysis, we identified pre-

dictions pertaining to the interactions among DTs of different vendors. Particularly, one interviewee who discussed transferring ownership of DTs across its entire lifecycle mentioned that when DTs become a tradeable asset for most real world products traded, this could result in several DTs for every organization or person using the corresponding AEs. Two interviewees shared that these 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.

- **DTs increasing AE's autonomy and adaptability:** Two interviewees emphasized that AEs could possibly become more autonomous and self-adaptive with the help of their DTs. One interviewee predicts that DTs will become part of the AE and thus, systems can reason about themselves with the help of their DT: e.g., detect when they are not functioning properly or optimally and request a human to further diagnose the issue. Once diagnosed, they might then optimize themselves to a certain extent, as long as it is within their scope of control, such as changing a few parameters, disabling functions, deploying necessary software and others. This would avoid the need for human intervention for making such changes. Moreover, when the environment around the AE changes or when the operator wants to use that system in a different way, then with the help of its DTs, the system would be able to autonomously adapt itself to these changes, increasing flexibility of usage.

- **DTs in improving automation and design support:** Two interviewees predicted DT usage to enable increased automation of real world processes. They mentioned that changes which need to be done in an AE, could be applied directly on the corresponding DT by giving commands; and that the DT could then automatically make those changes in the AE.

  They further predicted that the DT will play an important role in design support when developing new AEs. One interviewee mentioned that DTs will direct the designers and help in selecting the appropriate components needed for a factory floor when setting it up. Another interviewee expressed that with DTs' application of machine learning techniques, DTs will become more intelligent and could possibly become an interactive design support assistant. This clever assistant will help to solve problems and prevent pitfalls in the design that an engineer makes. For example, when systems are integrated and interact with each other to exchange data, the DT would possibly detect where and what inconsistencies occur in such interactions and help in fixing them in the design.

- **Role of AI in DTs:** While the terms 'Artificial Intelligence', 'Machine Learning' (ML) and 'Reinforcement Learning' (RL) were mentioned during different parts of the interview, six interviewees specifically mentioned these terms while discussing future trends of DTs. According to one interviewee from academia, ML and RL can be combined with DTs 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. Furthermore, he mentioned that ML algorithms could be used to learn control software using the DT, and then control the corresponding AE. Another interviewee from industry suggested that integration of AI and ML with DTs will be the biggest step for the next 10 years and can help to improve predictive maintenance of real-world systems. An academic interviewee mentioned that AI will be useful in the development of DTs when using data obtained from the AE for automated model improvement or refinement.

- **Other future predictions on DTs:** Two interviewees also 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. One of these interviewees identified a related challenge regarding resistance against accepting the results from DTs

in certain communities as they require a confirmation of those results from the real-world system. He believes that this would possibly change in the future and most communities would start accepting results from DTs.

5.8.3 Future evolution in the development of DTs

Below we discuss the various visions on future, improved DT development.

- Two industrial interviewees mentioned lack of standards as a challenge. They believe that such standards would become available for, e.g., developing and maintaining DTs, and for managing and combining data in DTs.
- One interviewee mentioned the interoperability challenge related to integrating different simulation software tools (discussed in Section 5.5.5) would be solved in the future. Moreover, he mentioned that a platform supporting tool interoperability would be available for DT developers to use in the future. We observe that this seems dependent on the definition and proper implementation of standards to ensure such interoperability.
- Two interviewees mentioned improvements in the ease-of-use and intuitiveness of DT development tools. They mentioned the current challenge in DT development as requiring software knowledge in order to develop them; and they believe this would change in the future. They expect DT development tools to become more intuitive such that DTs can be developed by defining simple functions, adding minimal code, and dragging and dropping components, without requiring assistance of software experts. This would enable experts from different domains with minimal software knowledge and expertise to develop DTs with ease.
- One interviewee from industry also mentioned that visualization in DTs would possibly improve in the future in order to present data to humans in a convenient way leading to better interpretations by humans.
- An academic interviewee compared DTs to software and mentioned that DTs will hence have to be versioned, tested, validated, and certified in the future, like regular software.
- Another interviewee from industry used the term 'Simulation as a Service' to describe a cloud-based service similar to a DT, which would possibly be available in the future for performing Finite Element Method (FEM) analysis by simply uploading CAD models and selecting the part where the analysis needs to be done, and providing the results.
- Two interviewees also 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. One of them further mentioned that the speed of this evolution depends on the DT's application and context.

5.8.4 Discussion

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 in the future. Some of the future predictions made by the interviewees have not been discussed in any literature to our knowledge. 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. It was interesting to observe several interviewees discuss future improvements in DT development. However, there was no explicit timeline specified by the interviewees on when they expect these improvements to come into existence. With DT development, we believe that improving the ease-of-use and intuitiveness

of DT development tools which would enable non-software experts from any domain to develop DTs with ease, would be a game changer and greatly improve the adoption and use of DTs across domains.

Takeaway message: 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. AI is expected to become highly pronounced in DTs in the future. 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, standards, and interoperability are predicted to be resolved in the future.

5.9 Additional findings

As mentioned in Section 3, this research was conducted using semi-structured interviews, allowing the interviewee to discuss any topic. We dedicate this section to presented results from topics discussed which do not fit any of our research questions. Topics mentioned by at least seven interviewees are discussed below, and summarized in Table 9. The first column shows our classification of the findings. The second column shows the number of interviewees who discussed a specific class in our classification.

Table 9: Overview of additional findings discussion. Columns #I and #A indicate the number of industrial and academic interviewees who mention that specific topic. We interviewed 10 individuals from industry and 9 from academia.

Topic	#I	#A
Architecture	7	4
Process	3	4
Goal's role in design	6	5
Modelling practices	2	5

5.9.1 Architecture

This section aims to explain the DT's architectural choices shared by 11 interviewees. We identified two key architectural properties mentioned by our interviewees, which are re-usability of the components and maintainability of the system. According to the interviewees, the main objective of the architecture is to aid rapid DT development. We observed four architectures mentioned by the interviewees, but we only report on the one with at least three mentions.

A block-based architecture for the VE was mentioned by six interviewees, four industrial and two academic. For each interviewee the entity encapsulated in such a block is different. For two academic interviewees, the block is a model that can exist at different levels of abstraction, e.g., a component of a machine, the complete machine, or the entire manufacturing system. For them each block must be configurable, to define limitations on the models and make them unique. Another two industrial interviewees explained that their notion of a block is a component of a machine or process, but never the entire system, e.g., in a belt conveyor DT, the blocks can be the belt, the motor, etc. For them, the separation of components should facilitate VE maintainability. Another industrial interviewee described each building block as a stage of the life cycle of the product, which can be assembled to compose the DT; e.g., a block is the design of a wind turbine, and another block is the wind turbine operation. Finally, another industrial interviewee explained that each engineering domain builds a block constituted of several models which later are encapsulated and connected to other blocks.

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 engineering domains—aid rapid DT development, due to component re-usability and maintainability.

5.9.2 Process

This section aims to understand the process seven interviewees shared to build a DT. Five of them follow

a software development process adjusted to DT development. Another two described specific, domain-dependent, processes to design a DT.

The five interviewees who follow software development processes expressed that they have adapted the processes but they did not specify how. The software development processes that were mentioned are Dev-Ops (Software Development and Operation) or Agile. They stated that these practices facilitate cross-domain cooperation.

The other two interviewees, who used specific design processes, recognized that each domain develops DTs in different ways, but still uses software development practices in different stages of their DT development. The industrial interviewee discussed that he uses Dev-Ops practices and tools to develop some of his sub-steps, such as the use of Continuous Integration-Continuous Deployment (CI/CD) tools. The academic interviewee shared that he uses practices from the V-model to perform his unit testing.

5.9.3 Goal's role 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.

Related to models, they mentioned that the goal defines the model's fidelity, level of abstraction, type (e.g., continuous time), and modelling 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. They mentioned that to design the collection of data, the data must have a purpose. The data purpose aids the selection of data sources, processing methodologies, and other properties such as collection frequency.

Other design choices influenced by the goal are tool selection, resource definition, and optimization methods. Moreover, one interviewee mentioned that the DT's goal should relate DT development to the business objectives.

5.9.4 Modeling practices

We found six interviewees who discussed the best practices to maintain or create models for DTs. Regarding maintainability, three interviewees shared that in DTs the models must be updated regularly since they should be synchronized with the AE during their life cycle. However, they mentioned that this is a challenge since current version control systems are not appropriate for complex systems such as DTs.

Another three interviewees mentioned that it is important to find a balance between model fidelity and system complexity. An example is the use of data-based models (e.g., machine learning models) in highly complex systems, which require real-time responsiveness. According to these interviewees, data-based models can execute faster, have good fidelity, and can learn from the environment, compared to physics-based models.

5.9.5 Role of humans in DT

As mentioned in section 5.1.2, two interviewees explicitly mentioned that humans play an important role in a DT and thus, they 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. One common observation from our analysis is that humans interact with a DT through a user interface for monitoring, training, and other purposes. 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. Two interviewees explicitly mentioned that complete automation in a DT is not possible and so, the human part in a DT is very important for its operation. Three interviewees from academia also mentioned that continuous information input and knowledge use from humans—such as field service engineers, subject matter experts, stakeholders part of or using the AE and others—help in updating and improving the models continuously while the DT is in operation. One industrial interviewee also mentioned that model calibration and rework which in-

volves changing and tuning parameters in the model based on the data from the real world, may require human intervention at times. 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 in order to effect their continuous synchronization. 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 the virtual counterpart of an AE, which then trained 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; this has been visualised in Figure 6.

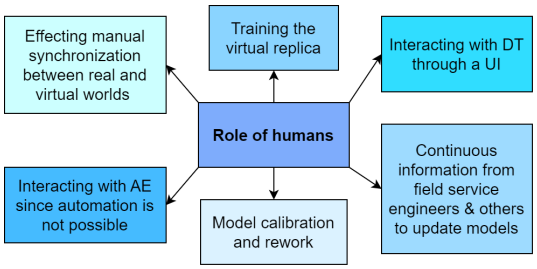


Figure 6: Role of humans in the operation of a DT

5.9.6 Discussion on additional findings

During the interview, the interviewees shared their thoughts and opinions on the development of DTs. The additional findings, as visualised in Table 9, are related to architecture, development process, the DT goal's role in design, and modelling practices.

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 DevOps 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.

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.

Related to models, interviewees discussed the importance of DT models' evolution during the life cycle of the AE. This evolution requires proper tools for version control, going beyond currently available ones.

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.

Takeaway message:
Our analysis shows that the DT architecture should facilitate maintainability, cross-domain collaboration, and rapid development. Also, our analysis 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. Concerning modelling practices, our analysis shows that models should evolve with the AE and that current technologies for version management are not sufficient for these complex systems.

6 Threats to Validity

This research is an empirical study, which is never completely devoid of omissions or pitfalls. We discuss the threats to validity as described by Easterbrook et al. [7] and the measures taken to mitigate these.

Regarding **construct validity** which pertains to quality of the measurement of the constructs for the experiment, one concern 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 threat 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.

A third threat 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 paraphrased for a seamless interview flow and ease of understanding, the process of which is explained in Section 3.1.2. A last construct validity threat 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 predefined questions to be asked. Furthermore, interviews were 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 Section 3.5. 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 was described in Section 3.5.2.

Regarding **external validity**, the generalizability of results is a threat. We tried to minimize this threat 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 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.

7 Discussion and 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. This research is focused on software aspects related to DTs, specifically the interviewees' DT's understanding, model consistency, integration, orchestration and validation. These challenges were discussed in [28]. In addition, we also discussed our interviewees' opinion on the future impact of DTs.

With regard to understanding of DTs as discussed in Section 5.1, our findings suggest that there is no consistency in the definition of DTs nor in the understanding of the components that make up a DT. However, commonality exists in the understanding that a DT is a virtual representation of an entity. Moreover, there is some agreement on certain components of a DT, namely, model, data, and some level of synchronization between the virtual representation and its AE. The level of agreement of the different components in a DT was depicted in Figure 4. In addition, we asked the interviewees for their opinion on Tao et al.'s DT model [17], explained in Section 2. 11 interviewees agreed to this model to a certain extent, although with some changes to this model. Some of the changes suggested pertaining to the connec-

tions in this model are eliminating certain connections among components and some of the connections can be made uni-directional instead of bi-directional. Furthermore, other changes which were suggested to this model are that there could more than one VE which could be separated or inter-connected, data and VE component in this model could be combined into a single component and humans play an important role in DTs, thus they could possibly be included as another dimension.

Regarding reuse practices in DTs (Section 5.2), we noticed positive effects in the development of DTs, yet in practice infrequent reuse due to its challenges. On the other hand, we observed a trend towards multi-tool development approaches to DTs which can benefit from re-use practices. Moreover, the multi-tool approach was also frequently used to integrate models, as shown in Section 5.4.

The findings on model consistency discussed in Section 5.3 show that inconsistency issues are recurrent, but not recognized as inconsistency issues specifically. We observed that these issues are solved as regular issues, with no specialized methods or tools implemented. We are convinced that further research on methods and tools to tackle inconsistencies is needed. Furthermore, such methods and tools can tackle orchestration challenges related to data exchange, as discussed in Section 5.5. These methods and tools can be based on highly standardized domains, e.g., the automotive and aerospace domains,which has demonstrated to be more mature in dealing with inconsistencies.

Our findings on integration (Section 5.4) show maintainability, cross-domain cooperation and effort as the main criteria to select the approach to integrate. The high frequency use of a multi-tool approach is due to facilitation of the first two criteria, although it requires more effort: heterogeneous components require encapsulation and interface definition to operate together. Finally, two main challenges were related to the heterogeneity: the cross-platform integration; and complexity, i.e., the difficulty of managing the number of components to integrate in a DT. In conclusion, we believe that research is needed to develop a tool for cross-platform integration. Such a tool could be based on technologies such as FMI and DSLs as mentioned by the interviewees, and help to tackle a DT's complexity and heterogeneity.

The model orchestration findings discussed in Section 5.5 show that all interviewees agree orchestration is about proper scheduling of model execution for a specific DT application. Thus, the type of DT application seems to define the trigger to execute a model, role of time for orchestration and how data is exchanged among models. Finally, the main technical challenges are cross-platform and combining continuous and discrete models' execution. In conclusion, research is required to generalize and implement the aspects to orchestrate different domain applications. A DSL seems to be a promising approach, since it was the most frequent technology mentioned. These aspects should be implemented in a tool for model scheduling. Finally, the developed technology should tackle the technical challenges on cross-platform and cross model-type execution. The technological solutions for integration and orchestration should be complementary because DT services' dynamic behavior require proper orchestration. However, before defining the orchestration, the DT integration is required since it defines its components and interconnections.

The verification and validation findings discussed in Section 5.6 show that it is considered an important task in DT development. In addition, they show that the most prominently used method for validation is the informal method of comparing the behavior of VE and the AE. Formal verification and testing has also been used for verification and validation of DTs. Each of these methods have different challenges which have been discussed in Section 5.6. Furthermore, some strategies for validating DTs were put forward such as validation after model reduction, validation by increasing complexity, validation by operating DTs at out-of-bound conditions and continuous validation of DTs. However, each presented technique only validates specific aspects of the system, thus we speculate that a combination of the techniques is required to rigorously validate a DT.

The properties for validation of DTs discussed in Section 5.7 are highly influenced by DT's domain, purpose or application. Several behavioral properties were discussed in the interviews which were specific

to the DT's domain, purpose or application. Moreover, some functional and temporal properties were put forward which are key to address dynamic consistency issues in a DT. One of the main challenges put forward in the interviews were how to measure the quality of DTs. This challenge was further discussed on which properties could be used for measuring quality of DTs and how to quantify these properties. We believe that further research is required to understand how to measure the quality of a DT and the corresponding properties for this purpose.

Finally, the findings regarding the future of DTs from Section 5.8 show that there are three main trends put forward by the interviewees. First, related to the influence of DT in business, interviewees expect that DTs will be assets that can be traded. Furthermore, DT technology will be used in many products; and as a result interaction of DTs between vendors will become common practice. Second, the influence of DT in engineering practices, interviewees shared their view on how DT technology will transform systems to become more autonomous and self-adaptive. Third, they shared how DTs' development will be facilitated in the future, by tackling its main challenges such as interoperability or standardization. Moreover, they shared that in the future tools will facilitate DT development by non-software developers.

Our analyses of the interviewees' expressions showed many challenges around DT development, maintenance, and operations. Yet also emerging were various ways to address or with potential to address these challenges, involving using concepts and results from more classical software and systems engineering. The field holds a lot of promise and need for applying and adapting such 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.

Acknowledgement

This research was funded by NWO (the Dutch national research council) under the NWO AES Perspectief program, project code P18-03 P3.

References

[1] James M. Wilson. "Gantt charts: A centenary appreciation". In: *European Journal of Operational Research* 149.2 (2003). Sequencing and Scheduling, pp. 430–437. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(02)00769-5.

[2] Gerd Behrmann, Alexandre David, and Kim G. Larsen. "A Tutorial on Uppaal". In: *Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, Bertinora, Italy, September 13-18, 2004, Revised Lectures*. Ed. by Marco Bernardo and Flavio Corradini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 200–236. ISBN: 978-3-540-30080-9. DOI: 10.1007/978-3-540-30080-9_7.

[3] S.E. Hove and B. Anda. "Experiences from conducting semi-structured interviews in empirical software engineering research". In: *11th IEEE International Software Metrics Symposium (METRICS'05)*. 2005, 10 pp.–23. DOI: 10.1109/METRICS.2005.24.

[4] Alistair Barros, Marlon Dumas, and Phillipa Oaks. "Standards for Web Service Choreography and Orchestration: Status and Perspectives". In: *Business Process Management Workshops*. Ed. by Christoph J. Bussler and Armin Haller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 61–74. ISBN: 978-3-540-32596-3. DOI: 10.1007/11678564_7.

[5] Parastoo Mohagheghi and Reidar Conradi. "Quality, productivity and economic benefits of software reuse: a review of industrial studies". In: *Empirical Software Engineering* 12.5 (May 2007), pp. 471–516. DOI: 10.1007/s10664-007-9040-x.

[6] Christal Baier and Joost P. Katoen. *Principles of Model Checking*. English. United States: MIT Press, May 2008. ISBN: 978-0-262-02649-9.

[7] Steve Easterbrook et al. "Selecting empirical methods for software engineering research". In: *Guide to advanced empirical software engineering* (2008), pp. 285–311. DOI: 10.1007/978-1-84800-044-5.

[8] Qing Li et al. "Business processes oriented heterogeneous systems integration platform for networked enterprises". In: *Computers in Industry* 61.2 (2010). Integration and Information in Networked Enterprises, pp. 127–144. ISSN: 0166-3615. DOI: 10.1016/j.compind.2009.10.009.

[9] Onur Hisarciklilar, Keyvan Rahmani, and Vince Thomson. "A Conflict Detection Approach for Collaborative Management of Product Interfaces". In: *Proceedings of the ASME Design Engineering Technical Conference* 6 (Mar. 2011), pp. 555–563. DOI: 10.1115/DETC2010-28464.

[10] James M. Joyce. "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327.

[11] Michael Grieves. "Digital twin: manufacturing excellence through virtual factory replication". In: *White paper* 1 (2014), pp. 1–7. DOI: 10.13140/RG.2.2.26367.61609.

[12] Brent Bielefeldt, Jacob Hochhalter, and Darren Hartl. "Computationally Efficient Analysis of SMA Sensory Particles Embedded in Complex Aerostructures Using a Substructure Approach". In: vol. 1: Development and Characterization of Multifunctional Materials; Mechanics and Behavior of Active Materials; Modeling, Simulation and Control of Adaptive Systems. Smart Materials, Adaptive Structures and Intelligent Systems. Sept. 2015. DOI: 10.1115/SMASIS2015-8975.

[13] Milagros Castillo-Montoya. "Preparing for interview research: The interview protocol refinement framework". In: *The qualitative report* 21.5 (2016), pp. 811–831. DOI: 10.46743/2160-3715/2016.2337.

[14] Desmond J. Higham and Nicholas J. Higham. *MATLAB Guide*. Third. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2017, pp. xxvi+476. ISBN: 978-1-61197-465-2.

[15] José L. Barros-Justo et al. "What software reuse benefits have been transferred to the industry? A systematic mapping study". In: *Information and Software Technology* 103 (2018), pp. 1–21. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2018.06.003.

[16] Abdulmotaleb El Saddik. "Digital Twins: The Convergence of Multimedia Technologies". In: *IEEE MultiMedia* 25.2 (2018), pp. 87–92. DOI: 10.1109/MMUL.2018.023121167.

[17] Fei Tao et al. "Digital twin driven prognostics and health management for complex equipment". In: *CIRP Annals* 67.1 (2018), pp. 169–172. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2018.04.055.

[18] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications". In: *IEEE Access* 7 (2019), pp. 167653–167671. DOI: 10.1109/ACCESS.2019.2953499.

[19] Elisa Negri et al. "FMU-supported simulation for CPS Digital Twin". In: *Procedia Manufacturing* 28 (2019). 7th International conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2018), pp. 201–206. ISSN: 2351-9789. DOI: 10.1016/j.promfg.2018.12.033.

[20] Per Erik Strandberg. "Ethical Interviews in Software Engineering". In: *International Symposium on Empirical Software Engineering and Measurement* 2019-Septemer (Sept. 2019), pp. 1–11. ISSN: 19493789. DOI: 10.1109/ESEM.2019.8870192.

[21] Fei Tao et al. “Digital Twin in Industry: State-of-the-Art”. In: *IEEE Transactions on Industrial Informatics* 15.4 (2019), pp. 2405–2415. DOI: 10.1109/TII.2018.2873186.

[22] Michelle E. Kiger and Lara Varpio. “Thematic analysis of qualitative data: AMEE Guide No. 131”. In: *Medical Teacher* 42 (8 Aug. 2020), pp. 846–854. ISSN: 0142-159X. DOI: 10.1080/0142159X.2020.1755030.

[23] Danielle Magaldi and Matthew Berler. “Semi-structured Interviews”. In: *Encyclopedia of Personality and Individual Differences*. Ed. by Virgil Zeigler-Hill and Todd K. Shackelford. Cham: Springer International Publishing, 2020, pp. 4825–4830. ISBN: 978-3-319-24612-3. DOI: 10.1007/978-3-319-24612-3_857.

[24] Weslley Torres, Mark G. J. van den Brand, and Alexander Serebrenik. “A systematic literature review of cross-domain model consistency checking by model management tools”. In: *Software and Systems Modeling* (Oct. 2020). ISSN: 1619-1366. DOI: 10.1007/s10270-020-00834-1.

[25] John Ahlgren et al. “Facebook’s Cyber–Cyber and Cyber–Physical Digital Twins”. In: *Evaluation and Assessment in Software Engineering*. EASE 2021. Trondheim, Norway: Association for Computing Machinery, 2021, pp. 1–9. ISBN: 9781450390538. DOI: 10.1145/3463274.3463275.

[26] Rutger van Beusekom et al. “Dezyne: Paving the Way to Practical Formal Software Engineering”. In: *Electronic Proceedings in Theoretical Computer Science* 338 (Aug. 2021), pp. 19–30. DOI: 10.4204/eptcs.338.4.

[27] Tim Bolender et al. “Self-Adaptive Manufacturing with Digital Twins”. In: *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2021, pp. 156–166. DOI: 10.1109/SEAMS51251.2021.00029.

[28] Mark van den Brand et al. “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)*. 2021, pp. 225–228. DOI: 10.1109/MODELS-C53483.2021.00039.

[29] Mengnan Liu et al. “Review of digital twin about concepts, technologies, and industrial applications”. In: *Journal of Manufacturing Systems* 58 (2021). Digital Twin towards Smart Manufacturing and Industry 4.0, pp. 346–361. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2020.06.017.

[30] Lin Zhang, Longfei Zhou, and Berthold K.P. Horn. “Building a right digital twin with model engineering”. In: *Journal of Manufacturing Systems* 59 (2021), pp. 151–164. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2021.02.009.

[31] Manuela Dalibor et al. “A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins”. In: *Journal of Systems and Software* 193 (2022), p. 111361. ISSN: 0164-1212. DOI: 10.1016/j.jss.2022.111361.

[32] Eklas Hossain. “Introduction to Simulink”. In: *MATLAB and Simulink Crash Course for Engineers*. Cham: Springer International Publishing, 2022, pp. 317–359. ISBN: 978-3-030-89762-8. DOI: 10.1007/978-3-030-89762-8_12.

[33] Marcus Parreiras, Pedro Marques, and Geraldo Xexéo. “Kano Model applied to the evaluation a collaborative board game for teaching in environmental education”. In: *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*. Natal/RN: SBC, 2022, pp. 198–207. DOI: 10.5753/sbgames_estendido.2022.226092.

[34] Angira Sharma et al. “Digital Twins: State of the art theory and practice, challenges, and open research questions”. In: *Journal of Industrial Information Integration* 30 (2022), p. 100383. ISSN: 2452-414X. DOI: 10.1016/j.jii.2022.100383.

[35] Gijs Walravens, Hossain Muhammad Muctadir, and Loek Cleophas. “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)*. Montreal, QC, Canada, 2022. DOI: 10.1145/3550356.3561586.

[36] Victoria Bogachenkova et al. “LaMa: a thematic labelling web application”. In: *Journal of Open Source Software* 8.85 (2023), p. 5135. DOI: 10.21105/joss.05135.

[37] Didem Gürdür Broo and Jennifer Schooling. “Digital twins in infrastructure: definitions, current practices, challenges and strategies”. In: *International Journal of Construction Management* 23.7 (2023), pp. 1254–1263. DOI: 10.1080/15623599.2021.1966980.

[38] Matthew Hause et al. *The SysML modelling language*. https://www.omgsysml.org/The_SysML_Modelling_Language.pdf. Accessed 12 May 2023.

[39] 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. IBM Corporation.

[40] Object Management Group. *Unified Modeling Language Specification Version 2.5.1*. https://www.omg.org/spec/UML/. Accessed 17 November 2022.

[41] Claudius Ptolemaeus. *System design, modeling, and simulation: using Ptolemy II*. https://ptolemy.berkeley.edu/books/Systems/. Accessed 02 January 2023. Ptolemy.org Berkeley.

[42] Unity Technologies. *Unity User Manual (2019.3)*. https://docs.unity3d.com/Manual/index.html. Accessed 02 January 2023.