Towards realization of Digital Twins for systems with coupled behavior

Santiago Gil Arboleda

PhD Dissertation
Department of Electrical and Computer Engineering
Aarhus University
2024

Supervisor
Prof. Peter Gorm Larsen, Aarhus University, Denmark

ISBN: 978-87-7507-569-0
DOI: 10.7146/aul.545

Towards realization of Digital Twins for systems with coupled behavior

Abstract

Manufacturing has undergone a number of revolutions, and automation including robotics and digital transformation are the core of these revolutions. Digital Twin (DT) is one of the enabling technologies to achieve digitalized optimizations. Although DT is a promising technology, it poses several challenges for its theory-to-practice transition and settlement. To this end, this PhD thesis addresses some of these existing challenges in four major areas with the aim of advancing this field and providing methods and insights for its easier adoption by practitioners.

To do so, this PhD thesis combines different research methods, including descriptive, exploratory, conceptual, and applied research. It explores current challenges and proposes methods that conceptualize those challenges; as a result, it brings solutions that address existing needs. To face such needs, applied research and case study research are used to ground the conceptualizations in close-to-real-life settings.

This PhD thesis focuses on four major areas. In the first area, related to tooling and considerations for realizing DTs, a systematic survey on open-source frameworks is conducted and a systematic reporting framework for DT case studies is proposed. In the second area, related to integrating simulation as a fundamental aspect of DTs, an architectural approach that bridges existing frameworks and black-box simulation is proposed. In the third area, regarding the realization of DTs for complex heterogeneous systems, a modeling approach for composed systems and an architectural approach to implement hierarchical DTs with coupled behavior are proposed. In the fourth area, regarding the application of DTs for robotics, two approaches to combine robot-specific methods with DTs are proposed.

The outcomes of this PhD project improve the ease of transfer to other case studies, especially in regards to the applicability in the robotics domain, while enhancing the reusability of existing assets/components and reducing the implementation effort. Moreover, this research can increase the adoption of DT technology, especially in Small and Medium Enterprises and individuals.

Resumé

Fremstillingsindustrien har gennemgået en række revolutioner, og automatisering inklusive robotteknologi og digital transformation er centrale i disse kvantespring. Digital Tvillinger (DTer) er én af de teknologier som gør det muligt at opnå optimeringer via digitalisering. Selvom DT er en lovende teknologi, stiller den mange udfordringer i at omsætte teori til praksis. Denne PhD afhandling adresserer nogle af de nuværende udfordringer, for at fremme dette felt og at frembringe metoder og indsigt i at gøre det nemmere at adoptere for praktiserende udviklere.

Derfor kombinerer denne PhD afhandling forskellige forskningsmetoder inkl. beskrivende, konceptuelle og anvendte. Den udforsker nuværende udfordringer og foreslår metoder der konceptualiserer disse udfordringer. Anvendt forskning og case-baseret forskning bruges for at imødekomme sådanne behov og at koble konceptualiseringerne til scenarier tæt på virkeligheden.

Denne PhD afhandling fokuserer på fire hovedområder. På det første område, der handler om værktøj og overvejelser for at realisere DTer, er en systematisk undersøgelse blevet gennemført og et systematisk rapporteringsframework for DT case-baserede studier er blevet foreslået. På det andet område, der handler om at integrere simulering som en grundlæggende aspekt af DTer hvor en arkitektonisk tilgang, der binder eksisterende frameworks og black-box simulering foreslås. På det tredje område, der handler om realiseringen af DTer for komplekse og heterogene systemer, er en modelleringstilgang for sammensatte systemer og en arkitektonisk tilgang for at implementere hierarkisk DTer med koblet adfærd foreslået. På det fjerde område, der handler om applikationen i robotteknologi, er der to forskellige forslag, der kombinerer robot-specifik metoder med DTer.

Resultaterne af dette PhD projekt gør det nemmere at overføre viden fra andre case-baserede studier, især omkring applikationen i robotteknologi, og forbedrer genbrugligheden af eksisterende komponenter samtidig med at implementeringsindsatsen reduceres. Desuden kan denne forskning øge adoptionen af DT teknologi, især i små og mellemstore virksomheder og for enkeltpersoner.

Thesis Details

Thesis Title: Towards realization of Digital Twins for systems with coupled behavior
Ph.D. Student: Santiago Gil Arboleda
Supervisor: Prof. Peter Gorm Larsen, Aarhus University, Denmark

The content of the thesis is based on the following publications:

[P1] S. Gil, P. H. Mikkelsen, C. Gomes and P. G. Larsen, "Survey on open-source digital twin frameworks–A case study approach," *Software: Practice and Experience*, vol. 54, no. 6, pp. 929-960, 2024, doi: 10.1002/spe.3305.

[P2] D. Lehner, S. Gil, P. H. Mikkelsen, P. G. Larsen and M. Wimmer, "An Architectural Extension for Digital Twin Platforms to Leverage Behavioral Models," 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, 2023, pp. 1-8, doi: 10.1109/CASE56687.2023.10260417.

[P3] S. Gil, P. H. Mikkelsen, D. Tola, C. Schou and P. G. Larsen, "A Modeling Approach for Composed Digital Twins in Cooperative Systems," 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania, 2023, pp. 1-8, doi: 10.1109/ETFA54631.2023.10275601.

[P4] S. Gil, C. Schou, P. H. Mikkelsen and P. G. Larsen, "Integrating Skills into Digital Twins in Cooperative Systems," 2024 IEEE/SICE 16th International Symposium on System Integration (SII), Ha Long Bay, Vietnam, 2024, pp. 1124–1131, doi: 10.1109/SII58957.2024.10417610.

[P5] S. Gil, E. Kamburjan, P. Talasila and P. G. Larsen, "An Architecture for Coupled Digital Twins with Semantic Lifting,", Submitted for publication to the *International Journal on Software and Systems Modeling (SoSyM)*.

[P6] S. Gil, B. J. Oakes, C. Gomes, M. Frasheri and P. G. Larsen, "Towards a Systematic Reporting Framework for Digital Twins: A Cooperative Robotics Case Study,", *Simulation: Transactions of the Society for Modeling and Simulation*, pp. to appear, 2024, in press.

[P7] S. Gil, A. Miyazawa, A. Badyal, P. G. Larsen and A. Cavalcanti, "A Model-based Approach for Co-simulation-driven Digital Twins in Robotics," In progress. To be submitted to the journal *Robotics and Autonomous Systems*.

In addition to the main publications, the co-authored publications that are related to the PhD project are as follows:

[CP8] H. Feng, C. Gomes, S. Gil, P. H. Mikkelsen, D. Tola, P. G. Larsen and M. Sandberg, "Integration Of The MAPE-K Loop In Digital Twins," in 2022 Annual Modeling and Simulation Conference (ANNSIM), 2022, pp. 102–113, doi: 10.23919/ANNSIM55834.2022.9859489.

[CP9] F. Naseri, S. Gil, C. Barbu, E. Cetkin, G. Yarimca, A.C. Jensen, P.G. Larsen and C. Gomes, "Digital twin of electric vehicle battery systems: Comprehensive review of the use cases, requirements, and platforms," *Renewable and Sustainable Energy Reviews*, Volume 179, 2023, 113280, doi:10.1016/j.rser.2023.113280.

[CP10] P. Talasila, C. Gomes, P. H. Mikkelsen, S. Gil, E. Kamburjan and P. G. Larsen, "Digital Twin as a Service (DTaaS): A Platform for Digital Twin Developers and Users," in the 2023 IEEE International Conference on Digital Twin, Portsmouth, United Kingdom, 2023, pp. 1–8, doi: 10.1109/SWC57546.2023.10448890.

[CP11] P. Talasila, P. H. Mikkelsen, S. Gil, and P. G. Larsen, "Realising digital twins", in The Engineering of Digital Twins, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds. Springer, 2024, pp. 225-256, in press.

[CP12] B. J. Oakes, H. Zhang, L. I. Hatledal, H. Feng, M. Frasheri, M. Sandberg, S. Gil, and C. Gomes, "Case Studies in Digital Twins", in The Engineering of Digital Twins, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds. Springer, 2024, pp. 257-310, in press.

[CP13] L. A. Cruz Salazar, S. Gil, G. D. Rueda Carvajal, G. J. Sánchez-Zuluaga and G. D. Zapata-Madrigal, "AI in assessing Industry 4.0 adoption in Colombia: a case study approach," Accepted for publication in the proceedings of the 2024 6th IFAC International Workshop on Advanced Maintenance Engineering, Services and Technology (AMEST).

Acknowledgments

I'm deeply grateful to all the people who have supported me throughout this journey called *PhD*. First of all, I want to thank my beloved wife, *Maria José*, who has been extremely supportive and positive, but also wise, in every aspect, in every decision, of this journey; and my family, especially my brother, *Sebastián*, and my mother, *Martha*, who despite the distance, have always been encouraging me. I'm glad and proud to share this achievement with you, *Maria José*, *Sebastián*, and *Martha*.

I want to thank Peter Gorm Larsen, my supervisor, for guiding me throughout this process, and more importantly, for giving me the chance to do my PhD under his supervision in such a top-quality university in such a wonderful country. I also want to thank Alexandros Iosifidis, my co-supervisor, for the interesting discussions and feedback. This gratitude is extended to Aarhus University and the Digital Transformation Lab in Skjern, who provided the financial support for this PhD project.

I appreciate all the collaborations I had during this process. I want to give special thanks to Daniel Lehner and Manuel Wimmer from JKU University Linz, and Ana Cavalcanti from the University of York, for giving me the chance to conduct research at their research groups abroad. Also, special thanks to Casper Schou, Bentley Oakes, Eduard Kamburjan, and Farshid Naseri for our very interesting and fruitful collaborations.

Last but not least, thanks to all my colleagues at Aarhus University, especially those whom I had the pleasure to work with the most, Peter Høgh Mikkelsen, Zahra Kazemi, Mehmet Can Türk, Jakob Lemming, Mirgita Frasheri, Cláudio Gomes, Daniella Tola, and Prasad Talasila. It's been a pleasure to work with you all during this exciting period of my life.

Aarhus, May 2024 Santiago Gil Arboleda

Glossary

AAS Asset Administration Shell.

AI Artificial Intelligence.

API Application Program Interface.

CAD Computer-Aided Design.

CPS Cyber-Physical System.

DM Digital Model.

DS Digital Shadow.

DT Digital Twin.

DTaaS Digital Twin as a Service.

DTCC Digital Twin Cities Centre.

DTDL Digital Twins Definition Language.

DTL Digital Transformation Lab.

FMI Functional Mock-Up Interface.

FMU Functional Mock-Up Unit.

HMLV High-Mix Low-Volume.

IoT Internet of Things.

MC Merged Characteristic.

OWL Web Ontology Language.

PE Physical Entity.

PT Physical Twin.

RDF Resource Description Format.

RMQFMU RabbitMQ FMU.

RQ Research Question.

SDF Simulation Description Format.

SME Small and Medium Enterprise.

SQWRL Semantic Query-enhanced Web Rule Language.

SWRL Semantic Web Rule Language.

SWT Semantic Web Technology.

Contents

Part I
Overview

1 Introduction

1.1 Motivation and Context

In this era of digitalization, industry needs to keep up with digital technologies, and one of them is the concept of Digital Twins (DTs), first introduced by Grieves in 2003 [14]. DT allows the creation of a digital representation of a physical system to improve its overall operation and perform tests in a risk-free environment [15] as a virtual commissioning mechanism [16]. More importantly, DT is an enabling technology that other emergent digital technologies can build upon, and therefore, it is a key to achieving convergence in digitalized environments. Hence, combining technologies in the areas of modeling, co-simulation, artificial intelligence, and DTs can convey digital solutions where, for example, it is possible to experiment with products and assets before they exist, during their execution, and after they are decommissioned for improving their upgraded versions [17, 18, 19].

Although using DTs seems like a sound alternative, approaching DT solutions for large, heterogeneous systems is a challenging task [20, 21]. This is partially due to the lack of consensus in this domain [P1, 17], where DTs are usually engineered and reported based on case-specific requirements [22, P6], which worsens their generalization to other case studies [23]. Thus, this limits the current adoption of DTs, increasing their costs and reducing sustainability, and thus, Small and Medium Enterprises (SMEs) face difficulties in getting on board this technology, which can be highly helpful in improving the performance of their factories [15].

This PhD project has been conducted in close collaboration with the Digital Transformation Lab (DTL) in the Ringkøbing-Skjern Municipality under the Framework Collaboration Agreement for Aarhus University Digital Transformation Lab-Skjern, targeting the digital transformation of five companies in the area: Vestas, Velux, Millpart, Hydrospecma, and Landia. Each company involved in the project focuses on different products and industries, but they share a common feature, which is that their production fits the paradigm of High-Mix Low-Volume (HMLV). The nature of their production is quite complex, and therefore, reusable and generalizable methods are suitable to address these systems that are rapidly changing, dynamic, and modular. The companies contributed to this PhD project with informal interviews to collect information about their needs and expectations with respect to adopting and using DT technology in their processes.

Hence, the main motivation of this project is to define a general method for creating DTs where flexibility and reusability are required, to reduce the overall time and costs of development, allowing smaller companies to also have the possibility of advancing their digitalization efforts through affordable methods. Therefore, the DTL was used for the experimental research of this PhD project, reflecting upon the use cases and challenges at the industrial partners since none of their particular case studies were used to showcase the methods proposed in this PhD thesis. However, the knowledge adquired has been transferred to companies in periodic update meetings.

1.2 Research Contributions

The main contributions of this PhD are presented below. These are presented in each chapter, where they are described as a major outcome of the research conducted. The contributions are further discussed in Chapter 7.

Contribution 1 (C1) Provided a survey and categorization of open-source DT frameworks.
Contribution 2 (C2) Unified features to be reported/elaborated on for case-independent DT case studies.
Contribution 3 (C3) Outlined an architectural extension to integrate simulation-driven DTs with existing IoT-based DT platforms.
Contribution 4 (C4) Demonstrated a modeling approach for DTs with composition enabled.
Contribution 5 (C5) Proposed an architecture that enables the functional implementation of composed DTs with coupled behavior.
Contribution 6 (C6) Created a mechanism to integrate DTs and the skill-based approach for applications in robotics.
Contribution 7 (C7) Coupled with the RoboSim modeling framework to generate model-based co-simulations for DTs in robotics.

1.3 Research Questions and Hypothesis

We pose the following hypothesis for the overall scope of this project and Research Questions (RQs) to confirm or disprove it:

Hypothesis

A generalizable method for DTs facilitates the realization of complex CPSs where flexibility and reusability are required.

DT is a modern technology with increased attention in both academic and industrial communities in the last decade [P1] as an enabling technology for Industry 4.0 [24] and its successor Industry 5.0 [25]. Although it is a promising enabling technology for digitalization and improvement of existing processes, it poses some challenges related to standardization [P1, 17] and the coordination of multiple cyber-physical components working together [P1, 21, 26]. These challenges become even more difficult when dealing with complex systems, where the investment that is done, in time and money, must return some kind of value, making the DT engineering worthy [27]. In the case of SMEs, these may not have enough budget to approach a DT for large systems at once[15]. Nonetheless, a method that enables the composition of reusable modules can be highly beneficial in these situations, since the development of large systems can be approached by smaller steps [17]. Additionally, if methods are

sufficiently generalizable, transferring the knowledge (or modules) from one case study to another, or to an industrial setting, can facilitate the DT realization process [23, 28].

The following RQs serve to investigate the different aspects of DT engineering for complex CPSs, from identifying the realization fundamentals, to mechanisms that enable reusable components, to more concrete methods to realize DTs in the robotics domain:

Research Question 1 (RQ1)

What are the main challenges in the theory-to-practice transition of DT technology?

This exploratory RQ [29] addresses the understanding of the gap between theory and practice of DT technology. It was used for the preliminary investigation of the overall framework of this project and subsequently led to defining **RQ2** and **RQ3**. **RQ1** is highly related to the industrial-state-of-the-art and needs in ready-to-use solutions.

Research Question 2 (RQ2)

How can existing DT solutions be extended to support an easier integration of (coupled) behavioral models?

This applied RQ [29] aims at improving existing solutions for DTs with mechanisms that enable an easier interface to integrate behavioral models. The rationale for this RQ comes from **RQ1**, where there was a clear gap related to the actual integration of DTs with behavioral models.

Research Question 3 (RQ3)

Is it possible to increase the reusability of DT components so these can be reused in different case studies?

This exploratory RQ [29] aims at finding, if any, suitable generalizable mechanisms to enable the reusability of DT components and structure, so there is less implementation effort time- and money-wise to realize new DTs for different case studies. This RQ is highly related to academic research, although there is a joint interest in both industry and academia in this challenge. The rationale for this RQ comes from **RQ1**, where one of the biggest limitations in the current state of DT technology is the generalizability of methods, so the reusability of DT components is further improved for multiple case-independent scenarios.

Research Question 4 (RQ4)

How to provide suitable methods for the realization of DTs of robotic systems?

This RQ is applied [29] and its intent is to find suitable methods that enable the representation of robotic systems with DTs. Although this RQ is not motivated by the identification of challenges, it is proposed as the application of DTs in robotics is fairly new, and the research can provide relevant knowledge in this domain. Additionally, since the PhD project poses a proof-of-concept within the manufacturing domain, a case study with robotic arms is used as an exemplar. Therefore, **RQ4** also addresses this need. This RQ is highly related to applied research of DT technology in the robotics domain.

Research Question 5 (RQ5)

How to systematically report a DT case study research?

This RQ is applied [29] and its intent is to research generalizable and systematic reporting principles for DT case study research, which are case-independent and based on a consensual basis. The rationale behind this RQ comes from the gaps found in **RQ1**, where there was a significant limitation of standardization and consensus within the DT domain. As some parts of this PhD research are conducted on a case study research basis [30], it is essential to follow systematic reporting principles for a better understanding of readers and easier comparison with other case studies.

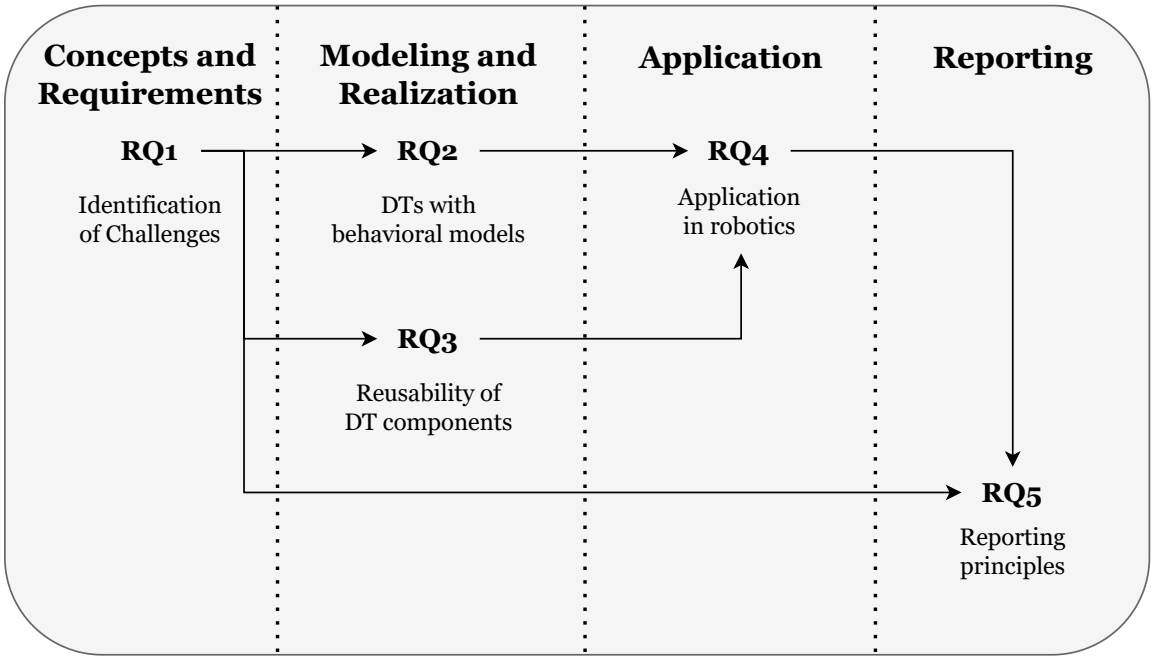

Figure 1.1: Relational connections among RQs across different phases of DT engineering.

Figure 1.1 provides an overview of the relationships among the proposed RQs and how they feed into other RQs. **RQ1** provides the basis for the exploratory research of this project, which is then used as an input in **RQ2**, **RQ3**, and **RQ5**. The outputs from **RQ2** and **RQ3** are used for addressing **RQ4**. Finally, the output from **RQ4** and the output from **RQ1** serve as the input for **RQ5**.

In terms of the nature of each of the proposed RQs, **RQ1** is chosen to be *exploratory* as the area of DT engineering is recent, it is relevant to gain familiarity with this phenomenon and identify gaps to focus the research efforts. **RQ2** and **RQ4** are chosen to be *applied* since they intend to solve particular needs identified in the industrial collaboration, elaborating on the gaps found in **RQ1**. **RQ3** is chosen to be *exploratory* since it intends to address the challenges found in **RQ1** related to the lack of mechanisms to effectively reuse components across different DT applications. On the other hand, **RQ5** is chosen to be *applied* as it addresses the problem of generalization in the reporting principles and foundations for DT case study research.

1.4 Research Methods and Research Methodology

The research methods utilized include [29, 30]:

Descriptive research in the survey on DT frameworks [P1], related to contribution **C1**.
Applied research in publication [P2] related to contribution **C3**, publication [P4] related to **C6**, publication [P6] related to **C2**, and publication [P7] related to **C7**.
Case study research in publications [P1, P2, P3, P4, P5, P6, P7] where the methods are demonstrated and evaluated in a case study research setting [31].
Exploratory/Conceptual research in the survey on DT frameworks [P1], related to contribution **C1**, and in publications [P3, P5] related to **C4** and **C5**.

As the research conducted has several fronts, the methodologies may slightly differ across the research fronts for each contribution. Nevertheless, this PhD project has followed an overall bottom-up research methodology due to its applied research nature. The rationale behind this research methodology is to aim at covering both industrial and academic needs, using a combination of applied, exploratory, descriptive, and case study research [29, 30]. A summary description of the methodology is as follows, which is illustrated in Figure 1.2:

We informally interviewed [32] the five involved Danish manufacturing companies to identify common problems related to digitalization. At the same time, we surveyed technical aspects regarding the realization of DTs in publication [P1]. Both the interviews and survey helped to specify the business problems for our applied research. We selected those problems feasible to be approached by DTs, and subsequently, we investigated the potential challenges to accomplish the solutions for such problems. From the abstraction of those challenges, we moved into finding generalizable solutions to convey both needs using exploratory, conceptual, and case study research.

To this end, we used a set of case studies to reflect upon the settings and challenges of DTs in the manufacturing industry within our scope to apply the methods generically. The methods were assessed for theoretical generalizability [23] using multi-case study research and case-based generalization by architectural similarity [28], such that they are applicable to the proposed case studies and extendable to other case studies.

As a last step, in order to produce reports that are comparable among different case studies, applied research was utilized to investigate a systematic reporting framework for writing DT experience reports.

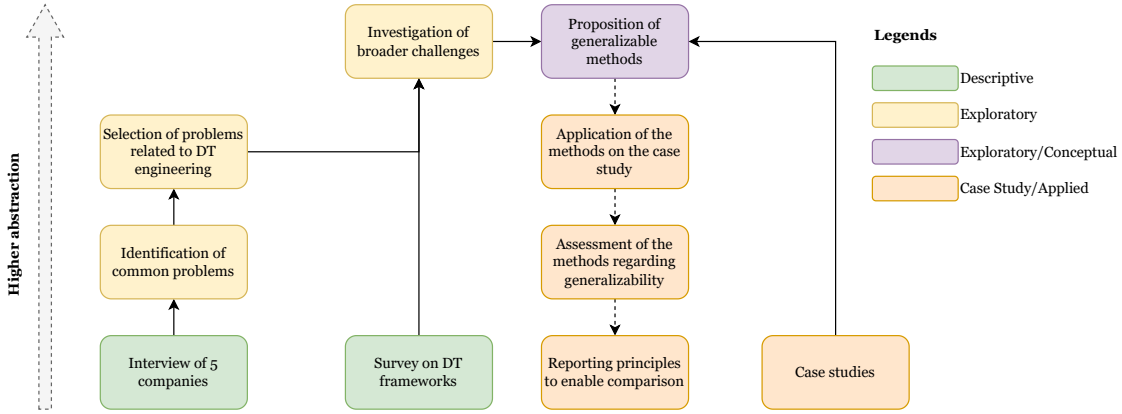

Figure 1.2: Bottom-up research methodology.

1.5 Assessment Criteria

In order to informally self-assess the contributions of this PhD project, we define a set of criteria to evaluate the successful aspects and deficiencies of this thesis. Although the assessment may introduce some subjectivity and bias, the endeavor is to be as objective as possible, which is also supported by the refereed publications provided along with this thesis. Additionally, for each criterion, a *Likert* scale [33] from zero to four is introduced, where *zero* stands for poorly covered and *four* stands for highly or successfully covered. The criteria are as follows:

Ease of transfer The assessment of contributions is tied to their ease of applicability in various industrial contexts. This criterion stands for the adaptability of developed methods in other case studies, including industrial settings.

Reduced implementation effort As one of the expected impacts is to provide mechanisms that can be easily adopted by smaller players, such as SMEs, reducing implementation effort for the realization of DTs is key. This criterion stands for the qualitative and quantitative aspects in terms of engineering effort and time required to perform a certain task.

Enhanced reusability Similar to the previous criterion, reusability is also key to achieving a better adoption of DT technology and is also aligned with the criterion for ease of transfer. This criterion stands for the ease with which the product or portions of the product can be reused in the development of other systems.

Applicability in robotics Since the scope of the PhD is to bring value, in particular to the manufacturing industry, it is relevant to guarantee the applicability of the methods for this particular case, which is focused on robotic systems. This criterion stands for the usefulness of the methods for particular robotic tasks.

Spiderweb charts are used to visually illustrate the assessment of each criterion per contribution. Figure 1.3 exemplifies a spiderweb chart for a test contribution, where such a contribution is assessed with *1* for ease of transfer, *2* for reduced implementation effort, *3* for enhanced reusability, and *4* for applicability in robotics. The assessment of the actual contributions is described in Chapter 7.

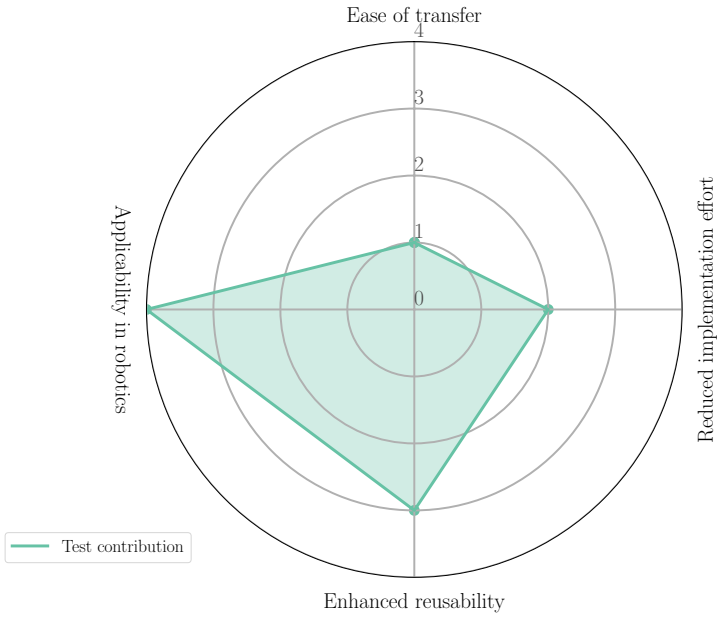

Figure 1.3: Example of Spiderweb chart used for the assessment of a test contribution.

1.6 Outline and Reading Guide

This PhD thesis is written as a 'thesis by article-compilation' consisting of both published and unpublished work conducted during the PhD project. The thesis is split into two parts; Part I provides a self-contained summary of the research conducted throughout the PhD project and Part II contains the publications, on which the summary of Part I is based. An overview of the relations of Chapters 3 to 6 and their corresponding RQs, contributions, and publications is illustrated in Figure 1.4.

Chapter 2 presents the foundational background for the concepts used throughout this thesis. It also introduces the case studies that have been used for the experimental research in the PhD project, which are used later in the subsequent chapters.

Chapter 3 elaborates on the results and analyses of [P1]. Then, it discusses some practicalities of DT technology in relation to theory-to-practice and wraps up with considerations for the DT engineering process about the implementation and reporting based on [P6].

Chapter 4 builds on the gap of existing tooling for DTs to propose an architectural extension to more easily integrate simulation into DT platforms based on [P2]. Then, it elaborates on the concept of *endpoint*, which is used to integrate black-box simulation with DTs.

Chapter 5 refines the method to integrate simulation with DTs by introducing a modeling approach that enables the composition of DTs based on [P3] and an architecture that enables DT systems with coupled behavior based on [P5]. Finally, it presents the details of the evolution of the semantic model for DT systems that the architecture is based upon.

Chapter 6 moves into the field of robotics where most experimental research of this PhD has been conducted. Here, a method to integrate robot skills into DTs is shown based on [P4]. Then, a method to generate model-based co-simulations for DTs in robotics is presented based on [P7].

Chapter 7 presents the discussions, concluding remarks, and potential research directions, which come from the results of the PhD project. Finally, the contributions presented in this thesis are self-assessed according to the criteria introduced in Section 1.5.

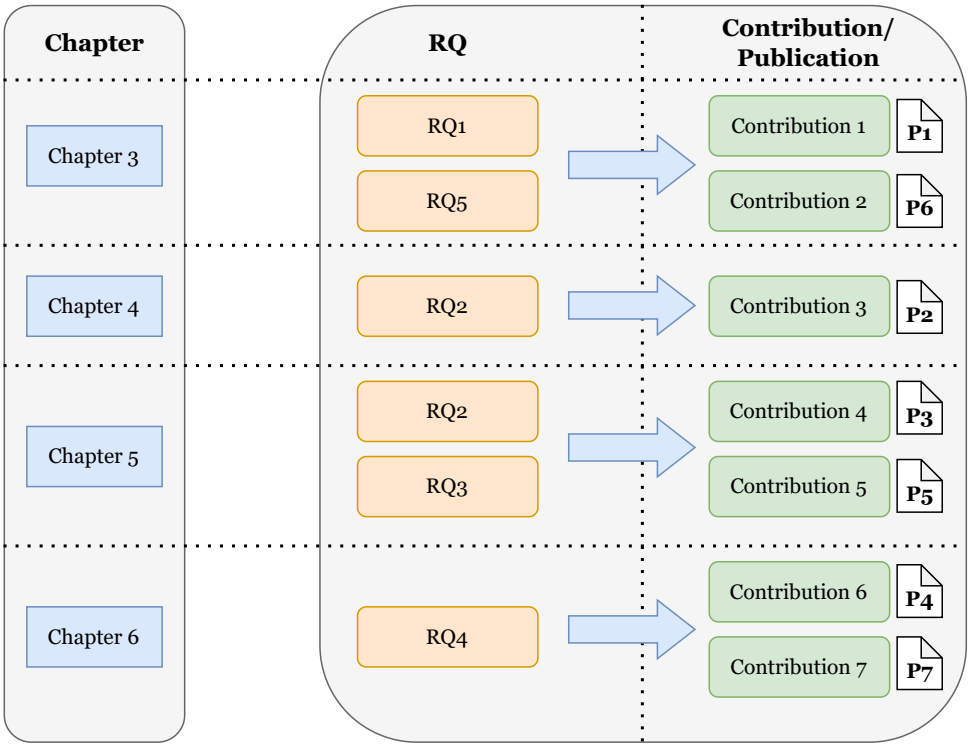

Figure 1.4: Mapping of chapters to RQ-contribution-publication triples.

2 Background and Research Context

This chapter provides the relevant background, research context, and foundations on which the rest of the chapters in Part I are based. Additionally, since case study research [30] has been conducted in this PhD, the case studies that are used throughout the thesis are also introduced.

2.1 Digital Twins

DTs were first introduced by Grieves in 2003 [14] as a concept to virtually represent physical assets, usually called the Physical Twins (PTs) (also called Physical Entities (PEs) in some other studies [34]). DTs are also a representation of Cyber-Physical Systems (CPSs), due to their nature of having physical and cyber attributes [35]. Although there have been multiple definitions of what a DT actually is in literature [27, 36, 37], the definition and categorization by Kritzinger [38] have been highly adopted [39]. Kritzinger's definition compares a DT to its downgraded versions a Digital Shadow (DS) and a Digital Model (DM). A DT has bidirectional communication, so it can be synchronized with data coming from and send actions/insights [22] to its physical counterpart. A DS only has unidirectional communication from its physical counterpart, being able to synchronize, but not to act. A DM, on the other hand, has no automated communication capabilities at all, and thus, it is not able to interact with its physical counterpart.

Even though the categorization of DT, DS, and DM is clear, it is still short in terms of the benefits/usages [34, 22] a DT is expected to provide in reality. A DT should provide some kind of business outcome, ranging from reducing costs and risk to improving efficiency, service offerings, and decision-making [27]. To achieve such a goal, having bidirectional communication capabilities is not sufficient; therefore, a DT must be provided with several additional components, ranging from models, data, tools, and services, depending on the business goals. Models, in particular, are the foundations for a DT so it can behave as expected. In this sense, DTs should include a variety of models, including but not limited to geometric, physical, behavioral, rule, assembly, verification, and management modeling [40].

As a result of the explosion of the DT concept, the DT engineering concept, also called Digital Twinning, has emerged too [41, 42]. Digital Twinning refers to the process that involves the creation of a DT and its multiple components, which are abstractly described by the DT *constellation* [22]. Such a constellation is described by *Models and Data*, *Tools and Enablers*, and *Services or Usages*. In particular,

the *Services*/*Usages* of a DT are what provide the *additionality*, i.e., the value, to a *DT-enabled system* (which is the abstraction delimitation of the joint system where PTs and DTs are twined for a particular business goal), and therefore, make a DT to be *insightful*.

Moreover, although some methods for Digital Twinning are conceptual, the application of this process is usually case-specific [38, 43], depending on the DT's underlying physical process, and therefore, case study research [30] is highly relevant in this field. As a side note, due to the cyber-physical nature of DTs, the guidelines to conduct case study research in software engineering by Runeson and Höst [31] can be adopted. However, Dalibor et al. [17] have identified that Digital Twinning can follow a joint engineering paradigm (where both the DT and PT are engineered together) or an explorative engineering paradigm (where the PT is engineered after the DT), i.e., Digital Twinning does not need to follow a subsequent engineering paradigm (where the DT is engineered after the PT) as commonly perceived.

Continuing on the Digital Twinning process, there are various enabling technologies and tools for realizing DTs [44, 45, 46], including Internet of Things (IoT), modeling and simulation, data analysis, Artificial Intelligence (AI), visualization, and edge/cloud. From the range of tools, it was not until the DT platforms were introduced as the pioneering tools to specifically assist Digital Twinning [47]. These DT platforms, built on top of IoT frameworks, provide the middle-ware to bidirectionally connect DTs and PTs seamlessly. To achieve this, DT platforms use the so-called *DT schemas*, which are usually referred to as *data models* that adopt a particular meta-model. One example is Asset Administration Shell (AAS) which provides a meta-model and specification under the German *Plattform Industrie 4.0* initiative [48], recently turned into the IEC-63278-1 standard [49]. On the other hand, these platforms do not necessarily cover the behavioral aspects of DTs, which are a backbone of DT technology [40] and essential to run *virtual* experiments [18], such as *what-if* simulations [50] and virtual commissioning [16].

Another more advanced concept to realize DTs is the Digital Twin as a Service (DTaaS) platform, where DTs can share their services with other DTs or users using cloud technologies. Through these DTaaS platforms, other mechanisms, such as micro-services and DevOps [51] can be adopted for the quick setup of DT applications.

2.2 Composition of Digital Twins

Composition refers to the representation of the entities as hierarchies of parts [52]. System composition enables composing and decomposing objects and scenes into a hierarchy of meaningful and generic parts while providing context and constraints due to the composition itself [52]. Software entities and applications have also been approached by composition, which has been effective for systems implementation [53]. Additionally, (de)composition is fundamental for supporting reuse in software engineering [17].

The composition of DTs is one of the essential characteristics for their effective use (in heterogeneous settings) and reuse (to avoid building DTs from scratch) [17]. With composition, DTs can, for example, be composed into larger and more complex systems while keeping a lower difficulty, i.e., at the level of the internal sub-components. DT composites can also represent different aspects by spatial or contextual reference [54].

The concept for the composition of DTs has also been presented in other formats or concepts, such as *composition of micro-services* [55], *aggregation* of DTs [56], *systems-of-systems* of DTs [21], and *hierarchical* DTs [57]. However, the composition of DTs is a current challenge in Digital Twinning in several dimensions, including [21, 20, P3] heterogeneity, representation of coupled behavior, interoperability of models and simulation environments, horizontal integration, and vertical composition, among others. It is also relevant that DTs mirror the valid logical composition of their physical counterparts and inherit their constraints and context [58].

2.3 Simulation and Co-Simulation

DTs are highly dependent on modeling and simulation technologies since these provide the functional pillars for the DTs to accurately represent their PTs. Modeling and simulation, together, have been used in engineering for a long time, to analyze, describe, and ask *what-if* questions about existing or conceptual systems [59]. Modeling, on the one hand, refers to the process of creating the models, typically stochastic and dynamic, while simulation refers to the operation of the model of the system in effect [60]. Simulation offers the ability to reconfigure and experiment with (in a risk-free scenario) with the system, more effectively and practically than when doing it with the real system [60].

Standard simulation is not sufficient when the system components are not homogenized into a single object. This is the case for multiple DT systems, which due to their heterogeneity DTs [21], cannot be composed straightforwardly. Co-simulation, on the other hand, enables the coupling of the behavior of multiple heterogeneous systems by composing the simulations of its parts [61], and thus, it is possible to represent the behavioral aspects of coupled but heterogeneous DT systems that are twined to complex physical systems. Co-simulation can be built upon the Functional Mock-Up Interface (FMI) standard[1] [62] and implemented through Functional Mock-Up Units (FMUs) for model exchange and co-simulation. FMUs can be orchestrated to represent the connections of the system in effect by inputs and outputs and can be used in discrete-event-based and continuous-time-based co-simulation settings [61].

Havard et al. [63] and Fitzgerald et al. [64] have proposed the use of co-simulation to approach applications for CPSs and their integration with DTs.

Some enabling tools of co-simulation, which are used later in this thesis include the co-simulation engine *Maestro* proposed by Thule et al. [65], the wrapping tool UniFMU by Legaard et al. [66], which allows wrapping models within the FMI standard format, and the FMI implementation of RabbitMQ, RabbitMQ FMU (RMQFMU) by Frasheri et al. [67], which allows to input to and retrieve data from a co-simulation experiment in real-time.

2.4 Ontological Engineering

Ontologies are formal representations of domain concepts that provide the structure for knowledge bases [68]. The word *ontology* was initially taken from philosophy and stands for a systematic explanation of existence, but it has been provided a different meaning (and use) in AI, as a model that comprises the terms, relations, and rules of a topic area or domain [69]. An ontological model has concepts (classes), individuals (instances), object properties (object-to-object relationships), data properties

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

(object-to-data relationships), and rules/axioms that are used to assert whether the structure and statements given in the model are *true* [69].

The process of designing ontologies is called *ontological engineering*, which involves defining the theme, design methods, applications, knowledge sharing and reuse, and development [68], following a set of principles or good practices [69]. Ontological engineering has vast uses in software engineering, including the use of ontologies for requirements engineering, component reuse, integration with modeling languages, coding support, providing business rules, providing semantic web services, and testing, among others [70].

Ontologies are usually engineered and deployed using Semantic Web Technologies (SWTs). SWTs have been proven to be relevant in the engineering for CPSs, by integrating data from heterogeneous sources, integrating and representing engineering knowledge, and providing access to and analytics on the previously integrated knowledge [71]. SWTs provide the stack to represent knowledge, structures, inference mechanisms. These technologies, which are defined as languages, include Resource Description Format (RDF), Web Ontology Language (OWL), and SPARQL [71]. RDF is used to represent data tiples in the form $Subject—Predicate—Object$. OWL disaggregates *Predicates* into *Object properties* and *Data properties*; OWL also provides cardinality constraints and property characteristics, such as inverse, transitive, reflexive, and symmetric. On the other hand, SPARQL provides the querying mechanism for RDF data represented as triples with inherited capabilities for semantic reasoning. Other SWTs include the Semantic Web Rule Language (SWRL) [72] and the Semantic Query-enhanced Web Rule Language (SQWRL) [73], which provides a querying mechanism on top SWRL.

2.5 Digital Twin Application in Robotics

Robotics gained popularity in the 20^{th} century as a field of science [74]. By the beginning of the 21^{st} century, it brought a significant expansion, especially in industrial applications, addressing hazardous environments and harmful tasks, assisting human operators to reduce fatigue, and developing products [74, 75].

Robotics was attributed as an enabling technology for Industry 3.0, the industry of automation and mass production. However, robots still play an essential role in Industry 4.0, especially as *connected* robots, enabling cooperation among robots and collaboration with humans [76]. Moreover, the recent Industry 5.0 initiative, which introduces the *human-in-the-loop* concept [77], further highlights the need for collaborative robots. The term *cooperative robots* refers to multi-robot systems (in this case multi-arm robot systems) that perform cooperative synchronized manipulation and motion tasks to complete a goal [74]. The newer term *collaborative robots*, also known as cobots, refers to robots that are designed to either assist humans in a specific task or cooperate with humans simultaneously in the same workspace [78].

A variety of industrial applications in robotics are performed with robot manipulators, which are the kinds of robots used in this thesis. Robot manipulators, also known as robotic arms, are a serial chain of rigid bodies (links) connected via motors (joints), and finally attached to an end-effector, can be used in multiple tasks, such as welding and grasping [75, 74].

Robotic arms can be modeled in different ways, including kinematics, such as the Denavit-Hantenberg method, dynamics, such as the Newtow-Euler and Lagrange methods, actuation, and control-related

models, among others [74, 79]. The vast amount of methods, plus the complexity involved in robotic arms, makes the modeling and control of these non-trivial tasks that require a certain level of expertise [80]. As a consequence, including the behavioral representations of robotic arms as part of DTs also becomes a complex task with several challenges [81].

Another relevant concept in terms of robot programming is the *skill-based engineering* concept, which was proposed as a method to disaggregate operations by abstraction level for applications in robot programming, and thus, users and automated planners have more flexibility at the engineering and (re-)programming of routines [82]. This concept disaggregates the term *operation* into *device primitives*, *skills*, and *tasks*. Device primitives refer to the most basic actions, which can not be further disaggregated. Sets of device primitives can be grouped into skills, which define value-added actions. Similarly, sets of skills can be grouped into tasks, which define routines. The skill-based engineering concept has been proven to assist with the programming of collaborative robots for different levels of expertise in robot programming [83].

The application of DT technology in robotics is manyfold. Existing applications of DTs in robotics include space robotics, medical and rehabilitation robotics, soft robotics, human-robot interaction, and industrial robotics [81]. DTs can be twinned to collaborative settings where human(s) and robot(s) share workspaces [84, 85]. This way, robotic arms can be integrated with DTs, so the latter act as an integral control and monitoring complement to provide extended usages in cooperative/collaborative robotics [86]. Current research trends of DTs in robotics include immersive virtual and augmented reality applications, smart manufacturing through robot work-cell simulation, haptic telerobotic applications, and DT-aided AI implementations [81].

The toolboxes for modeling robotics used throughout this PhD thesis are the Robotics Toolbox by Cork and Haviland [87] and RoboSim by Cavalcanti et al. [88]. The Robotics Toolbox provides a suite to represent the kinematics and dynamics of serial-link robotic manipulators, solve inverse kinematics with numerical methods, and approximate trajectories, among other functionalities. On the other hand, RoboSim provides a modeling framework to design, formally verify, and execute simulations of robotic systems. With Robosim, it is possible to (i) model the behavior of state machines for control software in a platform-independent manner, known as *d-model*, (ii) model the physical aspects of platforms, such as links, joints, sensors, and actuators, known as *p-model*, and (iii) model the particular associations between a d-model (state machine) and a p-model (platform-specific model), known as the *platform mapping model*. Additionally, RoboSim has the capability to automatically generate code for d-models as C code and for p-model as Simulation Description Format (SDF) files, enabling a modular approach to implement model-based software for robotic simulators or actual robots.

2.6 Introduction to Case Studies

Since case study research constitutes a part of the applied research methods in this PhD project, the case studies that are used throughout the thesis are introduced in this chapter.

The case studies are introduced incrementally according to their complexity. First, there is the *Three-Tank System*, which uniquely contains the models, that is, it does not count with a PT. Second, and increasing complexity, there is the *Incubator*, which provides a simple exemplar for Digital Twinning and counts with a proper PT. Third and last, there is the *Flex-cell*, which provides a more representative

exemplar that reflects upon industrial settings and counts with a PT composed of four independent assets.

2.6.1 Three-Tank System

The *Three-Tank System* provides a simple exemplar to investigate the multi-system DTs phenomenon, which has been used in publication [P5]. This descriptive case study [30] enables the representation of a system that is composed of three coupled individual components. The Three-Tank System case study does not count with a physical counterpart, thus, it falls into the category of explorative DT engineering [17].

Figure 2.1 illustrates an overview of the Three-Tank System. In this figure, the input of the first tank (i1) is connected to a water supply. The output of this first tank (o1) is connected to the input of the second tank (i2), and the output of the second tank (o2) is connected to the input of the third tank (i3). The output of the third tank (o3) is not connected, representing a sink.

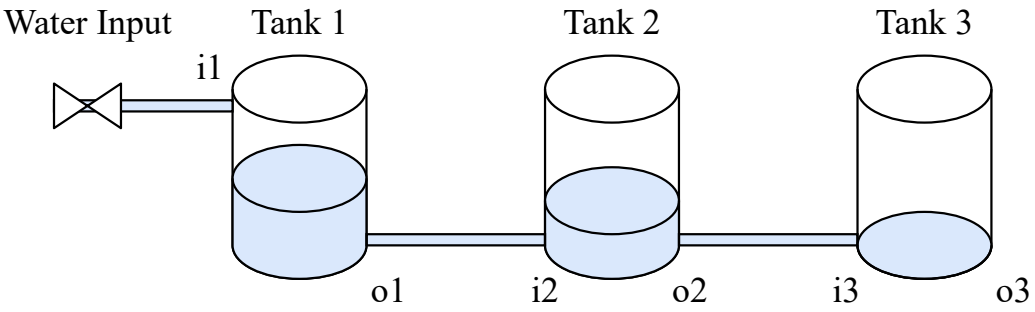

Figure 2.1: Overview of the Three-Tank System.

2.6.2 Incubator

The *Incubator* [89] is a thermal chamber with temperature control that has been used for the incubation of Tempeh, an Indonesian fermented soybean food [90]. It consists of a styrofoam box, a fan, three temperature sensors, a heating device, and a controller. Figure 2.2 displays an overview of the Incubator system. The Incubator DT[2], is the DT realization of the Incubator system, which has been designed using micro-services and has been provided with plant, controller, environment, and Computer-Aided Design (CAD) models. The Incubator DT has been provided with services for state estimation, real-time visualization, anomaly detection, *what-if* simulations, self-reconfiguration, and control policy optimization.

The Incubator DT has served as an exploratory and descriptive case study [30] within the single-system DT phenomenon in publications [P1, P2, P6]. The incubator has also served as a case study within the DT field in [91, 92, 42].

2.6.3 Flex-cell

The *Flex-cell* system is a manufacturing cell composed of four main assets, namely, a Kuka lbr iiwa 7 robotic arm, a UR5e robotic arm, an OnRobot RG6 gripper and an OnRobot 2FG7 gripper. The Flex-cell operates on a plate, which is a shared working environment for both robotic arms. Figure 2.2 illustrates the Flex-cell system.

[2]https://github.com/INTO-CPS-Association/example_digital-twin_incubator

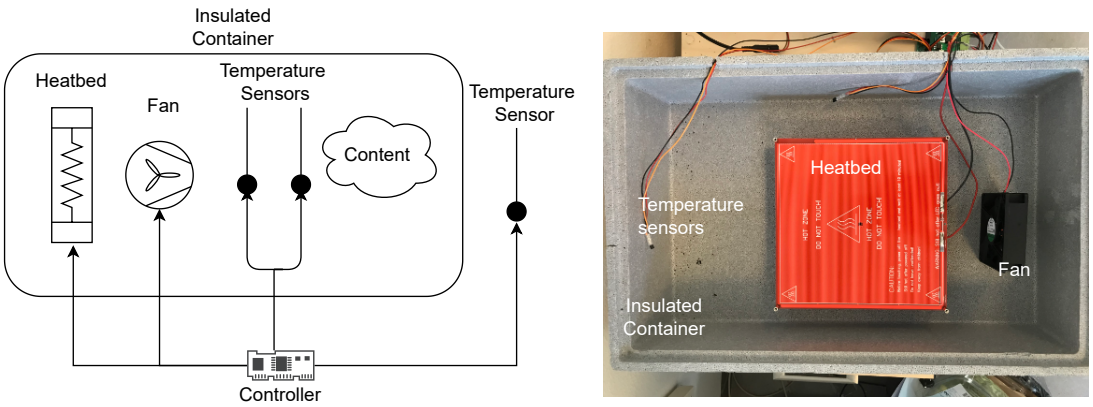

Figure 2.2: Overview of the Incubator system. *Left:* schematic. *Right:* real incubator. Taken from [P1], adapted from [91].

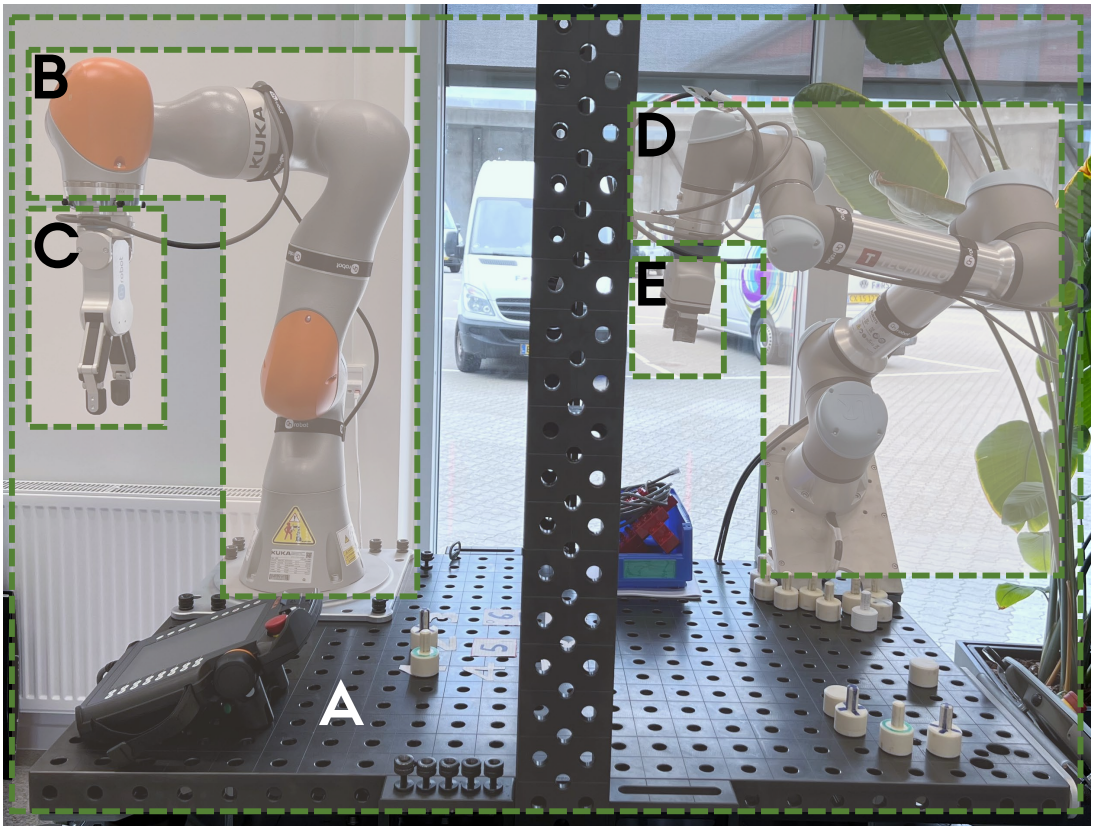

Figure 2.3: Flex-cell system. *A:* plate/workspace. *B:* Kuka lbr iiwa 7 robotic arm. *C:* gripper OnRobot RG6. *D:* UR5e robotic arm. *E:* gripper OnRobot 2FG7.

The Flex-cell has been used as an exploratory and descriptive case study [30] to investigate the phenomena of multi-system DTs and the application of DTs in robotics, which has been used in publications [P3, P4, P5, P6]. In particular, for the phenomenon of DT applied in robotics, the scope is within 2D cooperative assembly and robot positioning. To achieve this goal, the real workspace (x, y, z) of the Flex-cell plate has been transformed into a discrete space (X, Y, Z), which considers the hole distance (5mm) to create a grid of 16x24 for the actionable space.

As for the connectivity of each asset in the Flex-cell, the robots are connected through TCP sockets (the UR5e using the *URInterface*[3] and the Kuka lbr iiwa 7 using the *Kukalbrinterface*[4]) and the grippers through ModbusTCP.

The Flex-cell has provided sufficient complexity to conduct case study research on the phenomena described above, and in particular, concerning i) composable DTs; ii) cooperative systems with coupled behavior; iii) multiple abstraction levels of attributes and operations; iv) complex system with kinematic and dynamic models to account for behavioral aspects in a cooperative setting; and v) multiple robotics applications that include different operations, such as pick-and-place, peg-in-hole, and collision detection, among others.

[3] https://gitlab.au.dk/clagms/urinterface
[4] https://github.com/sagilar/kukalbrinterface

3 Digital Twins: a Challenging and Moving Domain

Digitalized ecosystems in Industry 4.0, usually represented by CPSs, are composed of physical and virtual components. Such virtual components can be approached by DTs, which at the same time, enable bidirectional communication with the physical components. Although the concept of a fully-integrated digital ecosystem sounds promising and has had a positive impact on businesses overall, achieving a complete CPS integration is an extremely challenging task due to several factors, such as multi-scale, multi-system, and multi-model aspects [93]. It is also challenging to cope with and stay ahead of fast-moving technologies [94], such as the emergent DT technology.

This chapter is based on the research output from publications [P1] and [P6]. The research method used in this chapter is mostly exploratory. Descriptive research and applied research are also used, the former regarding the investigation of tools to realize DTs and the latter to address the challenges posed regarding the DT engineering process, which were identified in the exploratory phase.

3.1 Overview

Figure 3.1 presents an overview of the features of this chapter. This chapter presents two contributions of the PhD project associated with Publications [P1, P6] (see Figure 1.4).

The first contribution, **C1**, is the result of surveying and analyzing open-source DT frameworks, which is presented in Section 3.2. The outcomes of this contribution are related to insights into current challenges and good practices to realize DTs, which are presented in Section 3.3.

These challenges also include the lack of standardization and consensus in the DT field, which is further described in Section 3.4. Such deficiencies also involve the lack of structures to provide comparable DT experience reports. In order to provide a consistent mechanism to approach the DT engineering process and its reporting, Section 3.5 presents the second contribution, **C2**, which is the result of the unification of characteristics to be reported on for case-independent DT case studies.

3.2 Tooling for Realizing Digital Twins

DT technology is a fast-moving technology and so are the tools to realize DTs. The pioneering tools to realize DTs are the so-called *DT platforms* [47], which have been DT frameworks provided by major software companies and referred to as platforms. Such platforms are often built on top of

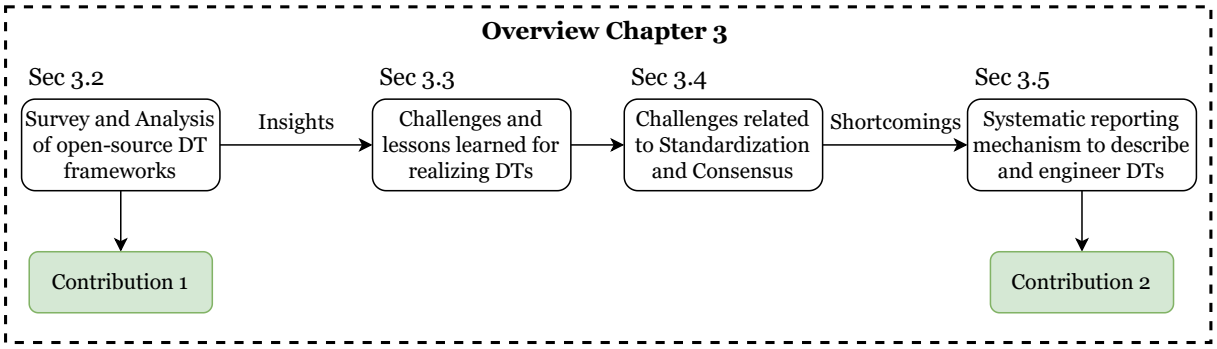

Figure 3.1: Overview of Chapter 3 with the features and sections that lead to contributions.

existing IoT frameworks and provide the capabilities to implement the fundamental requirement for DTs of having bidirectional communication enabled. They also enable the use of model-based engineering implementation by means of metamodels, reducing the implementation effort of simple DT applications with automated interfacing of communication channels, class and object initialization, and data binding. However, these aspects are just *some* of the requirements to build a value-added DT that contributes to a specific business goal [27].

Therefore, with the aim of further investigating tools specially designed to realize DTs, the survey presented in [P1] was conducted. This survey investigates 14 open-source DT frameworks collected from Google Scholar, Google Browser, and GitHub/GitLab. These frameworks are analyzed along 10 dimensions, which are inspired by the pioneer standards for DT technology, the ISO-23247 [95], which provides the recommendations for a *Digital twin framework for manufacturing*. The list of the 14 analyzed candidates is shown in Table 3.1.

The list of the 10 criteria used to analyze the frameworks is as follows:

Communication to identify whether the framework can build DMs, digital generators, DSs, or actual DTs.

Storage mechanisms for data.

Support for analytics to identify the capabilities offered by the frameworks in terms of automated analytics or other reasoning mechanisms.

Compositionality to identify if the framework and its modeling paradigm enable composable or hierarchical DTs.

Support for physical interventions during the operation of the DTs.

Scalability in terms of the deployment of multiple DT instances while preserving the computational performance and efficiency.

Standardization to identify the standards used at the communication, metadata, and behavior levels.

Steps to configure/reproducibility to assess the guidance to build DTs.

Interoperability to assess the capability of the frameworks to build DTs that can perform in heterogeneous environments.

Community support Since some of the frameworks are open-source, it is reasonable to have a sense for the level of long-term support.

Each of the frameworks listed above was descriptively analyzed in each of the dimensions and grouped by similarity into a category. The resulting categories are:

Table 3.1: List of analyzed framework candidates.

Name	Creator	Link
Eclipse Ditto (Framework 1)	The Eclipse Foundation and Bosch	https://www.eclipse.org/ditto/
Equinox (Framework 2)[1]	Murat Artim	https://github.com/muratartim/Equinox
AASX Package Explorer (Framework 3)	The Industrial Digital Twin Association	https://github.com/admin-shell-io/aasx-package-explorer
PYI40AAS (Framework 4)	RWTH Aachen	https://git.rwth-aachen.de/acplt/pyi40aas
SAP I4.0 AAS (Framework 5)	SAP	https://github.com/SAP/i40-aas
Eclipse BaSyx (Framework 6)	The Eclipse Foundation, Bosch, and the Fraunhofer Institute	https://projects.eclipse.org/projects/technology.basyx
NOVA AAS (Framework 7)	NOVA School of Science and Technology - NOVA University Lisbon	https://gitlab.com/gidouninova/novaas
CPS-Twinning (Framework 8)	SBA Research	https://github.com/sbaresearch/cps-twinning
Twined (Framework 9)	Octue Ltd.	https://github.com/octue/twined
Azure Digital Twins Definition Language (DTDL) (Framework 10)	Microsoft Azure	https://github.com/Azure/opendigitaltwins-dtdl
iTwin.js (Framework 11)	Bentley Systems Incorporated	https://www.itwinjs.org/
Digital Twin Cities Centre (DTCC) Platform (Framework 12)	Chalmers University of Technology	https://gitlab.com/dtcc-platform
TerriaJS (New South Wales Digital Twin implementation) (Framework 13)	New South Wales State, Australia	https://github.com/TerriaJS/terriajs
INTO-CPS Co-simulation Framework (Framework 14)	The INTO-CPS Association, hosted by Aarhus University	https://into-cps-association.readthedocs.io/en/latest/

Structured Data DT framework (Category 1): This category represents the frameworks that use structured data under the same metamodel, usually present in DT frameworks evolving from IoT frameworks. These frameworks focus on communication capabilities, and thus, they provide reasonably well support for the automation of communication interfaces. Frameworks 1, 3, 4, 5, 6, and 7 belong to this category.

Domain-specific DT framework (Category 2): This category represents the frameworks that are highly domain-specific, with special tools and modeling approaches for that specific domain, and are difficult to transfer to other domains due to their specialization. Framework 2 belongs to this category.

Language Specification DT framework (Category 3): This category represents the frameworks that provide a domain-specific language and a common metamodel. These frameworks provide the foundational bases for frameworks in the *Structured data DT framework* category. Frameworks 9 and 10 belong to this category.

Geospatial Data DT framework (Category 4): This category represents the frameworks that have a specialization in geospatial representation and management, which are useful in different domains, such as smart cities, transportation, and weather. Frameworks 12 and 13 belong to this category.

3D-based and Infrastructure-oriented DT framework (Category 5): This category represents the frameworks that focus on the visualization and infrastructural aspects of DTs. These frameworks are

usually strong in geometric modeling and interaction with humans for better interpretability. Framework 11 belongs to this category.

Co-simulation and Model-based DT framework (Category 6): This category represents the frameworks that focus more on the behavioral aspects of DTs, and therefore, integrated (co-)simulation and behavioral models are at the core of these frameworks. Frameworks 8 and 14 belong to this category.

Table 3.2 provides a summarized qualitative comparison of the framework categories across the selected criteria based on [P1] by averaging the assessment of the inner frameworks per category. Notice that the categories are not well-balanced since they contain different numbers of members, e.g., *Category 1* contains six members whereas *Categories 2 & 5* only contain one member each. Therefore, this qualitative analysis is intended to give an overview of the average coverage per framework category for each criterion. This overview also helps with the identification of which framework categories are ahead in terms of features for realizing DTs.

Table 3.2: Average qualitative comparison between framework categories with the selected criteria based on [P1] (● = high coverage, ◐ = medium coverage, ○ = low coverage).

Criterion	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6
Communication	●	◐	○	◐	◐	◐
Storage	●	◐	○	●	●	◐
Support for analytics	◐	◐	○	◐	◐	○
Compositionality	◐	○	●	○	◐	◐
Support for physical interventions	◐	○	○	◐	○	○
Scalability	●	○	●	◐	◐	○
Standardization	◐	○	○	○	○	●
Reproducibility	●	○	◐	◐	●	◐
Interoperability	●	◐	◐	◐	◐	●
Community support	●	○	●	●	●	●

Five of the frameworks, namely, SAP I4.0 AAS, Eclipse BaSyx, Eclipse Ditto, iTwin, and INTO-CPS Co-simulation framework, are used to replicate the Incubator DT (refer to Section 2.6.2) based on the expected services to be provided by the DT. Since the services expected to be provided by the Incubator DT are mostly simulation-driven (see Section 2.6.2), the frameworks within the *Co-simulation and model-based DT framework* category perform better than the other categories, in this case, the INTO-CPS Co-simulation framework, performed better than the other four selected candidates.

While conducting this survey on DT frameworks, it was also found that realizing DTs requires a good understanding and implementation of the DT *constellation* [22]. The constellation depends on the combination of multiple models, tools, and services, which are not yet provided as an as-a-whole suite by any DT framework. However, they can be complementary to one another and can be combined to achieve a partial or complete implementation of the constellation. Complementarity can be, for example, combining a framework in the *Structured data DT framework* category to provide automated communication capabilities, a framework in the *3D-based and infrastructure-oriented DT framework* category to provide the visualization and human-readable interface, and a framework in the *Co-simulation and model-based DT framework* category to integrate the behavioral models and perform the simulation-driven experiments. Nevertheless, other tools, not specifically to be used in the DT field,

are also required to achieve the DT constellation. These complimentary tools include visualization, communication, modeling, computation, and data analysis, among others.

After conducting the survey, another open-source framework to realize DTs was proposed by Talasila et al. [CP10]. This framework is designed under the DTaaS concept, and defines a DT by *data*, *models*, *tools*, and *functions*, enabling the reusability of any of such assets to (re-)define more DTs. The framework is called the DTaaS Platform and is available at https://github.com/INTO-CPS-Association/DTaaS.

Contribution 1 (C1):

Provided a survey and categorization of open-source DT frameworks.

3.3 Realizing Digital Twins

Further elaborating on the realization of DTs, there are relevant drawbacks and practicalities to be considered, which come from the findings of the exploratory research of [P1].

In terms of the drawbacks that affect somehow the adoption and reproducibility of solutions, and generalized to a certain extent for all the surveyed frameworks, there are:

Lack of documentation: This worsens the guidance on how to approach different phases of the DT engineering, including (i) how the conceptualization of DTs in the particular framework is approached, (ii) installation tutorials, (iii) creating use cases different from the examples provided, (iv) how to realize *insightful* DTs with value-added services, and (v) how to combine the framework with other tools to achieve a more complete solution.

Lack of representative examples: Usually there are no examples or the provided examples are too simplistic and do not reflect upon settings and complexity in reality.

Lack of public case studies: Public case studies that have been upgraded over time and can help practitioners to use them as a reference for their specific needs having a clear outlook of the DT evolution.

Lack of long-term support: The surveyed frameworks are open-source and can struggle with finding the community and economic support to survive over many years.

Complexity: In alignment with the lack of documentation, most of the DT applications must be created from scratch, which makes DT engineering more complex.

Lack of extensibility to other domains: This limits the usage of the frameworks for several case studies and their integration with other solutions, especially when the frameworks specialize in a particular domain.

Lack of openness with other tools: There are no interfaces or documentation on how to integrate the framework in heterogeneous environments, which is highly relevant to successfully achieve the DT constellation.

As part of the findings for the realization of insightful DTs, the general capabilities regarding automated built-in simulations and data analytics are generally low for all the surveyed frameworks. It means that the creation of DTs that actually contribute to a business goal requires the design and implementation from scratch of case-specific services, as for now, there are no ways to effectively reuse case-independent modules for different DTs in the surveyed frameworks. These automated

features regarding data analytics, reasoning, and decision-making are basically limited to providing Application Program Interfaces (APIs) to externally perform the data analyses; however, regarding built-in simulations, there is a positive outlook, where some frameworks, especially the category for *Co-simulation and model-based DT framework*, can execute cross-platform and *on-the-fly* simulations, for example, using the FMI interface. Therefore, the current effort to realize insightful DTs should be put into mechanisms for automated data analysis, reasoning, and decision-making.

Other relevant practicalities to consider when realizing DTs include:

Orchestration through model-based engineering mechanisms, which enable the rapid setup and reuse of software components.

Insightfulness since a DT that only fulfills bidirectional communication is *"empty"*, a data model which represents the interfaces is not sufficient to contribute to a particular business goal. Therefore, DTs need to be provided with simulation models and mechanisms for inference, reasoning, decision-making, and data analysis. In alignment with this, it is important to know that the interests of different stakeholders regarding DTs may differ, and thus, a DT-enabled system should provide additionality and insightfulness.

Sufficient accuracy because DTs are not only about heavy simulations; they need to be designed in such a way they provide sufficient accuracy to represent their PTs while keeping up with real-time or near-to-real-time requirements. Therefore, the models and reasoning mechanisms should be sufficient in accuracy and time response.

Multiplicities are essential to achieve the coexistence of multiple DTs featuring specialized views of the same PT. Multiplicities can also contribute to a sufficient representation of a PT with multiple, and perhaps heterogeneous, components.

Reusable modules to reduce engineering time and costs, usually assumed when creating DTs from scratch. This idea goes in line with orchestration mechanisms through model-based engineering. Reusable modules for DTs are a current challenge and research focus due to their complexity.

3.4 Standardization and Consensus

There is still a lack of consensus on what a *DT* is, what its minimum functionalities to achieve the diverse purposes are, and how to achieve the composition of heterogeneous DTs, where different communities or technical societies have different interpretations, and thus, expectations of DT technology. This is evidenced in the number of definitions [27, 96, 34, 19] and the lack of standards, with only the ISO-23247 [95] and the IEC-63278-1 [49] standards in this field with a narrow scope (the former focuses only on manufacturing and the latter focuses only on AAS representations).

In line with this lack of consensus, it is also unclear how to assess whether a DT is sufficiently insightful and provides actual value to a particular CPS. Is it enough that it has bidirectional communication enabled? Other components, such as services, models, and reasoning mechanisms should also take part in defining what an insightful and complete DT is. Then, other standards that can positively support the settlement of DT technology in these other dimensions, should be adopted as part of the standardization for DTs. A good example of this, in connection with behavioral models, is the FMI Standard [62].

From the practical side, using the findings from [P1], this lack of consensus affects the common understanding of how to systematically proceed with the Digital Twinning process, since the different categories pose a different expectation of the DT, and therefore, the realization process differs from one another.

As a workaround for this problem, description frameworks for the engineering and reporting process have been proposed by Oakes et al. [22] and Dalibor et al. [17] to provide a better alignment and understanding to a broader community. As an extension to these frameworks [P6] takes these two reference frameworks to provide a systematic reporting framework for DT case study research.

3.5 Considerations for the Digital Twin Engineering Process

With the aim of providing experience reports that are consistent and can be used to facilitate the understanding of practitioners of DT technology and comparison of different case studies, a systematic reporting framework with a unified characterization of DTs has been proposed in [P6].

This systematic reporting framework builds on top of three reference frameworks for DTs, namely, the engineering and reporting mechanisms proposed by Oakes et al. [22] (with 14 fundamental characteristics) and Dalibor et al. [17] (with a four-dimensional feature model) and the characterization proposed by Jones et al. [34] (with 19 themes split into 13 characteristics and seven knowledge gaps). These three reference frameworks are triangulated and systematically merged by meta-characteristics in the phenomenon of DTs following the methodology proposed by Kundisch et al. [97] to merge taxonomies. As a result, the merged framework provides a more complete and unified set of characteristics to report case-independent DT case studies.

Each element of the resulting characteristics after merging the three reference frameworks, named *Merged Characteristics (MCs)*, aligns the terminology and is given a name and a description, which cover the broader but sufficiently specific concept. Additionally, the resulting framework identifies six non-overlapping gaps and three cross-cutting characteristics and provides more specificity to some broad definitions in Jones et al. [34]. Finally, to provide a better ordering to the reporting framework, each MC is sorted by a DT engineering phase, using as a basis the phases proposed by Dalibor et al. [17], namely, *Requirements, Conceptualization, and Design*, *Realization*, *Deployment*, and *Operation*. Table 3.3 provides an overview of the proposed systematic reporting framework.

Contribution 2 (C2):

Unified features to be reported/elaborated on for case-independent DT case studies.

This systematic reporting framework has been proven to be applicable in cooperative robotics (see Section 4 of [P6]) and to be generalizable to other case studies by theoretical generalization [23] under architectural similarity [28], using the Incubator case study (see Section 2.6.2), the Flex-cell case study (see Section 2.6.3), and the Desktop Robotti case study (see [98, 99]).

Table 3.3: Merged characteristics after merging the reference frameworks by Oakes et al. [22], Dalibor et al. [17], and Jones et al. [34]. *Cells in red:* Non-overlapping gaps. *Cells in purple:* Cross-cutting characteristics. *Characteristics in yellow:* Part of the *Requirements, Conceptualization, and Design* phase. *Characteristics in green*:: Part of the *Realization* phase. *Characteristics in blue*:: Part of the *Deployment* phase. *Characteristics in orange*:: part of the *Operation* phase.

Oakes et al.	Dalibor et al.	Jones et al.	Merged Characteristic	Description
System-under-Study	Counterpart	Physical Entity; Physical Environment; Physical Processes	**MC1: System-under-Study**	Describes the SUS, i.e., the PT, of the system of interest.
Acting Components			**MC2: Physical acting components**	Describes the available acting components in the DT constellation, i.e., the mechanisms the DT can use to act on the PT.
Sensing Components			**MC3: Physical sensing components**	Describes the available sensing components in the DT constellation, i.e., the mechanisms the PT can use to transfer data to the DT.
Data Transmitted	Inputs and Events	Technical Implementations; Physical-to-Virtual Connection; Parameters	**MC4: Physical-to-Virtual Interaction**	Describes the interactions from the physical world to the virtual world, i.e., the data transmitted from PT to DT, including inputs and events that the DT processes.
Insights / Actions	Outputs; Asset Interaction	Technical Implementations; Virtual-to-Physical Connection; Parameters	**MC5: Virtual-to-Physical Interaction**	Describes the interactions from the virtual world to the physical world, i.e., the data transmitted from DT to PT, including outputs the DT generates as part of its services.
Services	Optimization	Perceived Benefits; Use Cases	**MC6: Digital Twin Services**	Describes the services, such as optimization, task planning, and visualization, which the DT provides to the users and the physical system.
Time-scale		Twinning and *Twinning Rate*	**MC7: Twinning Time-scale**	Describes the time-scale use and the time rates for the DT services and DT-to-PT synchronization.
Multiplicities	Multiple Representation	Integration between Virtual Entities	**MC8: Multiplicities**	Describes the multiplicities, i.e., the internal twins that compose the DT system, which can be implemented in a centralized or decentralized way.
Life-cycle Stages	Usage Phase; Representation Phase	DT across product Life-Cycle	**MC9: Life-cycle stages**	Describes the lifecycle phases in which the DT takes place. It also informs which representation phase the DT covers of its physical counterpart, i.e., as designed (ideal), as manufactured, or as operated.
Models and Data	Implementation	Virtual Entity; State; Parameters	**MC10: Digital Twin Models and Data**	Describes the DT components, including available models and data, and their role in the DT constellation.
Enablers	Tools	Technical Implementations	**MC11: Tooling and Enablers**	Describes the tools or enablers that are used to achieve the goals of the DT, i.e., they enable the DT to provide the DT services.
Constellation	Consists Of	Virtual Environment; Virtual Processes	**MC12: Digital Twin Constellation**	Describes the orchestration of the DT system, components, and services as a whole.
Evolution	Process	*Twinning* and Twinning Rate; DT across product Life-Cycle	**MC13: Twinning Process and Digital Twin Evolution**	Describes the engineering process involved in the DT implementation, including the development process, quality assurance, and definition of requirements. It also informs on the milestones of the DT engineering process over time and intended upgrades.
Fidelity Considerations	Process: *Quality Assurance*	Fidelity; Levels of Fidelity	**MC14: Fidelity and Validity Considerations**	Describes the fidelity and validity considerations behind the models that constitute the DT, including verification and validation mechanisms, uncertainty, and errors.
	Connection	Technical Implementations; Virtual-to-Physical Connection	**MC15: Digital Twin Technical Connection**	Describes the technical network connection details between PT and DT, including the network protocols and architectures.
	Hosting	Technical Implementations	**MC16: Digital Twin Hosting/Deployment**	Describes the technical hosting aspects of the DT and the associated technology.
Insights / Actions	Decision Making	Use Cases	**MC17: Insights and Decision Making**	Defines the insights and decision making, i.e., indirect outputs of the DT, which have no direct effect on the PT, such as update of parameters, plans, and so on.
	Horizontal Integration	*Integration* between Virtual Entities	**MC18: Horizontal Integration**	Describes the information exchange with external information systems not limited to other DTs.
			MC19: Data Ownership and Privacy	**Refers to the ethical and technical aspects regarding data ownership and data privacy. *Is the data owned by the PT owner or by the DT service provider?***
			MC20: Standardization	**Refers to the standards being followed for the engineering of the DT and its components.**
			MC21: Security and Safety Considerations	Refers to the ethical and technical aspects regarding data cybersecurity and safety on operation. *Can a DT execute operations remotely on a PT where there may be accidents with humans?*

4 Simulation-driven Digital Twins

As claimed in Section 2.1 and Section 3.3, behavioral models and simulation are essential to provide sufficient additionality and insightfulness to DTs. However, the findings in [P1] have shown that the capabilities of existing DT frameworks regarding built-in simulation are still limited at different levels for the different categories of open-source frameworks surveyed. Hence, mechanisms that enable the automatic integration of behavioral models, and therefore, simulation, are highly needed for the continuous evolvement of DTs.

This chapter is based on the research output from publication [P2]. The research method used in this chapter is exploratory and applied, the former is used for the proposition of the method to easily integrate simulation and DTs and the latter for the case study approach to demonstrate and evaluate the method.

4.1 Overview

Figure 4.1 presents an overview of the features of this chapter. This chapter presents one contribution of the PhD project associated with Publication [P2] (see Figure 1.4), **C3**, which is the architectural approach to easily integrate simulation with DTs.

The challenges in relation to integrating simulation as a fundamental aspect of DTs, firstly identified in the survey presented in Section 3.2, are further described in Section 4.2. As a solution for this challenge, Section 4.3 proposes the architectural approach related to **C3**.

One of the key aspects of the architecture is the feature to integrate black-box simulation using, among others, the FMI interface. Thus, Section 4.4 presents the details of how the FMI interface and FMUs are adapted to the architecture.

4.2 Challenges in Integrating Simulation

Even though simulation is a critical factor in DT-enabled systems to provide sufficient additionality, its integration is not an easy task and a current limitation of DT platforms [P1], which fall into the category of *Structured Data DT framework* (see Section 3.2). In [P2], it has been identified that when realizing DTs with DT platforms requires the simulation to be treated externally from the DT services

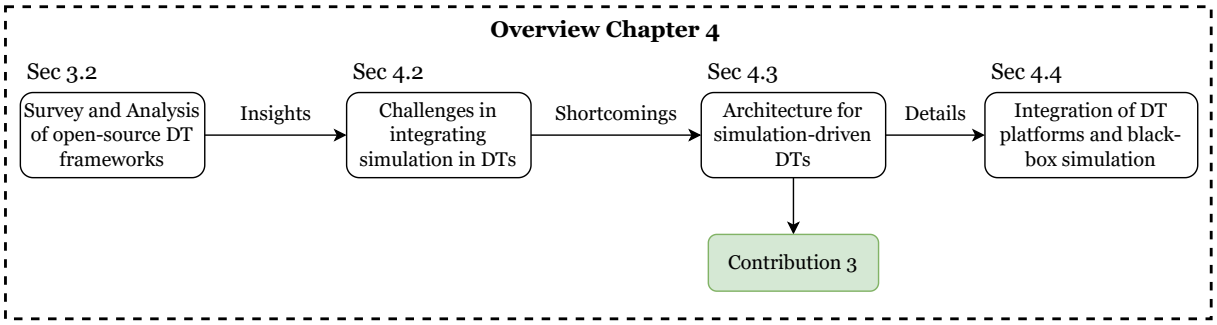

Figure 4.1: Overview of Chapter 4 with the features and sections that lead to contributions.

rather than having it integrated as a fundamental component of the system. This fact leads to more challenges and architectural considerations when integrating simulation with DT platforms.

In fact, the interface between DT platforms and simulation needs to be customized or created from scratch if the platform in effect does not provide an interface to the simulation unit in effect. The problem is that simulation software is diverse, and therefore, heterogeneous, which further complicates the integration process of simulation and DT platforms. Hence, simulation becomes a *DT service* rather than a fundamental component of the DT itself.

Considering the above-mentioned FMI interface (see Section 2.3), a platform- and vendor-independent interfaces, or channels, to integrate simulation becomes crucial to address the complexity involved when dealing with diverse simulation software.

4.3 An Architecture for Simulation-driven Digital Twins

As a workaround to integrate simulation while still using existing DT platforms, which are indeed useful in terms of connectivity and data modeling capabilities, an architectural extension to integrate behavioral models has been proposed in [P2]. Such an architectural extension proposes an abstraction, where a PT and a simulation-driven DT behave similarly through the `Endpoint` abstract interface, as shown in Figure 4.2. The purpose of having this abstraction is to enable the use of *experimental* DTs [18] as entry points for multiple virtual testbeds. This approach is hereafter called *TwinManager*.

This abstract interface enables the specialization of particular interfaces to connect to different endpoints, these being either PTs or simulation units, which in this case, refer to the behavioral aspects of the DTs. Therefore, both the PTs and DTs can be managed at the same abstraction level, i.e., the same class, namely, `Twin`. More precisely, if an object of the class `Twin` has an interface `Endpoint` with specialization to connect to a physical system, such an object is the representation of a PT. Similarly, if an object of the class `Twin` has an interface `Endpoint` with specialization to connect to a simulation unit, such an object is the representation of a DT.

The `Endpoint` interface provides the methods to administrate the twins in the DT-enabled system transparently of their final endpoint, i.e., for PTs and DTs and it can also be specialized with as multiple particular interfaces as needed. Currently, the `Endpoint` interface has been provided with the following specializations: the `MQTTEndpoint` to connect to MQTT brokers[1], the `RabbitMQEndpoint` to connect to

[1] https://mqtt.org/

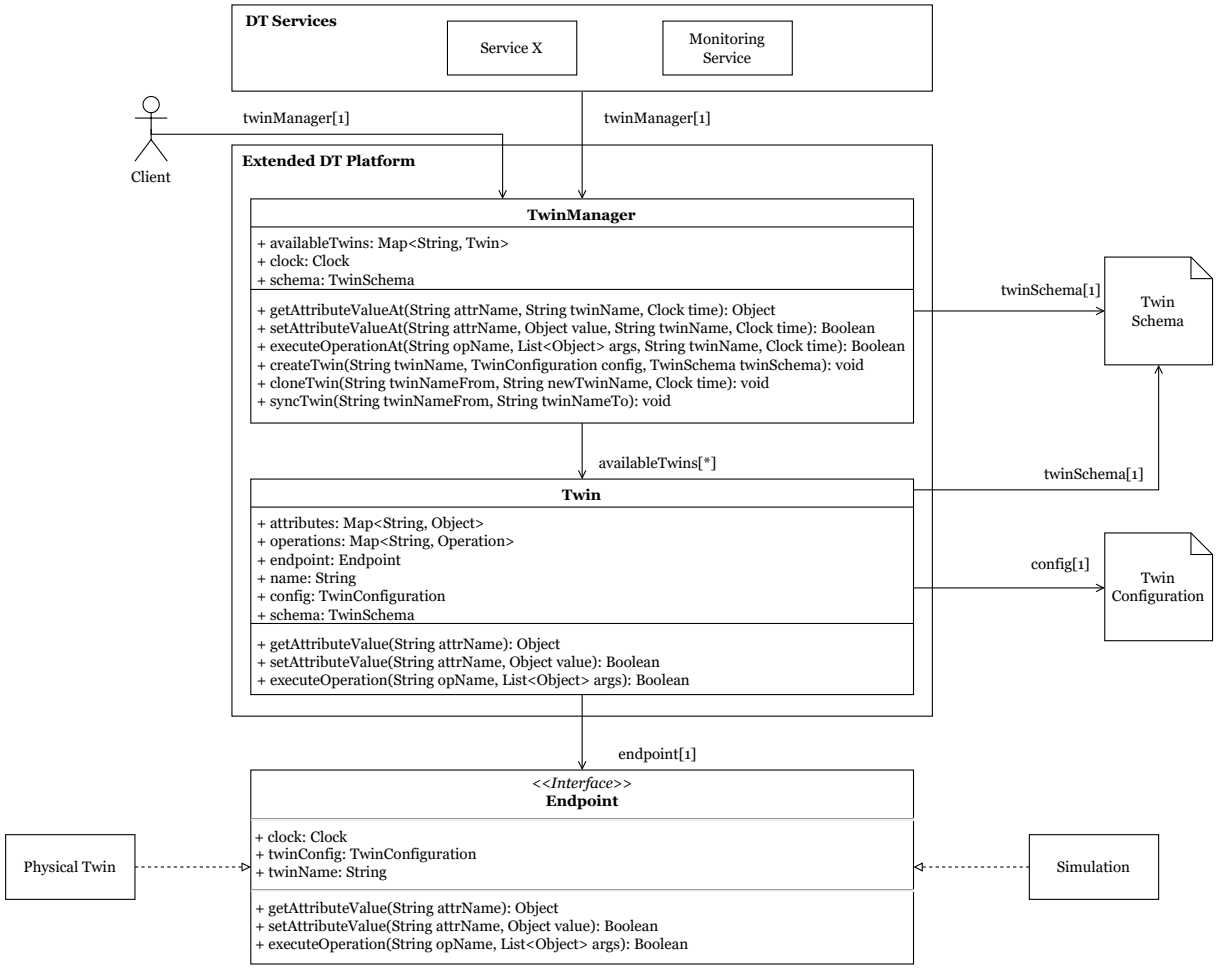

Figure 4.2: Architecture of the extended DT platform with the TwinManager approach. Taken and modified from [P2].

RabbitMQ brokers[2], the `HenshinEndpoint` to connect to the Eclipse Henshin modeling interface[3], and the `FMIEndpoint` to connect to behavioral models wrapped as FMUs under the FMI interface.

The methods in the `Endpoint` interface are accessible by the clients, e.g., a user or application, through the `TwinManager`, which follows a Façade pattern [100]. These methods have been designed having in mind four requirements, as follows:

Requirement 1: Bi-directional communication between DT and PT A DT should be able to read and write information from/to its PT. This is possible via the methods `getAttributeValue` and `setAttributeValue` that can be used to get and set the respective attribute value in a particular endpoint. The method `executeOperation` can be used to run a specific operation in the set of operations contained in the *Twin Schema*. As a remark, the method `setAttributeValue` is preferred to write indirect actions (a.k.a. *insights* [22]) on the PT, such as parameter updates, while the method `executeOperation` is preferred to write, or more precisely, execute, direct actions on the PT. This requirement does not consider other aspects involved in the bi-directional communication, such as connection speed, delay tolerances, and security.

[2] https://www.rabbitmq.com/
[3] https://projects.eclipse.org/projects/modeling.emft.henshin

Requirement 2: Bi-directional communication between DT and simulation A DT should be able to read and write information from/to its simulation unit. Since the `Endpoint` interface behaves transparently, the methods `getAttributeValue`, `setAttributeValue`, and `executeOperation` can be used indifferently on the endpoint of a PT or the endpoint of a simulation unit. However, the number of observations (in this case, *attributes*) and executable operations in the simulation may differ from the number of observations and executable operations in the physical system.

Requirement 3: Ability to create a new experiment of the PT during runtime Taking as a basis the concept of multiplicity (see Section 3.2), a PT could have multiple DTs featuring different *what-if* scenarios with, perhaps, different behavioral models. Therefore, the method `createTwin` enables creating new DTs connected to the given simulation endpoints (according to the DTs' configuration files) during runtime. Additionally, the method `cloneTwin` enables cloning a twin based on the current state and endpoint of an existing twin, which allows, for example, to run two different experiments simultaneously on the "same" twin during runtime.

Requirement 4: Synchronization The instances of the `Twin` class, being either PTs and/or DTs, should be able to synchronize in the present and future. Particularly for DTs, these should be also able to synchronize in the past (because they are in a virtual space). To achieve the different functionalities of synchronization, the methods `getAttributeValueAt` and `setAttributeValueAt` enable the synchronization of attributes considering time as a factor. Similarly, the method `executeOperationAt` synchronizes an operation at the given time delta. In this case, the time, managed by timesteps, can be passed as an argument to the method in continuous or discrete time (as time in seconds for the former or number of steps for the latter) as positive (for the future) or negative (for the past). Additionally, the method `syncTwin` allows to pass the state, i.e., the values of all attributes, from one twin to another.

Following the recommendations proposed in Section 3.3 and Section 3.5, DT-enabled systems are initialized using model-driven engineering mechanisms. To do so, the *Twin Schema* and *Twin Configuration* components in Figure 4.2 are used. The former defines the data modeling structure of twins in the DT-enabled system, namely, *attributes* and *operations* of the twin-specific class. The latter defines the information to interface the twins to particular endpoints and a mapping of the endpoint-specific naming to the Twin Schema's definitions. This way, the endpoint parameters for a specific twin are derived from the Twin Configuration, and the twin instance and its endpoint are initialized using the Factory pattern [100]. Figure 4.2 presents the architecture and relationships of classes and components using the TwinManager approach.

Contribution 3 (C3):

Outlined an architectural extension to integrate simulation-driven DTs with existing IoT-based DT platforms.

The architectural extension has been demonstrated using case study research in a multi-case study setting [23]. The first case study uses a simulation-only DT-enabled system example, called the stack (see [P2]), where a `HenshinEndpoint` specialization has been used. The second case study uses the Incubator (see Section 2.6.2), where an `FMIEndpoint` specialization has been used.

Additionally, this approach has also been evaluated. The evaluation analyzes the difference in using this approach for realizing simulation-driven DTs in comparison to existing approaches, namely, doing

it completely from scratch (baseline implementation **I**) or doing it with existing DT platforms (baseline implementation **II**). More precisely, the evaluation measured the implementation effort and adaptation effort in comparison to the three implementations (baseline **I**, baseline **II**, and TwinManager approach) by counting the number of high-level operations required to achieve a given scenario.

The evaluated scenarios are as follows:

Scenarios for evaluating implementation effort. In these scenarios, the operations to set up a simulation-driven DT for a particular case study are counted.

1. Setup simulation.
2. Execute operation.
3. Creating a new DT.
4. Get future value.
5. Synchronize in the future.
6. Change simulation endpoint.
7. Add a new (extra) simulation endpoint.

Scenarios for evaluating adaptation effort. In these scenarios, the operations to switch from realizing the DT for one case study to realizing the DT for another case study while reusing as many components as possible are counted.

1. Switch to a new PT (i.e., a new DT-enabled system).
2. Adapt the simulation to the new PT.
3. Change time nature (from continuous-time to event-based or vice versa).
4. Reconfigure a DT service.

After comparing the two baseline implementations and the proposed architectural extension, it has been shown that the TwinManager performs better in terms of implementation effort by 57% and 56% and in terms of adaptation effort by 51% and 39% compared to baseline implementation **I** and baseline implementation **II** respectively.

4.4 Bridging Digital Twin Platforms and Black-box Simulation

One of the major findings of [P2] is the specialization of the `Endpoint` interface to work with the FMI interface, and thus, FMUs, i.e., the `FMIEndpoint`. The `FMIEndpoint` allows bridging the gap between existing IoT-based DT platforms and black-box simulation in a platform- and vendor-specific manner.

Using the `FMIEndpoint`, any behavioral model wrapped as an FMU can be attached to DTs on DT platforms, enabling simulation to be a fundamental component of DTs realized by this approach. This endpoint wraps the methods offered in the FMI interface into the methods provided by the `Endpoint` interface, which is basically, translated into the read and write methods of the FMI interface and the `doStep` operation (refer to the FMI standard[4] for more information). This way, an instance of the `TwinManager` class has control to administrate FMUs by updating the simulation model variables and state (managed as attributes) with the method `setAttributeValue` and stepping the simulation at given timesteps with the method `executeOperation`. After each step, such an instance can query the available observations stored as model variables (managed as attributes) in the simulation with the method `getAttributeValue`.

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

Since FMUs behave as simulations, which can also be stepped in real-time, administrating the synchronization in time becomes easy for past, present, and future, as required by the requirements listed in Section 4.3.

For example, consider a simulation setting with an FMU with a fixed timestep $\Delta_{ts} = 0.5$, an experiment to synchronize an attribute in the future, e.g., in five seconds, can be achieved by using the method `getAttributeValueAt` with an argument time delta $\Delta_t = 5.0$. This, in the background, is translated into stepping 10 times the FMU and running the method `getAttributeValue` when Δ_t has passed. This example is representative of a what-if scenario of *what will happen if*.

Similarly, performing an experiment to execute an operation in the past with the same settings, represented by `executeOperationAt` with an argument time delta Δ_t, requires initializing the FMU at time zero and passing the state of the PT in the past given $\Delta_t = 5.0$, coming, for example, from a database. The state is then passed to the simulation by the method `setAttributeValue`. At this point, the simulation, and thus, the DT, are synchronized in the past. Finally, the method `executeOperation` achieves the execution of an operation in the past, which is representative, for example, of a what-if scenario of *what would have happened if*.

The current prototype implementation of the `FMIEndpoint` specialization uses the JavaFMI library by Siani[5] and is limited to the FMI operations *fmiDoStep*, *fmiEnterInitializationMode*, and *fmiTerminate*. Figure 4.3 presents an overview of the `FMIEndpoint` interface and its mapping to the FMI interface to access black-box behavioral models and run stepped simulations.

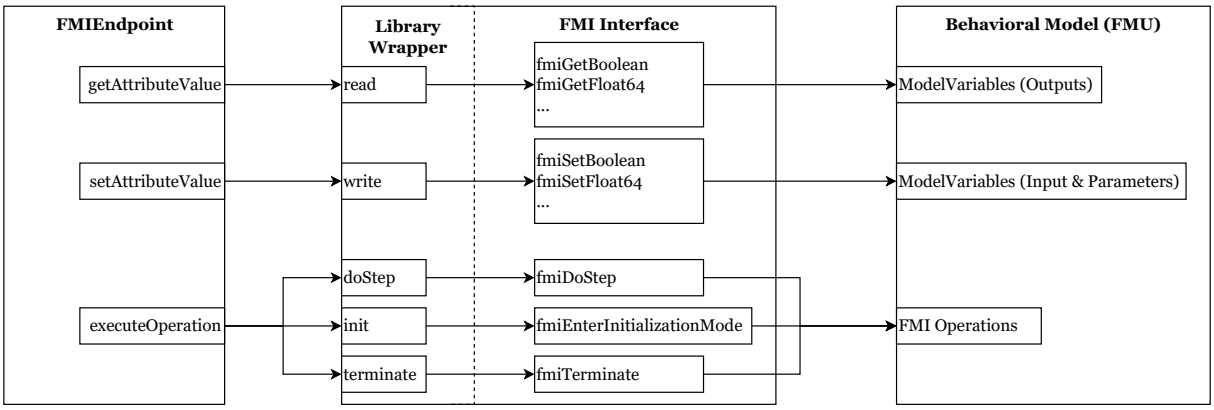

Figure 4.3: Overview of the FMIEndpoint interface and its mapping to the FMI interface.

As a result, a DT implementing the `FMIEndpoint` interface can easily be set up from black-box models, and it will be comparable to its DT, being able to respond in real-time by stepping the simulation as time goes. Additionally, such a DT can also be decoupled from its PT, or even cloned into decoupled versions of the same PT, to run what-if scenarios and provide the additionality required by DT-enabled systems.

[5] https://bitbucket.org/siani/javafmi

5 Digital Twins for Systems with Coupled Behavior

Although the idea of having DTs integrated with behavioral models and simulation at all times is quite interesting and useful, integrating simulations of composed heterogeneous CPSs is much more challenging and not so straightforward. At this point, the approach presented in Section 4.3 only works for single-system DTs or decoupled multi-system DTs. Composing DTs is a current challenge of DT technology as mentioned in Section 2.2 and Section 3.4. One of the critical factors of the composition of DTs is dealing with the internal coupling of the sub-components of the physical system on the simulation side. Ideally, addressing this challenge also requires focusing on the generalizability and reusability of the solution, so it can be extended to multiple DT case studies in different domains.

This chapter is based on the research output from publications [P3] and [P5]. The research methods used in this chapter are conceptual, exploratory, and applied. Conceptual research is used to propose a conceptual modeling approach for DTs with composition enabled. Exploratory research is used to find a method to realize DT-enabled systems with coupled behavior. Finally, applied research is used to first, address the challenges that are faced in this matter in the DT field, and second, for the case study research that is used to demonstrate and evaluate the methods.

5.1 Overview

Figure 5.1 presents an overview of the features of this chapter. This chapter presents two contributions of the PhD project associated to Publications [P3, P5] (see Figure 1.4), namely, **C4**, which is the modeling approach for DTs with composition enabled, and **C5**, which is the extension to the architecture presented in Section 4.3, to support DT systems with coupled behavior.

Section 5.2 takes the challenges concerning the composition of DTs mentioned in Section 3.4 to present a conceptual modeling approach related to **C4** that provides an abstraction to compose DTs.

The implementation of this modeling approach still faces challenges regarding the composition of DTs with coupled behavior, which was also a limitation of the architecture presented in Section 4.3. Therefore, Section 5.3 presents the extension related to **C5**, so the composition of DT systems with coupled behavior is enabled.

A key aspect of this extension is the inclusion of co-simulation within the architecture, which is further described in Section 5.4.

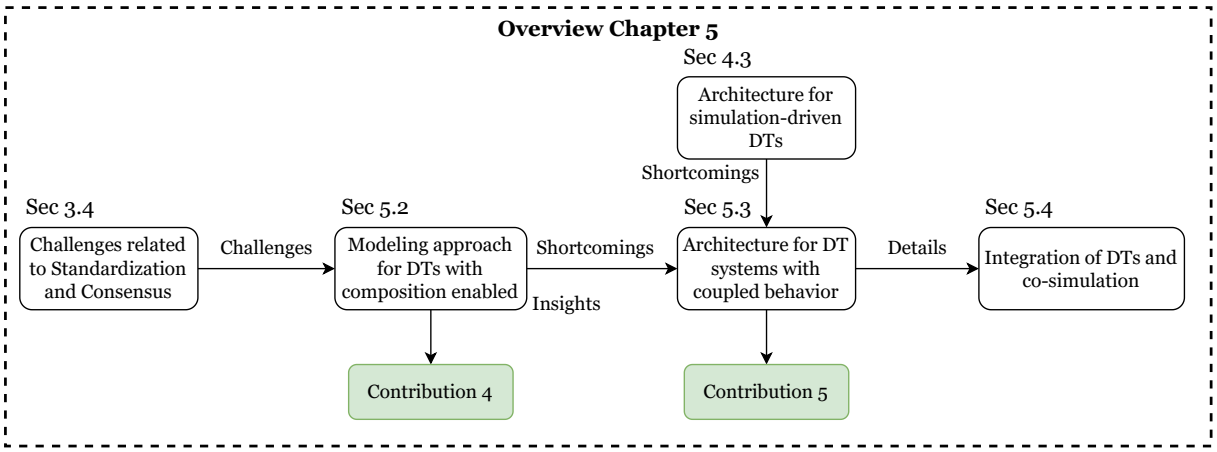

Figure 5.1: Overview of Chapter 5 with the features and sections that lead to contributions.

5.2 Digital Twins for Composed Systems

As a first approach to deal with composed DTs, [P3] proposes a three-layer architecture to represent DTs in (i) the individual aspect (*Model Layer*), (ii) the cooperative aspect (*Semantic Relationship Layer*), and (iii) the composition as larger systems (*Composition Layer*), as shown in Figure 5.2.

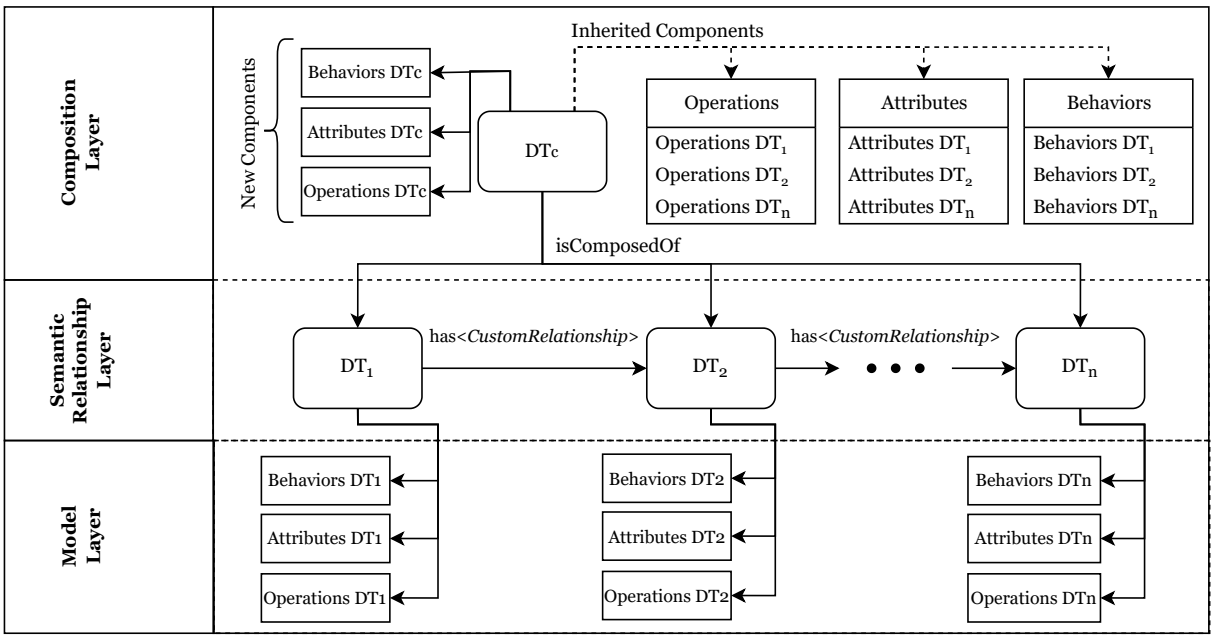

Figure 5.2: Three-layer architecture of the modeling approach for DTs with composition enabled.

The first layer, in alignment with the architecture presented in Section 4.3, proposes the definition of a DT as a collection of *Attributes*, *Operations*, and *Behaviors*. *Attributes* contain the state and parameters, *Operations* contain the actionable actions and insights, and *Behaviors* contain the behavioral/simulation models that enable the reliable representation of PTs.

The second layer, which now introduces the concept of cooperativity, is where two individual DTs can share some relations to achieve a common goal, while still behaving semi-independently. These

relationships are handled as *Relationships* in the model, which are intended to be semantic object relationships between the two individual DTs. The idea behind this term is to be able to represent situations in cooperative systems, such as *I execute first, and then you*, which can be for example, represented as $Subject$—*executesAfter*—$Object$ or $Subject$—*isExecutedAfter*—$Object$, depending on the active/passive structure of the semantic relationships, which are equally valid.

Finally, in the third layer, an extension of the second layer is used to achieve the composition of DTs following this conceptual model. To do so, a particular semantic relationship, namely, *isComposedOf* is used to define DTs that are composed of smaller DTs objects. Unlike the open relationships available in the second layer, this layer only works on the *isComposedOf* relationship. Therefore, a triple $DigitalTwin_a$—*isComposedOf*—$DigitalTwin_b$ indicates that $DigitalTwin_b$ is a composite of $DigitalTwin_a$. This way, $DigitalTwin_a$ can be composed of multiple DTs. Moreover, the *isComposedOf* relationship is transitive, which means that composition can be achieved at several hierarchy levels. To exemplify this, if $DigitalTwin_a$—*isComposedOf*—$DigitalTwin_b$ and $DigitalTwin_b$—*isComposedOf*—$DigitalTwin_c$, the model can assert that $DigitalTwin_a$—*isComposedOf*—$DigitalTwin_c$.

To implement such a conceptual model, an application ontology is used. The ontology formalizes the definitions in the three-layer architectural model and provides the constraints and properties to increase the semantic richness of the model. Moreover, the ontological model, while using Protégé[1] and its automatic Java code generator[2], provides an alternative implementation for deploying DTs where it is possible to combine semantic reasoning and inference in a running application. Figure 5.3 illustrates the proposed ontology with its corresponding classes and object properties and its mapping to some of the components in the TwinManager architecture proposed in Section 4.3 as follows: *Attributes* are mapped to the *Twin.attributes* field, which is indicated by the yellow arrows in Figure 5.3. *Operations* are mapped to the *Twin.operations* field, which is indicated by the blue arrows in Figure 5.3. And *Behaviors* are mapped to the *Simulation* component, representing the behavioral models, which is indicated by the red arrows in Figure 5.3.

This modeling approach has been demonstrated with the Flex-cell case study Section 2.6.3, where the four independent assets composing the Flex-cell, namely, the two robotic arms and the two grippers, have been modeled as individual DTs. Since the modeling approach supports multiple levels of hierarchical composition, each robot-gripper pair has been composed into a robotic arm + gripper DT, which have then been composed into the Flex-cell DT. Additionally, some OWL and SWRL rules have been provided to the ontological model, which enables some reasoning capabilities in the model, such as a robotic arm itself cannot run any grasping operation unless it is attached to a gripper, i.e., the composition of a robotic arm and gripper is required to execute a grasping operation. These rules, combined with some querying mechanisms like SPARQL and SQWRL, can be used along with a decision-maker to create new sets of composed DTs during runtime, enabling the flexibility to have reconfigurable DTs.

[1] https://protege.stanford.edu/

[2] https://github.com/protegeproject/code-generation

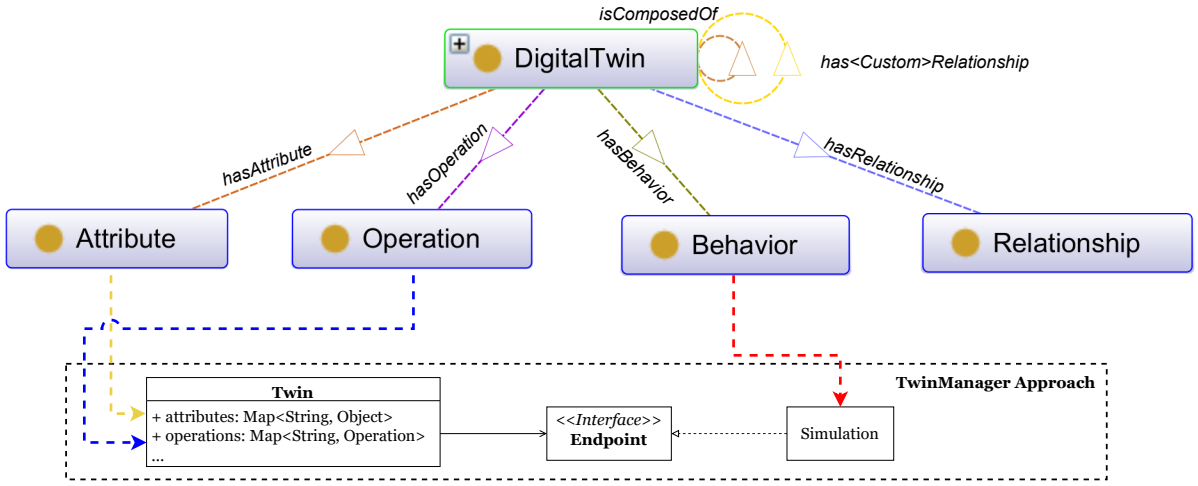

Figure 5.3: Ontology for DTs in cooperative systems with composition enabled mapping the components in Figure 4.2 presented in Section 4.3. *Yellow:* attributes mapping. *Blue:* operations mapping. *Red:* behavior-to-simulation mapping.

Contribution 4 (C4):

Demonstrated a modeling approach for DTs with composition enabled.

The qualitative results of having implemented this modeling approach with the Flex-cell case study show that this method provides a mechanism to incrementally build DTs for complex systems. In other words, the modular DTs can be composed, decomposed, and reconfigured on demand at different aggregation levels, and therefore, the reusability of components is increased and the difficulty in approaching complex systems is reduced. On the other hand, this approach only composes the structural aspects, and thus, its biggest limitation is regarding the composition of the behavioral aspects of internally coupled systems. Hence, even though there is a semantic relationship for the composition of behaviors of two or more DTs into a larger DT, this composition is uniquely logical and not functional, and therefore, this functionality needs to be provided externally.

At this point, simulation-driven DTs are yet limited to single-system or decoupled multi-system scenarios, which does still not address the challenge of the heterogeneous composition of DTs with coupled behavior.

5.3 An Architecture for Digital Twins for Systems with Coupled Behavior

With the aim of addressing the challenge of realizing composable DTs with coupled behavior, [P5] proposes an extension to the architecture presented in Section 4.3 for simulation-driven DTs (TwinManager) adopting the co-simulation principles and following the principles of the modeling approach with composition enabled presented in Section 5.2 to functionally realize DTs with coupled behavior.

To do so, this architecture introduces a new concept, turned into a class, named `TwinSystem`, which is the functional implementation of a composition of several DTs into a larger DT. However, this implementation, unlike the modeling approach in Section 5.2, only supports a one-level hierarchical composition, which is a limitation that can easily be solved by defining the hierarchy-highest DT in the model as the instance of the `TwinSystem` class.

Such a class contains a field that lists the internal twins and a new configuration file, the *Twin System Configuration*, which describes the input/output relationships between twins inside the twin system. These relationships can be, for example, specified as part of the *Relationships* component of the ontological model (see Figure 5.3).

Similar to the `Twin` class, the `TwinSystem` class also follows a Façade pattern and is connected to an `Endpoint` interface. This endpoint, unlike the `FMIEndpoint` mentioned in Section 4.4, needs to support composed systems with intrinsically coupled behavior. For this, a new abstraction for endpoints is introduced. The `Endpoint` interface is extended by two children interfaces, namely, the `AggregateEndpoint` and the `IndividualEndpoint`. The `IndividualEndpoint` interface works as the `Endpoint` interface in its previous version (see Section 4.3), i.e., it basically provides access to individual twins and/or simulators. The `AggregateEndpoint` interface, on the other hand, provides the methods to add endpoint specializations for twin systems. A new endpoint specialization of the `AggregateEndpoint` interface, the `MaestroEndpoint`, is also introduced to run co-simulation scenarios using the co-simulation engine Maestro as a slave. This way, the previously defined behavioral models of the individual twins (*Behaviors* in the ontological model), which were wrapped as FMUs, can still be reused at the DT system level. More precisely, the composite DTs can be bound to `FMIEndpoint` interfaces, and their composed system, to a `MaestroEndpoint`, enabling the functionality both at the individual and system levels. Currently, instances of the `TwinSystem` class can only be attached to `MaestroEndpoint` specializations; however, other co-simulation engines can also be adapted through new `Endpoint` specializations. The architecture for the TwinManager with extended capabilities to realize DT systems with coupled behavior is shown in Figure 5.4.

Contribution 5 (C5):

Proposed an architecture that enables the functional implementation of composed DTs with coupled behavior.

The design of this architecture includes two additional requirements, as follows:

Requirement 5: Support for the composition of DTs Given the fact a DT system can be composed of smaller composites, the architecture should be able to provide the interfaces to functionally compose DTs into a larger system. This is possible via the class `TwinSystem` and method `createTwinSystem`, which requires a list of twins that compose the twin system. Additionally, the TwinManager methods `get/setAttributeValue` and `executeOperation` are extended at the system level as `get/setSystemAttributeValue` and `executeOperationOnSystem`.

Requirement 6: Support for co-simulation In case the simulation of a composed DT system includes internal coupling, an object of the `TwinSystem` class, representing a composed DT, should be able to read and write information from/to its heterogeneous co-simulation setting. This is achieved by the `MaestroEndpoint`, which creates an interface between the TwinManager and Maestro to read from, write to, and execute co-simulation experiments.

It is worth mentioning that the new architecture for the TwinManager keeps the consistency and base components of its previous version, and therefore, single-system and (de-)coupled multi-system simulations can be run under the same application. It is also worth pointing out that a DT that has

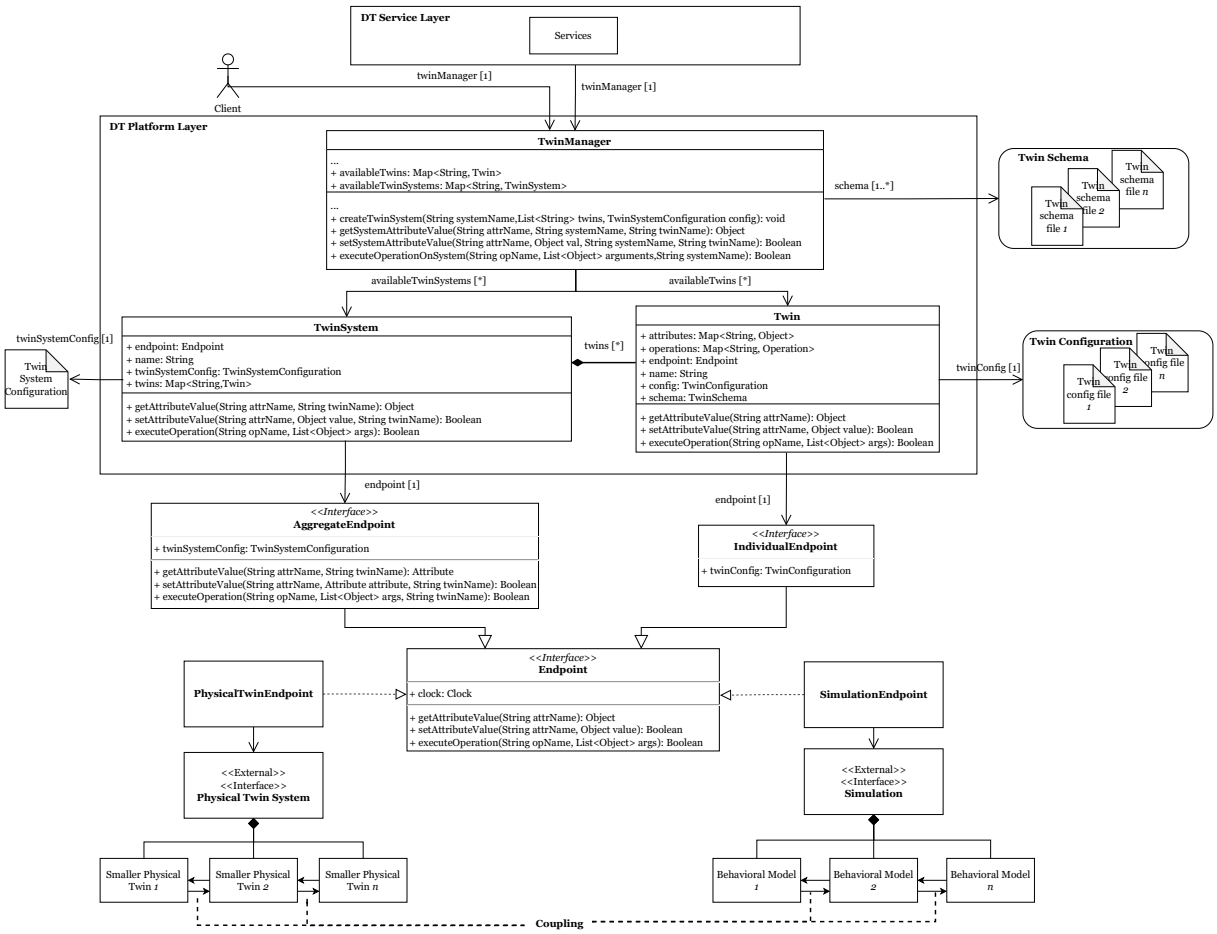

Figure 5.4: Architecture for DT systems with coupled behavior using the TwinManager approach.

been built to work at the single-system level can be reused to work at the system level, increasing the modularity of existing DT implementations.

This architecture has been demonstrated using a multi-case study research setting, namely, with the Three-Tank System (see Section 2.6.1) and the Flex-cell (see Section 2.6.3). The implementation in these two settings has shown that this approach is sufficient to overcome the challenges concerning the heterogeneous composition of DT with coupled behavior due to system coupling and system synchronization (as in cooperative/collaborative settings). Additionally, the combination of efforts in modeling approach and architecture, has also been proven to increase the reusability of assets and its adaptation to a different execution environment, in this case, the DTaaS platform proposed by Talasila et al. [CP10]. The examples for the Three-Tank System and Flex-cell case studies using the DTaaS platform are available on GitHub[3,4].

Using the multi-case study setting, the approach is also evaluated in terms of the reusability of its components and reduction of implementation effort. To conduct such an evaluation, the components required to instantiate both case studies, namely, the Three-Tank System and the Flex-cell are compared to analyze how many of the components can be reused or replaced, and how many need to be added or deleted. In particular for this multi-case study setting, based on the instantiation of the Three-Tank System DT, 29.2% of the components used can be reused and 29.2% can be replaced with

[3]https://github.com/INTO-CPS-Association/DTaaS-examples/tree/main/digital_twins/three-tank
[4]https://github.com/INTO-CPS-Association/DTaaS-examples/tree/main/digital_twins/flex-cell

slight parameterization in order to instantiate the Flex-cell DT, which means that there is a saving in engineering effort of 58.4%. This evaluation process is further explained in Section 5.2 of [P5].

5.4 Bridging Digital Twins and Co-Simulation

Similar to Section 4.4, the `MaestroEndpoint` specialization is one of the major findings of [P5]. This endpoint extends the capabilities of the `FMIEndpoint` specialization by providing an interface to access a full co-simulation setting rather than just decoupled behavioral models stored as FMUs. This way, the co-simulation, and so the behavioral models, can have the internal coupling, which was a limitation of [P2] and a functional implementation of [P3]. As a result, the TwinManager is provided with the versatility to run simulations of individual behavioral models with the `FMIEndpoint` and to run simulations of composed behavioral models with coupled behavior with the `MaestroEndpoint`, all under the same application.

This endpoint specialization uses Maestro the co-simulation engine as a slave to run co-simulation settings. It is therefore limited to the capabilities Maestro offers, though it provides good insights into the potential of combining co-simulation and data-model-driven DTs, which bridges the architectural differences between the previously mentioned *Co-simulation and model-based DT framework* and *Structured data DT framework* categories (see Section 3.2).

The `MaestroEndpoint` specialization maps the operations available in Maestro to run co-simulation experiments and the mechanisms to handle data for reading and writing, bridging the interface for DTs and co-simulation. More precisely, the method `executeOperation` is mapped to Maestro's operations *import*, which converts the JSON-based configuration representing the *Twin System Configuration* in Figure 5.4 to a *MABL* specification, which is the language Maestro understands; and *interpret*, which executes the co-simulation given the settings. For data interfacing, there are two setups with Maestro. The first setup uniquely considers Maestro and the second setup considers Maestro and RMQFMU. For the former setup, the interface adapter uses file reading from CSV files (where Maestro saves outputs) and file writing to JSON files (where the *Twin System Configuration* is stored and the parameters for co-simulation are defined). For the latter setup, RMQFMU enables a publish/subscribe channel to interact with the co-simulation in real-time via a RabbitMQ Broker. Therefore, both reading and writing can be performed over RabbitMQ by consuming and publishing to a particular channel given by the RMQFMU. The overview of the mapping of the interface of the `MaestroEndpoint` to the two setups is shown in Figure 5.5.

For the definition of the co-simulation settings, the configuration must follow Maestro's requirements. However, step size, co-simulation time, and parameters can be directly from the `MaestroEndpoint`. The current prototype implementation for the first setup is completely working, while the second setup works by providing external management to the RMQFMU communication, which is a feature under development to be incorporated into the `MaestroEndpoint`.

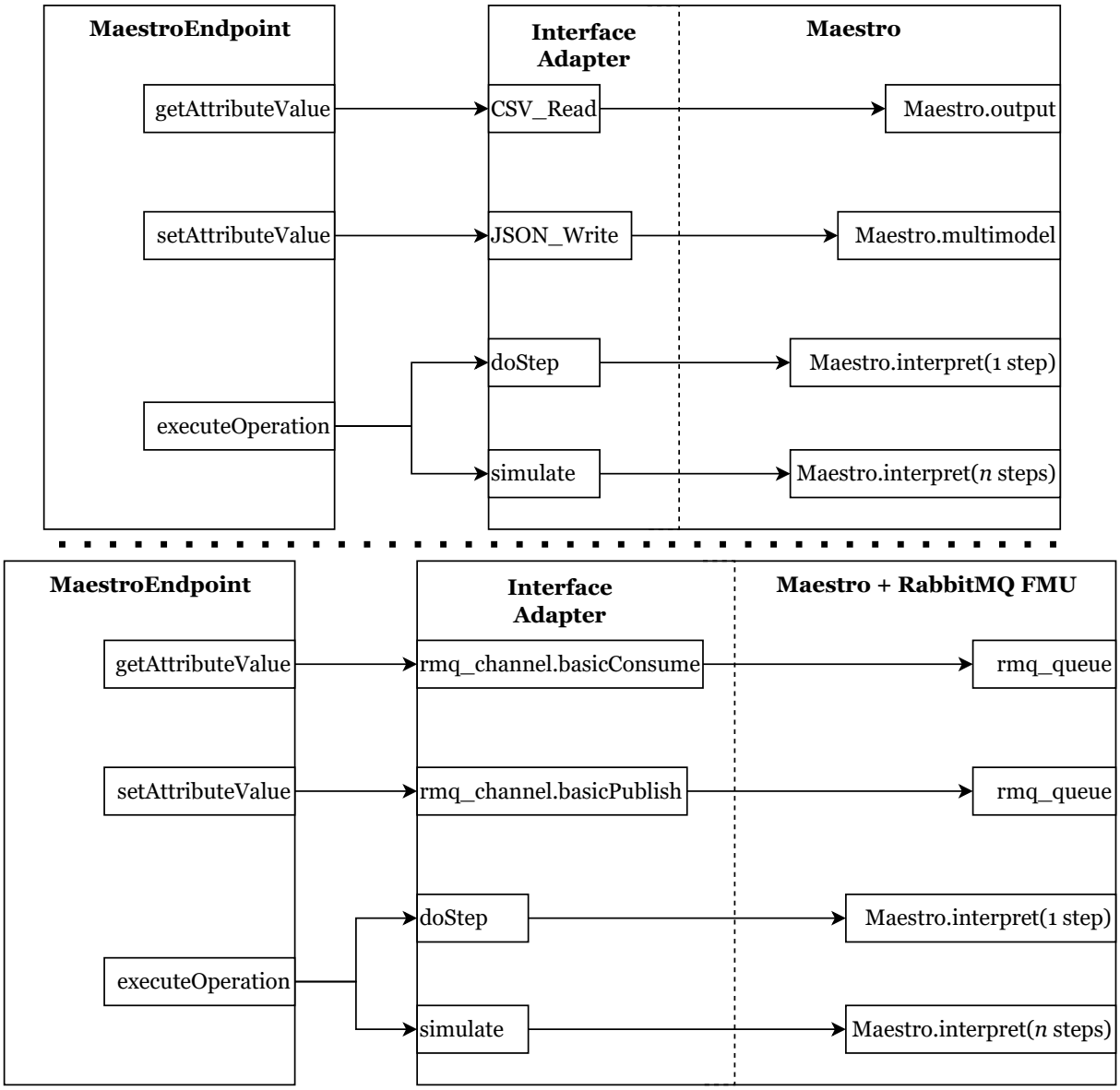

Figure 5.5: Overview of the MaestroEndpoint interface and its mapping to Maestro (above) and Maestro + RMQFMU (below).

6 Digital Twins in Robotics

DTs are intended to provide value to a particular business goal [27], which are usually engineered for particular case studies, but in a wide range of domains, including manufacturing, power systems, and robotics, among others. DTs have gained interest in the robotics domain as a promising technology that provides persistence, insights, and explainability [81]. Moreover, robots are also widely used in industry, which fits the particular interests of this PhD project in binding research and industry.

This chapter is based on the research output from publications [P3], [P4], and [P7]. The research method used in this chapter is applied, where the efforts are put into the applicability and usability of DTs and the approaches provided throughout this PhD thesis in the robotics domain in a case study research setting.

6.1 Overview

Figure 6.1 presents an overview of the features of this chapter. This chapter presents the results of the applied research in robotics and two contributions of the PhD project associated with Publications [P4, P7] (see Figure 1.4), **C6**, which is related to the integration of the skill-based engineering concept with DTs, and **C7**, which is related to coupling to RoboSim to generate model-based co-simulation setups for realizing DTs in robotics.

Section 6.2 takes the modeling approach presented in Section 5.2 as an input to provide an extension related to **C6**, so robot skills are considered under the modeling approach.

Finally, Section 6.3 extends the capabilities of RoboSim, a modeling framework for robotics (see Section 2.5), to propose a methodological approach related to **C7**, which provides the steps to create model-based co-simulations to realize DTs for robotics.

6.2 Modeling Extension for Digital Twins in the Robotics Domain

In Section 5.2, a modeling approach for composed DTs has been proposed and demonstrated in the Flex-cell case study (see Section 2.6.3). [P4] proposes an extension for such a modeling approach to include a better definition of the term *Operation* by disaggregating operations based on their hierarchical abstraction. To do so, the *skill-based engineering* concept [82], which divides operations into three abstraction levels, namely, *Device Primitives*, *Skills*, and *Tasks*, is adopted.

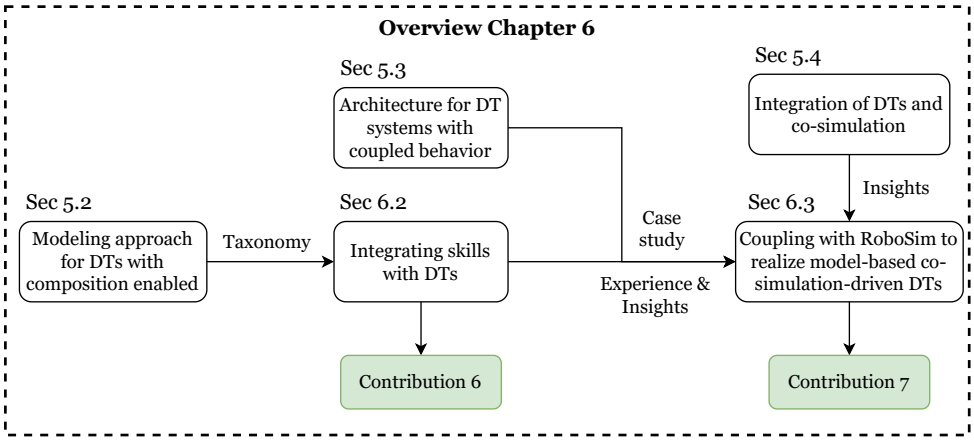

Figure 6.1: Overview of Chapter 6 with the features and sections that lead to contributions.

With the skill-based approach, *Tasks* are divided into sequences of *Skills*, and *Skills* into sequences of *Device Primitives*. This way, operations are represented in a more meaningful way, which additionally, increases the reusability of high-level operations. These operations can be reused within the same case study, e.g., to reconfigure a routine based on new inputs dynamically, or in different case studies that share architectural similarities, i.e., a *Screwdrive* operation will be valid in any scenario where there is a screwdriver, which would have a similar parameterization. Integrating the skill-based concept with an approach that already enables composition also enables the creation of combined skills, i.e., skills that require primitive operations from two or more individuals, e.g., a *grasp* skill requires the composition of a robotic arm and a gripper to work. Therefore, integrating skills and composition is highly convenient to approach flexible systems, in this case, in cooperative robotics, with a more meaningful definition of operations and extended constraints.

The extension is done on the ontological model presented in Figure 5.3. The ontology is provided with three additional classes, namely, *DevicePrimitive*, *Skill*, and *Task*, which are subclasses of the class *Operation*. A new property is added, namely, *isCombinationOf*, which associates the combination of device primitives into skills, and skills into tasks. Figure 6.2 shows the extended ontology for skills-enabled DTs.

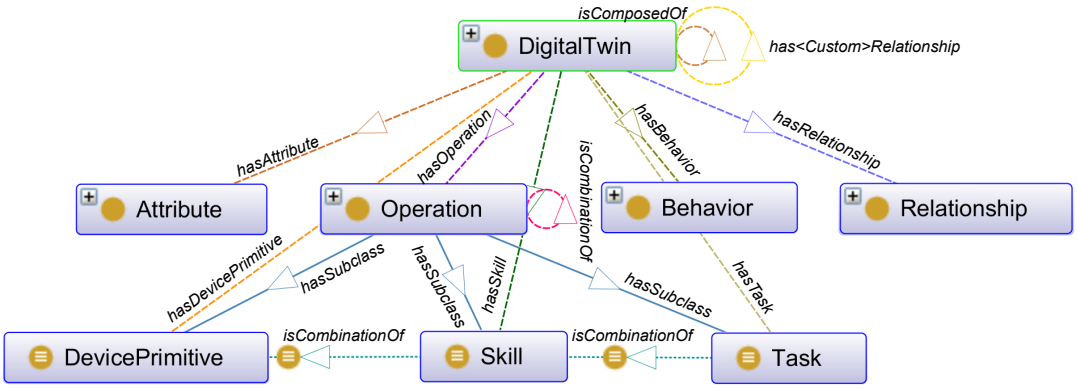

Figure 6.2: Extended ontology for DTs in cooperative systems with skills enabled.

Contribution 6 (C6):

Created a mechanism to integrate DTs and the skill-based approach for applications in robotics.

This extended approach with skills has been demonstrated in the Flex-cell case study Section 2.6.3, but now, with a more meaningful and structured definition for two experiments to be executed, following the skill-based concept, as shown in Figure 6.3. These two experiments represent two different shapes on the 2D grid of the Flex-cell, which can be executed via the skills *GraspObject* and *PlaceObject*, which include the device primitives *Moving*, *Gripping*, *ComputePosition*, *ComputeRotation*, and *ComputeFeasibility*.

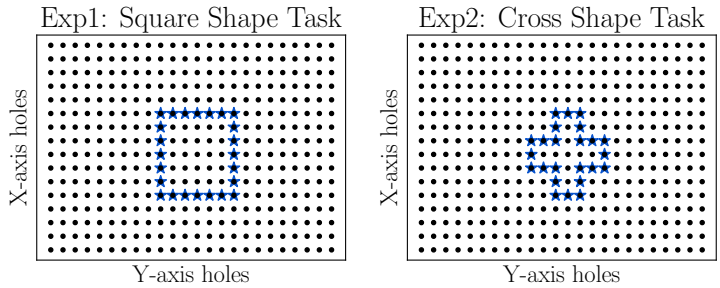

Figure 6.3: Experiments performed with the skill-based engineering approach.

After conducting the experiments on the Flex-cell case study, it has been demonstrated that this approach provides an improvement in terms of implementation effort of 81.2% and in terms of reusability of 80.3% from a robotics software programming perspective. The evaluation settings and results are described in more detail in Section 4 of [P4].

6.3 Coupling with the RoboSim Modeling Framework

In Sections 5.3 and 5.4, co-simulation was used to bridge the gap for composing heterogeneous DT systems with coupled behavior. Moreover, it provided insights into co-simulation as a key technology to realize such complex systems, which was showcased with the Flex-cell case study (see Section 2.6.3) in a cooperative robotics setting. [P7] proposes an improved approach to be adopted for co-simulation-driven DTs in the robotics domain. To do so, RoboSim (see Section 2.5), a modeling framework for verified robotic simulations, is combined with the FMI interface to achieve co-simulation settings. Such settings, inheriting the capabilities of RoboSim, provide a suitable mechanism to realize DT-enabled robotic systems. The overview of this approach is composed of three phases, namely, the modeling, implementation, and co-simulation phases, as shown in Figure 6.4.

RoboSim is currently capable of automatically generating model-based software for the state machine models of robotic controllers (d-model) as C code and for the robots' rigid body dynamic properties (p-model) as SDF files, which is represented in Figure 6.4 by the green arrows and blocks inside the green dotted box. However, there is yet lack of (i) automatically generating software for the model that maps the interactions between the state machine and the robot (Platform Mapping model) and (ii) enabling a (simulated) robot endpoint to be used in the co-simulation, which is represented by the blue arrows and blocks in Figure 6.4. As the idea is to achieve model-based co-simulations settings starting from RoboSim models, this approach addresses the software implementation for the Platform Mapping model, and the integration with the FMI interface for these components, so they can be managed as FMUs, which is represented by the orange arrows and blocks in Figure 6.4. In addition, since such a process involves several steps, proper guidance is needed. Therefore, a methodological approach to achieve such co-simulation settings for their subsequent use in DTs for robotics is proposed.

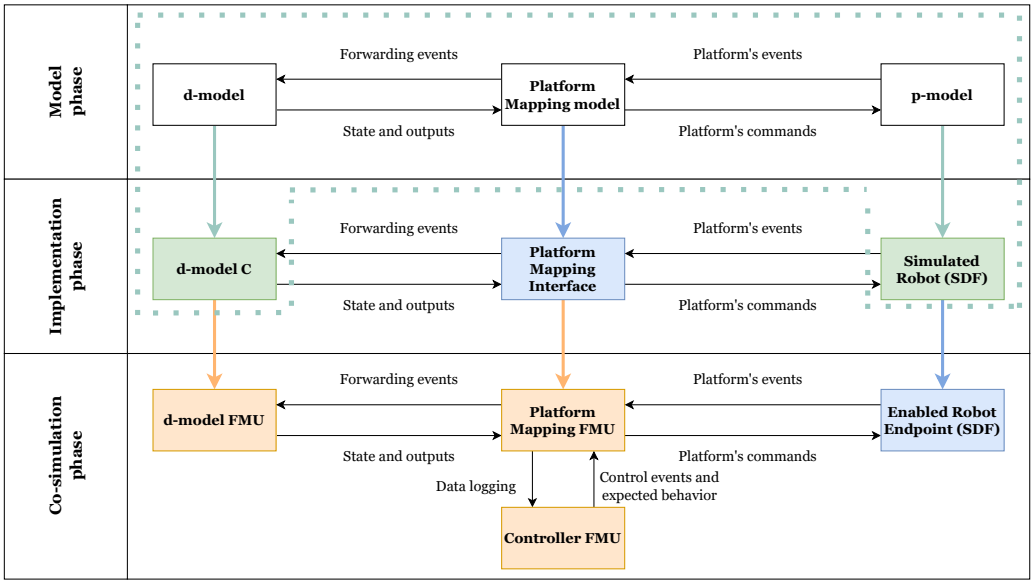

Figure 6.4: Overview of the phases to achieve co-simulation settings with RoboSim. The green dotted box delimits the current capabilities of RoboSim.

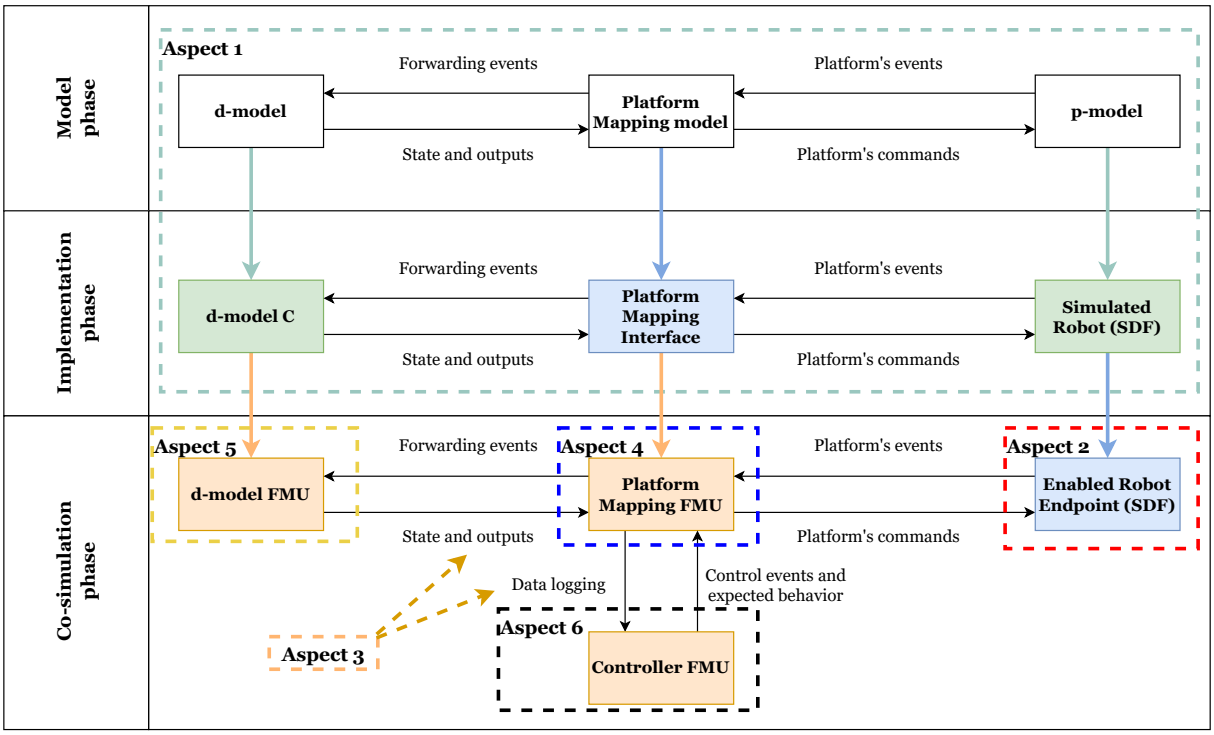

Figure 6.5: Mapping of the aspects of the methodology to the three-phase overview architecture of the integration with RoboSim presented in figure 6.4.

Such methodology is divided into 14 steps in six major aspects, as shown in Figure 6.5. A more detailed description of the methodology steps is available in Section 5.2 of [P7]. The first aspect, represented by the green dashed box in figure 6.5, is about the modeling and implementation phases in RoboSim, including the implementation of the Platform Mapping model, hereafter called *Platform Mapping Interface*. The second aspect, represented by the red dashed box in figure 6.5, is related to enabling the robot-specific endpoint. The third aspect, represented by the orange dashed box in figure 6.5 pointing to the inputs and outputs of the components in the *Co-simulation phase*, is about designing the

co-simulation setting, including the definition of input and output ports for the FMUs. The fourth and fifth aspects, represented by the blue and yellow dashed boxes respectively in figure 6.5, are associated with wrapping the Platform Mapping Interface and the d-model C code into FMUs. The sixth and last aspect, represented by the black dashed box in figure 6.5, is about adapting the robot routines to a *Controller FMU*, which is intended to convey the expected behavior to the co-simulation setting.

Once the FMI-enabled components have been enabled, the co-simulation can orchestrated by a co-simulation engine. The co-simulation engine Maestro, which was used in Section 5.4, is selected to orchestrate the co-simulation with RoboSim-based components.

Contribution 7 (C7):

Coupled with the RoboSim modeling framework to generate model-based co-simulations for DTs in robotics.

For the demonstration of this approach, the UR5e robotic arm belonging to the Flex-cell case study is selected. The UR5e is modeled with RoboSim to obtain its p-model based on an approximation made by cylinders and its d-model based on a state machine that can operate a gripper-enabled robotic arm with discretized move commands (see Section 4.1 of [P7] for more information). Similarly, its Platform Mapping model captures the relationships between the state machine that operates with discrete moving commands and the joint-based properties of the robot. This Platform Mapping model and logic behind are then instantiated into the Platform Mapping Interface using the model as a basis and the simulation engine CoppeliaSim[1] to enable the remote simulated endpoint. The endpoint is enabled by a scene that uses joint-based position control and includes the SDF file automatically generated from the UR5e p-model. The Platform Mapping Interface is instantiated using the CoppeliaSim Remote API[2] as the endpoint-specific interface.

The generated co-simulation settings can be used as comparable virtual representations of any robotic platform, which can be used to leverage DTs in the robotics domain. Additionally, such settings can be coupled to the TwinManager architecture (see Section 5.3) through the MaestroEndpoint specialization (see Section 5.4), which further enables the comparison and administration of DTs developed with this approach under the same operating environment. The co-simulation results of orchestrating the components in the Co-simulation phase of Figure 6.5 for the UR5e belonging to the Flex-cell case study, shown in Figure 6.6, demonstrate the capabilities of this approach to provide a suitable mechanism to realize DTs for robotics using RoboSim and co-simulation, while inheriting the capabilities of these two technologies. DTs realized following this approach can keep track of a combination of continuous-time and discrete-event data, such as commands, command arguments, state, and positions as shown in the plots of Figure 6.6. Other variables, such as velocity, torque, and multiple types of events can also be obtained from such an implementation, which further enhances the observability of the DT in effect.

[1] https://www.coppeliarobotics.com/
[2] https://manual.coppeliarobotics.com/en/remoteApiOverview.htm

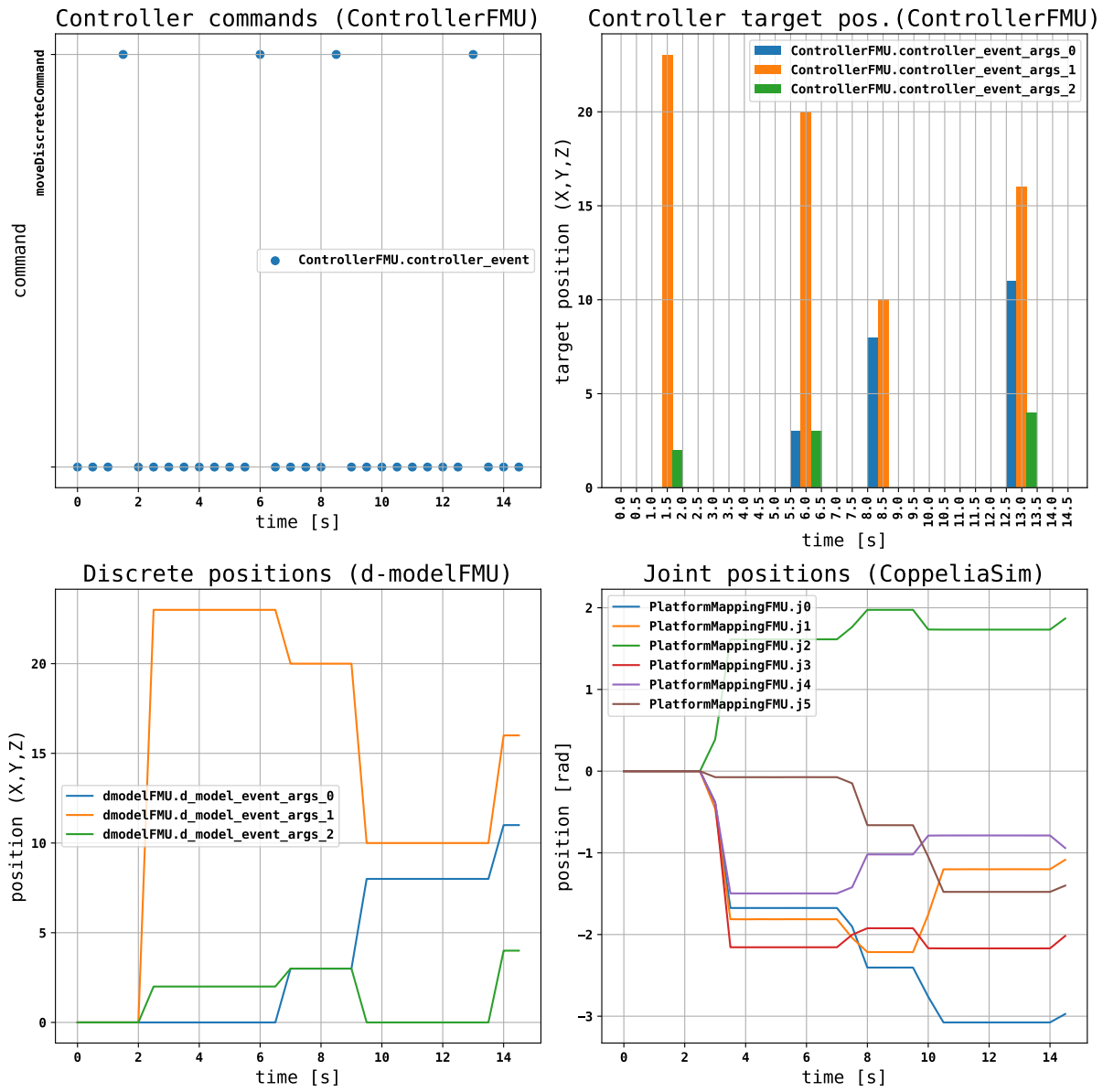

Figure 6.6: Co-simulation results for the UR5e case study using the methodological approach with RoboSim.

7 Concluding Remarks

This chapter concludes this PhD thesis by discussing the most relevant aspects and the answers for the RQs stated in Section 1.3, analyzing the overall impact of the outcomes, assessing the contributions presented in Section 1.2 with the proposed criteria presented in Section 1.5, and providing potential research directions to work upon the research foundations built throughout this PhD project.

7.1 Discussion

This thesis has taken an angle to approach DTs mainly from the simulation perspective. Although the categorization of DTs, DSs, and DMs by Kritzinger [38] poses a clear distinction between the communication capabilities of the three categories, it is still uncertain what the internal components and services of DTs are. By integrating simulation, and in general, behavioral models, DTs are expected to be reliable representations of their PTs, habilitating the way to incorporate value-added services, such as optimization, monitoring, and visualization, among others. Ideally, such services could be attached straightforwardly to any DT, if the methods and the structure of services and DTs are sufficiently generalizable.

The use of DT technology in realistic settings is another aspect to discuss. DTs are expected to provide value to businesses, i.e., be additional. Currently, DTs are developed in a case-specific manner, which challenges the transfer to or reuse of components in other case studies, which also means that realizing DTs can be an expensive task. For the settlement of DT technology in the market, these are expected to provide more value than what they require to be realized. Therefore, the methods to do so should ensure a low implementation effort, high reusability of components, and high generalizability to various case studies. These properties, in addition to being suitable for any hardware and software artifacts, provide the flexibility to approach large and complex systems, which approximate more to realistic settings, such as HMLVs systems.

These aspects, represented by the proposed criteria for the assessment of the contributions of this PhD project (see Section 1.5), have been addressed via the methods proposed throughout, which are assessed later in Section 7.3. Such criteria are also aligned with the RQs stated in Section 1.3. In order to respond to such RQs, each RQ is taken individually and analyzed with respect to its nature, i.e., as exploratory or applied, and in accordance with Figure 1.1. Exploratory research is intended to provide

the foundations to develop hypotheses and gain new insights through exploration [29]. On the other hand, applied research aims at finding a solution for an immediate problem, and therefore, it can be responded to more objectively [29].

Responding to **RQ1**, some of the exploratory findings of this RQ come from **C1**, which are presented in Section 3.3, where the main gaps and good practices regarding the theory-to-practice transition of DT technology are listed.

Responding to **RQ2**, the output from **RQ1** is taken to propose specific solutions to support the integration of (coupled) behavioral models. This RQ is addressed in two phases. First and leading to **C3**, regarding the integration of individual behavioral models using the approach presented in Section 4.3 and its capabilities to bridge DT platforms and black-box simulation. Second and leading to **C5**, by extending the capability **C3** to support co-simulation for enabling the integration of coupled behavioral models with DT platforms, which is presented in Section 5.3.

Responding to **RQ3**, the exploration effort focused on the use of the composition of DTs and multiplicities, which was one of the exploratory findings from **RQ1**, as a mechanism that increases the reusability of DT components, which enables the realization of incremental applications by aggregating smaller DT composites. Some of the tangible findings of this RQ are presented in Sections 5.2 and 5.3, where composition is used to approach DT systems and the internal composites can be reconfigured or reused according to the business goals, which leads to **C4**.

Responding to **RQ4**, which is focused on applied research in robotics, two methods integrating robot-specific approaches and DTs were proposed in Sections 6.2 and 6.3. The first method, which leads to **C6**, incorporates the skill-based engineering concept, which has been proven to improve reusability and reduce the implementation effort of robot programming. The second method, which leads to **C7**, adopts the modeling framework RoboSim, and thus, it reduces the implementation effort by automatically generating portions of code and providing the methodology to realize DTs for robotics.

Responding to **RQ5**, the output from **RQ1** is taken to address the lack of standardization and consensus for reporting case study research in the DT field. To do so, Section 3.5 presents a systematic reporting framework to report comparable and consistent case studies in the DT field using three reference frameworks as a basis, which leads to **C2**.

These RQs lead to testing the hypothesis stated in Section 1.3, and recalling: *A generalizable method for DTs facilitates the realization of complex CPSs where flexibility and reusability are required.* This hypothesis states that generalizable methods, i.e., case-independent methods, facilitate the DT engineering of complex CPSs. Additionally, these methods are suitable for settings where flexibility and reusability are required. In this regard, the methods proposed throughout this PhD thesis are case-independent, have been theoretically validated to be generalizable using multi-case study research [23, 28], and have been proven to either reduce the implementation effort or to be easily adaptable to new settings. The reduced implementation effort has been validated by reusing components of composable systems, which enables approaching complex systems more incrementally and responds to the need for reusability. In addition, the adaption to new settings can be, for example, adapting the same case study to different business outcomes or switching to different case studies under architectural similarity, which responds to the need for flexibility.

On the other hand, there are some limitations which may threaten the validity of the hypothesis, as follows:

- The claim for generalizability has been inferred from using multi-case study research in a few samples, which may limit the validity for generalization to case studies in the same or similar domain with architectural similarity.
- The claim for facilitating the DT engineering of complex CPSs has been inferred from executing the methods in systems with reduced scale. This may threaten the validity of generalization in more realistic systems of larger scales.
- The claim for suitability in flexible settings has been inferred from executing the methods in changing environments where the business logic is not totally under control and several assumptions in various aspects, such as plans (routines), services, models, and communication, have been considered.

With these tangible results and limitations, we can confirm the hypothesis that generalizable methods for DTs facilitate the DT engineering process in settings where flexibility and reusability are required. Additional work is though required to further ensure the generalizability of the methods to more realistic systems of larger scales.

7.2 Impact

The overall impacts of this PhD thesis can be described as follows:

Easier integration of simulation. This PhD project has addressed a current challenge in integrating simulation with DTs in an easy-to-deploy manner. This includes the simulation for single systems or coupled systems.

Easier adoption for SMEs. This PhD project has provided generalizable methods to incrementally build DTs, which include conceptual modeling, architectural design, functional implementation, and reporting. This impact intends to assist smaller stakeholders, such as SMEs, to build their on DT applications in a *do-it-yourself* manner.

Approaching complex systems. Further elaborating on the two previous impacts, this PhD project has focused on providing the methods to approach DTs for complex systems with heterogeneous data sources and coupled behavior, more precisely, by composing smaller composites into larger DTs, enabling a more incremental way to approach the realization of DTs for such complex systems.

Bridging domains and methods. This PhD project has bridged gaps between domains and methods, namely, model-based engineering and black-box simulation, ontological engineering and Digital Twinning, DTs and co-simulation, and robot-specific modeling methods and co-simulation, to the same end, which is the easier realization of case-independent DTs.

7.3 Assessment of Contributions

Figure 7.1 and Figure 7.2 illustrate the assessment of contributions based on the proposed criteria (see Section 1.5), the former at the individual level and the latter at the combined level. This process is conducted from the author's perspective, which may introduce bias in the assessment. As a remark,

contributions **C1**, **C2**, and **C7** are assessed individually, while **C3** and **C5**, and **C4** and **C6**, are assessed as groups. This is done since **C5** provides an extension to **C3**, and similarly, **C6** provides an extension to **C4**. The groups are named **C5'** and **C6'**.

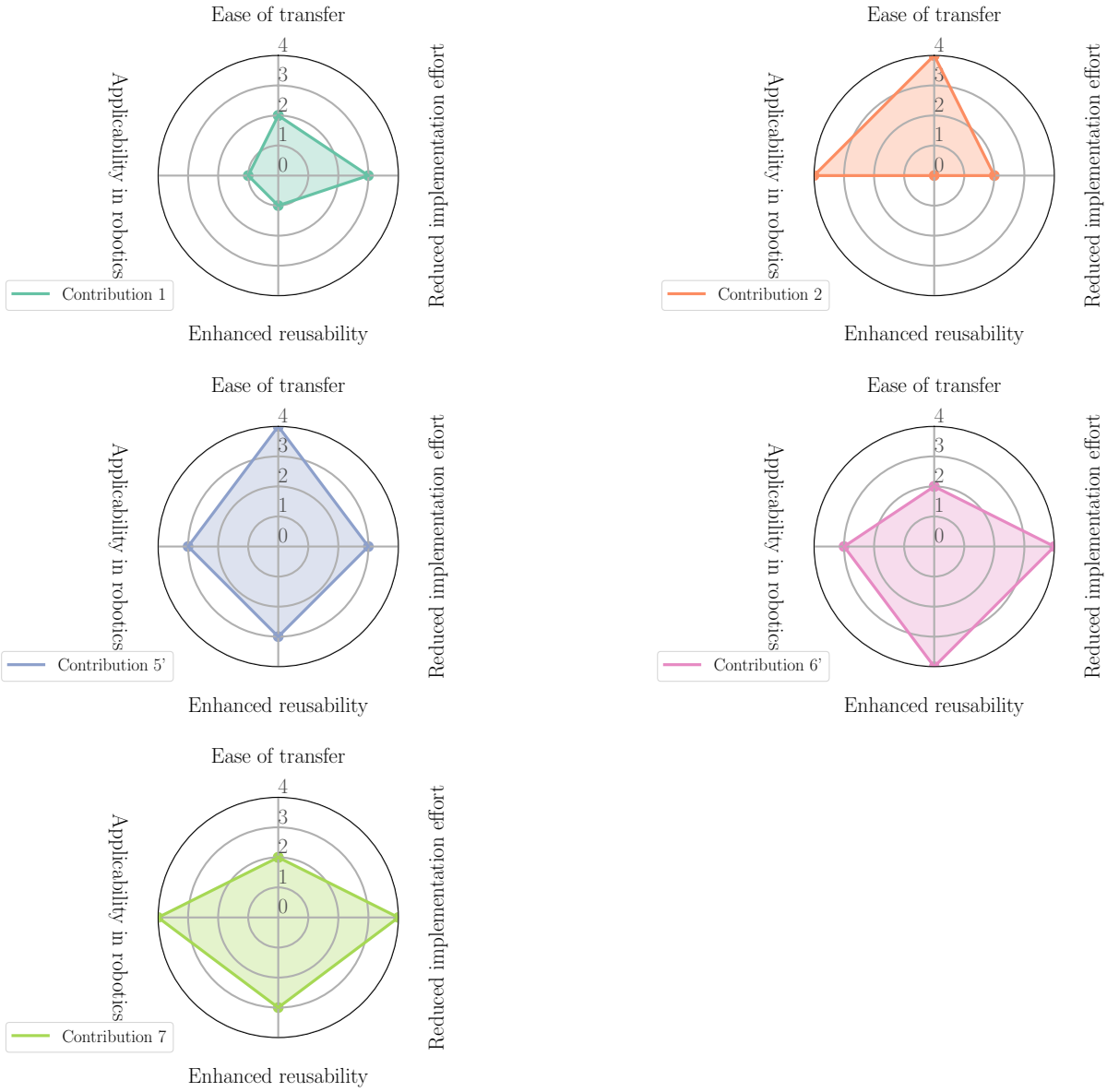

Figure 7.1: Assessment of contributions based on the proposed criteria.

7.3.1 Assessment of C1

Ease of transfer. Assessed (2/4). This contribution intends to find and analyze frameworks that enable the transfer of already developed DTs or methods for DTs to other case studies and different domains, and their suitability for large-scale or industrial deployments. However, the contribution is limited to open-source frameworks, which limits the spectrum of tools that can be used for more realistic deployments in industrial or field settings.

Reduced implementation effort. Assessed (3/4). The main aim of this contribution is to survey frameworks that assist practitioners in the realization of DTs with a set of guidelines and existing infrastructure instead of realizing DTs from scratch. However, this contribution only surveys the frameworks and not the methods to reduce implementation effort.

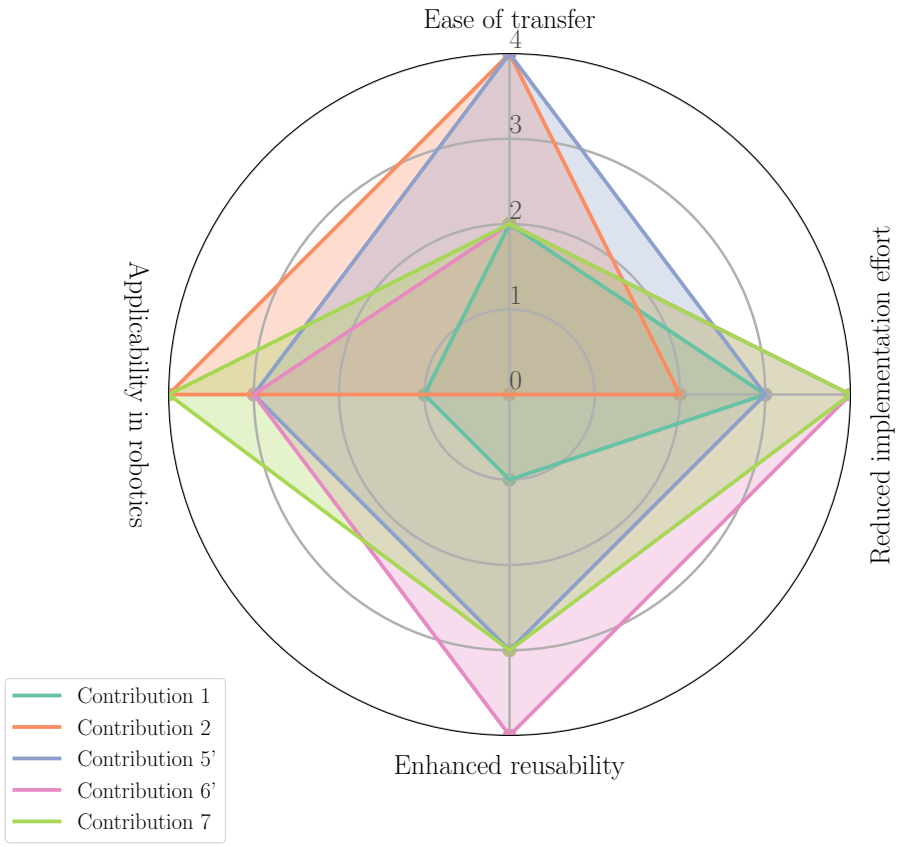

Figure 7.2: Combined overview of the assessment.

Enhanced reusability. Assessed (1/4). This contribution elaborates on the use of composition and frameworks that allow composition as a mechanism to increase the reusability of existing DT assets. However, it does not elaborate explicitly on methods to increase reusability.

Applicability in robotics. Assessed (1/4). This contribution provides a domain-independent view of frameworks for the realization of DTs, which can also be used in the robotics domain. However, it does not survey any particular framework or characteristics that are specific to the realization of DTs for robotics.

7.3.2 Assessment of C2

Ease of transfer. Assessed (4/4). The rationale of this contribution is to provide a guideline to report DT case studies for a better understanding of practitioners in other domains and settings. It also provides the reproducibility and considerations required to transfer a case study from one setting, e.g., a lab setting, to another setting, e.g., a field setting.

Reduced implementation effort. Assessed (2/4). Although this contribution does not explicitly provide methods for the reduced implementation effort, it contributes to the documentation and reporting effort of DT case study research as well as the lessons learned in Digital Twinning and reporting.

Enhanced reusability. Assessed (0/4). This contribution does not elaborate on the reusability of DT assets.

Applicability in robotics. Assessed (4/4). This contribution, which has been showcased in a cooperative robotics setting (see section 4 of [P6]), has been proven to provide a systematic method to report DT case studies in robotics.

7.3.3 Assessment of C5'

Ease of transfer. Assessed (4/4). These contributions have been specifically evaluated in relation to the adaptation effort of DT setups to work with different case studies by replacing modules and models.

Reduced implementation effort. Assessed (3/4). These contributions have also been specifically evaluated in terms of implementation effort, showing that they further reduce the implementation effort of simulation-driven DTs compared to implementing the same DTs from scratch or using existing DT platforms. However, it requires having the models and the engineering knowledge to replace the modules.

Enhanced reusability. Assessed (3/4). Although **C3** does not focus on the reusability of DT assets to a great extent, it elaborates on how different DT services can be reused with several DT instances by following the architecture's structure and the twin schemas. **C5** extends the capabilities of **C3** to include composition, and therefore, enhances the reusability of DT assets. However, **C5** is still limited to a one-level hierarchical aggregation at the implementation level.

Applicability in robotics. Assessed (3/4). **C5** focuses on the application of coupled systems, such as in cooperative or collaborative robotics. **C5** has also been showcased to work on a cooperative robotics case study.

7.3.4 Assessment of C6'

Ease of transfer. Assessed (2/4). These contributions provide a domain- and case-independent conceptual representation for DTs. This fact enables the transfer to other case studies or settings. However, the approach, due to being abstract, may not include special considerations required in other more realistic systems, e.g., for field or industrial deployment.

Reduced implementation effort. Assessed (4/4). These contributions, by adopting composition, ensure some reduction in the implementation effort when implementing complex systems. It also supports automatic code generation based on the model, which can further reduce the implementation effort. Moreover, **C6** has been specifically proven to reduce the software implementation effort of robot routines due to using the skills-enabled approach in comparison to manually create the same routines.

Enhanced reusability. Assessed (4/4). The rationale for these contributions is to enable the composition of DTs, and therefore, to enable the reusability of DT composites and their internal components in different setups and case studies.

Applicability in robotics. Assessed (3/4). **C4** does not provide any particular considerations for its application in robotics, though it has been demonstrated to work on a robotics case study with certain limitations. **C6** extends the capabilities of **C4** by integrating a robot-specific approach to address a more complete robot programming paradigm, which also enables the composition of

operations. However, both contributions are limited concerning their lack of support for coupled behavior.

7.3.5 Assessment of C7

Ease of transfer. Assessed (2/4). **C7** provides a methodological approach that assists with the ease of transfer to other case studies; however, it is limited to the robotics domain.

Reduced implementation effort. Assessed (4/4). **C7** takes advantage of existing tools, namely, RoboSim, Maestro, and RMQFMU, to reuse the implementation effort of co-simulation setups for robotics. Additionally, RoboSim's capabilities regarding automatic code generation and reusability of d-models and platform mapping interfaces further contribute to the reduced implementation effort.

Enhanced reusability. Assessed (3/4). **C7** elaborates on how d-models and platform mapping interfaces can be reused or instantiated for different endpoints and then reused. However, the reusability is limited to applications in the robotics domain and the (re)use of the modules requires a certain level of knowledge.

Applicability in robotics. Assessed (4/4). **C7** has focused on applied research in robotics. It provides a robotics-specific methodological approach to generate model-based co-simulations, which can be used to realize DTs for different robotic platforms while ensuring safety and trustworthiness in the software behind the CPS.

7.4 Potential Research Directions

Although this thesis has covered a wide spectrum of current research challenges related to realizing DTs and their application in robotics, there are still several areas that require further exploration and research. Some of the potential areas envisioned to be covered in future work include:

Generalizable services for DTs. This area certainly requires more research, so that DTs, no matter what their PTs are, can incorporate easy-to-deploy services, which could basically be attached to the DTs, and given their structure, models, configurations, and light parameterization, should be able to provide these services in a plug-and-play fashion.

On-the-fly-learning. Given a well-defined structure for DTs and their data, i.e., taking advantage of DT schemas, DTs should be able to integrate AI mechanisms, more specifically, Machine Learning models, that can easily be attached to DTs and learn/predict during their execution. DTs convey lots of data, and these data are semantically structured, which may ease the process of learning from data.

Joint engineering for PTs. An interesting area of research is the engineering of PTs which are ready to be coupled to DTs. A significant portion of the implementation effort when doing the Digital Twinning process is the coupling of the DT with the communication interfaces of the PT which are not always standardized or properly structured. Therefore, having joint engineering from the conceptualization of a product/asset with a well-defined scope would decrease the implementation effort of the overall Digital Twinning process.

Applications of DTs in collaborative robotics. An extension to DT systems with coupled behavior (see Section 5.3) should include the *human-in-the-loop* concept, which would be represented by collaborative robotics in the robotics domain. Such an extension would require incorporating behavioral models of humans, which are likely stochastic, into the DT systems with coupled behavior.

Framework for assessing DTs. The reporting framework provided in Section 3.5 could be extended to provide an assessment mechanism per characteristic, so DTs can be case-independently assessed to objectively identify strengths and weaknesses regarding their capabilities.

Evaluation of methods in real industrial settings. The methods presented throughout this PhD thesis have been demonstrated and/or evaluated in theoretical or lab settings. Therefore, getting feedback from practitioners from applying the methods in more realistic industrial settings would provide data that can be used to more objectively assess the qualities of such methods.

Part II Publications

Survey on open-source Digital Twin frameworks - A case study approach

The paper presented in this chapter has been peer-reviewed and published in the journal *Software: Practice and Experience*.

[P1] S. Gil, P. H. Mikkelsen, C. Gomes and P. G. Larsen, "Survey on open-source digital twin frameworks–A case study approach," *Software: Practice and Experience*, vol. 54, no. 6, pp. 929–960, 2024, doi: 10.1002/spe.3305.

An Architectural Extension for Digital Twin Platforms to Leverage Behavioral Models

[P2] D. Lehner, S. Gil, P. H. Mikkelsen, P. G. Larsen and M. Wimmer, "An Architectural Extension for Digital Twin Platforms to Leverage Behavioral Models," 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, 2023, pp. 1-8, doi: 10.1109/CASE56687.2023.10260417.

A Modeling Approach for Composed Digital Twins in Cooperative Systems

[P3] S. Gil, P. H. Mikkelsen, D. Tola, C. Schou and P. G. Larsen, "A Modeling Approach for Composed Digital Twins in Cooperative Systems," 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania, 2023, pp. 1-8, doi: 10.1109/ETFA54631.2023.10275601.

Integrating Skills into Digital Twins in Cooperative Systems

[P4] S. Gil, C. Schou, P. H. Mikkelsen and P. G. Larsen, "Integrating Skills into Digital Twins in Cooperative Systems," 2024 IEEE/SICE 16th International Symposium on System Integration (SII), Ha Long Bay, Vietnam, 2024, pp. 1124-1131, doi: 10.1109/SII58957.2024.10417610.

An Architecture for Coupled Digital Twins with Semantic Lifting

The paper presented in this chapter has been submitted to the *International Journal on Software and Systems Modeling*. The paper is currently under *Major revisions* and the authors are working on the revised version at the time of writing. The current status of the revised manuscript is included.

[P5] S. Gil, E. Kamburjan, P. Talasila and P. G. Larsen, "An Architecture for Coupled Digital Twins with Semantic Lifting," Submitted for publication to the *International Journal on Software and Systems Modeling (SoSyM)*.

Towards a Systematic Reporting Framework for Digital Twins: A Co-operative Robotics Case Study

The paper presented in this chapter has been accepted for publication in the journal *Simulation: Transactions of the Society for Modeling and Simulation.*

[P6] S. Gil, B. J. Oakes, C. Gomes, M. Frasheri and P. G. Larsen, "Towards a Systematic Reporting Framework for Digital Twins: A Cooperative Robotics Case Study," *Simulation: Transactions of the Society for Modeling and Simulation*, pp. to appear, 2024, in press.

A Model-based Approach for Co-simulation-driven Digital Twins in Robotics

The paper presented in this chapter is *in progress* and the authors are working on it at the time of writing. It is intended to be submitted to the journal *Robotics and Autonomous Systems*.

[P7] S. Gil, A. Miyazawa, A. Badyal, P. G. Larsen and A. Cavalcanti, "A Model-based Approach for Co-simulation-driven Digital Twins in Robotics," In progress. To be submitted to the journal *Robotics and Autonomous Systems*.

References

[P1] S. Gil, P. H. Mikkelsen, C. Gomes, and P. G. Larsen, "Survey on open-source digital twin frameworks–A case study approach", *Software: Practice and Experience*, vol. 54, no. 6, pp. 929–960, 2024, ISSN: 1097024X. DOI: 10.1002/spe.3305.

[P2] D. Lehner, S. Gil, P. H. Mikkelsen, P. G. Larsen, and M. Wimmer, "An architectural extension for digital twin platforms to leverage behavioral models", in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2023, pp. 1–8. DOI: 10.1109/CASE56687.2023.10260417.

[P3] S. Gil, P. H. Mikkelsen, D. Tola, C. Schou, and P. G. Larsen, "A Modeling Approach for Composed Digital Twins in Cooperative Systems", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2023, pp. 1–8. DOI: 10.1109/ETFA54631.2023.10275601.

[P4] S. Gil, C. Schou, P. H. Mikkelsen, and P. G. Larsen, "Integrating Skills into Digital Twins in Cooperative Systems", in *2024 IEEE/SICE 16th International Symposium on System Integration (SII)*, IEEE, 2024, pp. 1124–1131, ISBN: 9798350312072. DOI: 10.1109/SII58957.2024.10417610.

[P5] S. Gil, E. Kamburjan, P. Talasila, and P. G. Larsen, *An architecture for coupled digital twins with semantic lifting*, under review, 2023.

[P6] S. Gil, B. J. Oakes, C. Gómes, M. Frasheri, and P. G. Larsen, "Towards a systematic reporting framework for digital twins: A cooperative robotics case study", *Simulation: Transactions of the Society for Modeling and Simulation*, pp. to appear, 2024, in press.

[P7] S. Gil, A. Miyazawa, A. Badyal, P. G. Larsen, and A. Cavalcanti, *A model-based approach for co-simulation-driven digital twins in robotics*, in progress, 2024.

[CP8] H. Feng *et al.*, "Integration Of The Mape-K Loop In Digital Twins", in *2022 Annual Modeling and Simulation Conference (ANNSIM)*, IEEE, 2022. DOI: 10.23919/annsim55834.2022.9859489.

[CP9] F. Naseri *et al.*, "Digital twin of electric vehicle battery systems: Comprehensive review of the use cases, requirements, and platforms", *Renewable and Sustainable Energy Reviews*, vol. 179, p. 113 280, 2023, ISSN: 1364-0321. DOI: 10.1016/j.rser.2023.113280.

[CP10] P. Talasila, C. Gomes, P. H. Mikkelsen, S. Gil Arboleda, E. Kamburjan, and P. G. Larsen, "Digital twin as a service (dtaas): A platform for digital twin developers and users", in *IEEE Smart World Congress (SWC)*, Portsmouth, UK: IEEE, 2023, pp. 1–8. DOI: 10.1109/SWC57546.2023.10448890.

[CP11] P. Talasila, P. H. Mikkelsen, S. Gil, and P. G. Larsen, "Realising digital twins", in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds. Springer, 2024, pp. 225–256, in press.

[CP12] B. J. Oakes *et al.*, "Case studies in digital twins", in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds. Springer, 2024, pp. 257–310, in press.

[CP13] L. A. C. Salazar, S. Gil, G. D. R. Carvajal, G. J. Sánchez-Zuluaga, and G. D. Zapata-Madrigal, "AI in assessing Industry 4.0 adoption in Colombia: a case study approach", in *2024 6th IFAC International Workshop on Advanced Maintenance Engineering, Services and Technology (AMEST)*, accepted for publication, Elsevier, 2024, to appear.

[14] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication", 2014.

[15] E. Ribeiro da Silva, A. Assad Neto, and C. P. Nielsen, "Digital twins: Making it feasible for smes", in *The Future of Smart Production for SMEs: A Methodological and Practical Approach Towards Digitalization in SMEs*, O. Madsen, U. Berger, C. Møller, A. Heidemann Lassen, B. Vejrum Waehrens, and C. Schou, Eds. Cham: Springer International Publishing, 2023, pp. 343–348, ISBN: 978-3-031-15428-7. DOI: 10.1007/978-3-031-15428-7_30.

[16] G. Barbieri *et al.*, "A virtual commissioning based methodology to integrate digital twins into manufacturing systems", *Production Engineering*, vol. 15, no. 3-4, pp. 397–412, 2021, ISSN: 18637353. DOI: 10.1007/s11740-021-01037-3.

[17] M. Dalibor *et al.*, "A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins", *Journal of Systems and Software*, vol. 193, p. 111 361, 2022, ISSN: 01641212. DOI: 10.1016/j.jss.2022.111361.

[18] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, "Experimentable Digital Twins-Streamlining Simulation-Based Systems Engineering for Industry 4.0", *IEEE TII*, vol. 14, pp. 1722–1731, 2018.

[19] R. Stark and T. Damerau, "Digital twin", in *CIRP Encyclopedia of Production Engineering*, S. Chatti and T. Tolio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 1–8, ISBN: 978-3-642-35950-7. DOI: 10.1007/978-3-642-35950-7_16870-1.

[20] M. Heithoff, M. Konersmann, J. Michael, B. Rumpe, and F. Steinfurth, "Challenges of Integrating Model-Based Digital Twins for Vehicle Diagnosis", in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, 2023, pp. 470–478, ISBN: 9798350324983. DOI: 10.1109/MODELS-C59198.2023.00082.

[21] J. Michael, J. Pfeiffer, B. Rumpe, and A. Wortmann, "Integration Challenges for Digital Twin Systems-of-Systems", in *SESoS*, ser. SESoS, IEEE/ACM, 2022, pp. 9–12. DOI: 10.1145/3528229.3529384.

[22] B. J. Oakes *et al.*, "Improving digital twin experience reports", in *MODELSWARD 2021 - Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development*, 2021, pp. 179–190, ISBN: 9789897584879. DOI: 10.5220/0010236101790190.

[23] E. W. Tsang, "Generalizing from research findings: The merits of case studies", *International Journal of Management Reviews*, vol. 16, no. 4, pp. 369–383, 2014, ISSN: 14682370. DOI: 10.1111/ijmr.12024.

[24] C. Schou, "Introduction to part 4", in *The Future of Smart Production for SMEs: A Methodological and Practical Approach Towards Digitalization in SMEs*, O. Madsen, U. Berger, C. Møller, A. Heidemann Lassen, B. Vejrum Waehrens, and C. Schou, Eds. Springer, 2023, pp. 299–308, ISBN: 978-3-031-15428-7. DOI: 10.1007/978-3-031-15428-7_24.

[25] P. K. R. Maddikunta *et al.*, "Industry 5.0: A survey on enabling technologies and potential applications", *Journal of Industrial Information Integration*, vol. 26, no. June 2021, p. 100 257, 2022, ISSN: 2452414X. DOI: 10.1016/j.jii.2021.100257.

[26] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective", *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2970143.

[27] E. VanDerHorn and S. Mahadevan, "Digital Twin: Generalization, characterization and implementation", *Decision Support Systems*, vol. 145, no. February, p. 113 524, 2021, ISSN: 01679236. DOI: 10.1016/j.dss.2021.113524.

[28] R. Wieringa and M. Daneva, "Six strategies for generalizing software engineering theories", *Science of Computer Programming*, vol. 101, pp. 136–152, 2015, ISSN: 01676423. DOI: 10.1016/j.scico.2014.11.013.

[29] C. R. Kothari, *Research Methodology: Methods and Techniques*. NEW AGE INTERNATIONAL (P) LIMITED, PUBLISHERS, 2004, vol. 2, ISBN: 978-81-224-2488-1.

[30] R. K. Yin, *Case study research: Design and methods*. sage, 2018, vol. 6.

[31] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009, ISSN: 13823256. DOI: 10.1007/s10664-008-9102-8.

[32] G. Muller, "Systems engineering research methods", *Procedia Computer Science*, vol. 16, pp. 1092–1101, 2013, ISSN: 18770509. DOI: 10.1016/j.procs.2013.01.115.

[33] A. T. Jebb, V. Ng, and L. Tay, "A Review of Key Likert Scale Development Advances: 1995–2019", *Frontiers in Psychology*, vol. 12, no. May, pp. 1–14, 2021, ISSN: 16641078. DOI: 10.3389/fpsyg.2021.637547.

[34] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review", *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020, ISSN: 17555817. DOI: 10.1016/j.cirpj.2020.02.002.

[35] E. A. Lee, "Cyber physical systems: Design challenges", in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369. DOI: 10.1109/ISORC.2008.25.

[36] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications", *Journal of Manufacturing Systems*, vol. 58, no. PB, pp. 346–361, 2021, ISSN: 02786125. DOI: 10.1016/j.jmsy.2020.06.017.

[37] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review", *Computers in Industry*, vol. 130, 2021, ISSN: 01663615. DOI: 10.1016/j.compind.2021.103469.

[38] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification", in *IFAC*, ser. IFAC, vol. 51, Elsevier, 2018, pp. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

[39] T. Ji, H. Huang, and X. Xu, "Digital Twin Technology — A bibliometric study of top research articles based on Local Citation Score", *Journal of Manufacturing Systems*, vol. 64, no. April, pp. 390–408, 2022, ISSN: 02786125. DOI: 10.1016/j.jmsy.2022.06.016.

[40] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling", *Journal of Manufacturing Systems*, vol. 64, no. July, pp. 372–389, 2022, ISSN: 02786125. DOI: 10.1016/j.jmsy.2022.06.015.

[41] M. M. Rathore, S. A. Shah, D. Shukla, E. Bentafat, and S. Bakiras, "The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities", *IEEE Access*, vol. 9, pp. 32 030–32 052, 2021, ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3060863.

[42] R. Paredis, C. Gomes, and H. Vangheluwe, "A family of digital t workflows and architectures: Exploring two cases", in *Innovative Intelligent Industrial Production and Logistics*, A. Smirnov, H. Panetto, and K. Madani, Eds., Springer, 2023, pp. 93–109, ISBN: 978-3-031-37228-5.

[43] C. K. Lo, C. H. Chen, and R. Y. Zhong, "A review of digital twin in product design and development", *Advanced Engineering Informatics*, vol. 48, no. March, 2021, ISSN: 14740346. DOI: 10.1016/j.aei.2021.101297.

[44] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research", *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2998358. arXiv: 1911.01276.

[45] Q. Qi *et al.*, "Enabling technologies and tools for digital twin", *Journal of Manufacturing Systems*, vol. 58, no. PB, pp. 3–21, 2021, ISSN: 02786125. DOI: 10.1016/j.jmsy.2019.10.001.

[46] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital Twin in Industry: State-of-the-Art", *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019, ISSN: 15513203. DOI: 10.1109/TII.2018.2873186.

[47] D. Lehner *et al.*, "Digital Twin Platforms: Requirements, Capabilities, and Future Prospects", *IEEE Software*, vol. 39, no. 2, pp. 53–61, 2022, ISSN: 19374194. DOI: 10.1109/MS.2021.3133795.

[48] Plattform Industrie 4.0, "Reference Architectural Model Industrie 4.0 (RAMI 4.0) - An Introduction", ZVEI - German Electrical and Electronic Manufacturers Association, Tech. Rep., 2016. [Online]. Available: https://web.archive.org/web/20210615073607/https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf?{_}{_}blob=publicationFile{\&}v=7.

[49] IEC, *Asset Administration Shell for industrial applications - Part 1: Asset Administration Shell structure*, IEC 63278-1:2023. Geneva, Switzerland: International Electrotechnical Commission, 2023. [Online]. Available: https://webstore.iec.ch/publication/65628.

[50] S. B. Trickett and J. G. Trafton, ""What if...": The use of conceptual simulations in scientific reasoning", *Cognitive Science*, vol. 31, no. 5, pp. 843–875, 2007, ISSN: 03640213. DOI: 10.1080/03640210701530771.

[51] J. Fritzsch *et al.*, "Adopting microservices and DevOps in the cyber-physical systems domain: A rapid review and case study", *Software - Practice and Experience*, vol. 53, no. 3, pp. 790–810, 2023, ISSN: 1097024X. DOI: 10.1002/spe.3169. arXiv: 2210.06858.

[52] S. Geman, D. F. Potter, and Z. Chi, "Composition systems", *Quarterly of Applied Mathematics*, vol. 60, no. 4, pp. 707–736, 2002.

[53] R. K. Keller and R. Schauer, "Design components: Towards software composition at the design level", in *Proceedings - International Conference on Software Engineering*, 1998, pp. 302–311, ISBN: 0818683686. DOI: 10.1109/icse.1998.671356.

[54] W. Jia, W. Wang, and Z. Zhang, "From simple digital twin to complex digital twin Part I: A novel modeling method for multi-scale and multi-scenario digital twin", *Advanced Engineering Informatics*, vol. 53, no. July, p. 101 706, 2022, ISSN: 14740346. DOI: 10.1016/j.aei.2022.101706.

[55] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Robust Digital Twin Compositions for Industry 4.0 Smart Manufacturing Systems", in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*, vol. 2018-Octob, IEEE, 2018, pp. 69–78, ISBN: 9781538641415. DOI: 10.1109/EDOCW.2018.00021.

[56] C. Human, A. H. Basson, and K. Kruger, "A design framework for a system of digital twins and services", *Computers in Industry*, vol. 144, no. June 2022, p. 103 796, 2023, ISSN: 01663615. DOI: 10.1016/j.compind.2022.103796.

[57] G. N. Schroeder, C. Steinmetz, R. N. Rodrigues, R. V. B. Henriques, A. Rettberg, and C. E. Pereira, "A Methodology for Digital Twin Modeling and Deployment for Industry 4.0", in *Proceedings of the IEEE*, vol. 109, 2021, pp. 556–567. DOI: 10.1109/JPROC.2020.3032444.

[58] D. Tola *et al.*, "Towards Easy Robot System Integration: Challenges and Future Directions", in *2022 IEEE/SICE International Symposium on System Integration, SII 2022*, 2022, pp. 77–82, ISBN: 9781665445405. DOI: 10.1109/SII52469.2022.9708846.

[59] J. Banks, *Handbook of Simulation*. Wiley, 1998, p. 849, ISBN: 0471 - 13403 - 1.

[60] A. Maria, "Introduction to modeling and simulation", in *Winter Simulation Conference Proceedings*, 1997, pp. 7–13. DOI: 10.1145/268437.268440.

[61] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey", *ACM Computing Surveys*, vol. 51, no. 3, 2018, ISSN: 15577341. DOI: 10.1145/3179993.

[62] S. T. Hansen *et al.*, "The fmi 3.0 standard interface for clocked and scheduled simulations", *Electronics*, vol. 11, no. 21, 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11213635. [Online]. Available: https://www.mdpi.com/2079-9292/11/21/3635.

[63] V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, "Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations", *Production and Manufacturing Research*, vol. 7, no. 1, pp. 472–489, 2019, ISSN: 21693277. DOI: 10.1080/21693277.2019.1660283.

[64] J. Fitzgerald, P. G. Larsen, and K. Pierce, *Multi-modelling and Co-simulation in the Engineering of Cyber-Physical Systems: Towards the Digital Twin*. Springer International Publishing, 2019, vol. 11865 LNCS, pp. 40–55, ISBN: 9783030309855. DOI: 10.1007/978-3-030-30985-5_4.

[65] C. Thule, K. Lausdahl, C. Gomes, G. Meisl, and P. G. Larsen, "Maestro: The INTO-CPS co-simulation framework", *Simulation Modelling Practice and Theory*, vol. 92, no. August 2018, pp. 45–61, 2019, ISSN: 1569190X. DOI: 10.1016/j.simpat.2018.12.005.

[66] C. M. Legaard, D. Tola, T. Schranz, H. D. Macedo, and P. G. Larsen, "A universal mechanism for implementing functional mock-up units", in *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, ser. SIMULTECH, Virtual Event, 2021, pp. 121–129.

[67] M. Frasheri, H. Ejersbo, C. Thule, and L. Esterle, "Rmqfmu: Bridging the real world with co-simulation for practitioners", in *Proceedings of the 19th International Overture Workshop*, H. D. Macedo, C. Thule, and K. Pierce, Eds., Overture, Oct. 2021.

[68] V. Devedžic, "Understanding Ontological Engineering", *Communications of the ACM*, vol. 45, no. 4, pp. 136–144, 2002, ISSN: 15577317. DOI: 10.1145/505248.506002.

[69] A. Gómez-Pérez, "Ontological engineering: A state of the art", *Expert Update: Knowledge Based Systems and Applied Artificial Intelligence*, vol. 2, no. 3, pp. 33–43, 1999, ISSN: 1465-4091. [Online]. Available: http://oa.upm.es/6493/1/Ontological{_}Engineering{_}A{_}st.pdf.

[70] H.-j. Happel and S. Seedorf, "Applications of Ontologies in Software Engineering", in *Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC*, Citeseer, 2006, pp. 5–9. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.5733{\&}rep=rep1{\&}type=pdf.

[71] M. Sabou, "An Introduction to Semantic Web Technologies", in *Semantic Web Technologies for Intelligent Engineering Applications*, 2016, pp. 53–81. DOI: 10.1007/978-3-319-41490-4_3.

[72] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, *et al.*, "Swrl: A semantic web rule language combining owl and ruleml", *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[73] M. J. O'Connor and A. K. Das, "Sqwrl: A query language for owl.", in *OWLED*, vol. 529, 2009, pp. 1–8.

[74] B. Siciliano and O. Khatib, *Springer handbook of robotics*. 2016, pp. 1–2227, ISBN: 9783319325521. DOI: 10.1007/978-3-319-32552-1.

[75] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research", *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 90–103, 2007. DOI: 10.1109/MRA.2007.339608.

[76] K. H. Tantawi, A. Sokolov, and O. Tantawi, "Advances in Industrial Robotics: From Industry 3.0 Automation to Industry 4.0 Collaboration", in *TIMES-iCON 2019 - 2019 4th Technology Innovation Management and Engineering Science International Conference*, IEEE, 2019, pp. 1–4, ISBN: 9781728137551. DOI: 10.1109/TIMES-iCON47539.2019.9024658.

[77] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and Industry 5.0—Inception, conception and perception", *Journal of Manufacturing Systems*, vol. 61, no. October, pp. 530–535, 2021, ISSN: 02786125. DOI: 10.1016/j.jmsy.2021.10.006.

[78] C. S. Franklin, E. G. Dominguez, J. D. Fryman, and M. L. Lewandowski, "Collaborative robotics: New era of human–robot cooperation in the workplace", *Journal of Safety Research*, vol. 74, pp. 153–160, 2020, ISSN: 00224375. DOI: 10.1016/j.jsr.2020.06.013.

[79] V. Satya Durga Manohar Sahu, P. Samal, and C. Kumar Panigrahi, "Modelling, and control techniques of robotic manipulators: A review", in *Materials Today: Proceedings*, vol. 56, Elsevier Ltd, 2022, pp. 2758–2766. DOI: 10.1016/j.matpr.2021.10.009.

[80] E. Madsen, D. Tola, C. Hansen, C. Gomes, and P. G. Larsen, "AURT: A Tool for Dynamics Calibration of Robot Manipulators*", in *Proc. of SII*, IEEE/SISE, 2022, pp. 190–195, ISBN: 9781665445405. DOI: 10.1109/SII52469.2022.9708769.

[81] A. Mazumder *et al.*, "Towards next generation digital twin in robotics: Trends, scopes, challenges, and future", *Heliyon*, vol. 9, no. 2, e13359, 2023, ISSN: 24058440. DOI: 10.1016/j.heliyon.2023.e13359.

[82] M. R. Pedersen *et al.*, "Robot skills for manufacturing: From concept to industrial deployment", *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016, ISSN: 07365845. DOI: 10.1016/j.rcim.2015.04.002.

[83] C. Schou, R. S. Andersen, D. Chrysostomou, S. Bøgh, and O. Madsen, "Skill-based instruction of collaborative robots in industrial settings", *Robotics and Computer-Integrated Manufacturing*, vol. 53, no. June 2016, pp. 72–80, 2018, ISSN: 07365845. DOI: 10.1016/j.rcim.2018.03.008.

[84] L. Gualtieri, E. Rauch, and R. Vidoni, "Methodology for the definition of the optimal assembly cycle and calculation of the optimized assembly cycle time in human-robot collaborative assembly", *The International Journal of Advanced Manufacturing Technology*, vol. 113, no. 7, pp. 2369–2384, 2021, ISSN: 1433-3015. DOI: 10.1007/s00170-021-06653-y.

[85] A. Bilberg and A. A. Malik, "Digital twin driven human-robot collaborative assembly", *CIRP Annals*, vol. 68, no. 1, pp. 499–502, 2019, ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2019.04.011.

[86] N. Kousi, C. Gkournelos, S. Aivaliotis, C. Giannoulis, G. Michalos, and S. Makris, "Digital twin for adaptation of robots' behavior in flexible robotic assembly lines", in *Procedia Manufacturing - CARV*, vol. 28, Elsevier, 2019, pp. 121–126. DOI: https://doi.org/10.1016/j.promfg.2018.12.020.

[87] P. Corke and J. Haviland, "Not your grandmother's toolbox-the robotics toolbox reinvented for python", in *ICRA*, IEEE, 2021, pp. 11 357–11 363.

[88] A. Cavalcanti *et al.*, "Verified simulation for robotics", *Science of Computer Programming*, vol. 174, pp. 1–37, 2019, ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2019.01.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642318301655.

[89] H. Feng, C. Gomes, C. Thule, K. Lausdahl, M. Sandberg, and P. G. Larsen, "The Incubator Case Study for Digital Twin Engineering", *journal=arXiv preprint arXiv:2102.10390,*, pp. 1–18, 2021. arXiv: 2102.10390. [Online]. Available: http://arxiv.org/abs/2102.10390.

[90] K. H. Steinkraus, Y. B. Hwa, J. Van Buren, M. Provvidenti, D. Hand, *et al.*, "Studies on tempeh. an indonesian fermented soybean food.", *Food Research*, vol. 25, pp. 777–788, 1960.

[91] H. Feng, C. Gomes, C. Thule, K. Lausdahl, A. Iosifidis, and P. G. Larsen, "Introduction to digital twin engineering", in *2021 Annual Modeling and Simulation Conference (ANNSIM)*, IEEE, 2021. DOI: 10.23919/annsim52504.2021.9552135.

[92] H. Feng *et al.*, "Integration Of The Mape-K Loop In Digital Twins", in *Annual Modeling and Simulation Conference (ANNSIM 2022)*, IEEE, 2022, pp. 102–113. DOI: 10.23919/annsim55834.2022.9859489.

[93] H. Ahuett-Garza and T. Kurfess, "A brief discussion on the trends of habilitating technologies for industry 4.0 and smart manufacturing", *Manufacturing Letters*, vol. 15, pp. 60–63, 2018, ISSN: 22138463. DOI: 10.1016/j.mfglet.2018.02.011.

[94] G. S. Day and P. J. Schoemaker, "Adapting to fast-changing markets and technologies", *California Management Review*, vol. 58, no. 4, pp. 59–77, 2016, ISSN: 21628564. DOI: 10.1525/cmr.2016.58.4.59.

[95] ISO, *Automation systems and integration - Digital twin framework for manufacturing*, ISO 23247:2021(E). Geneva, Switzerland: International Organization for Standardization, 2021. [Online]. Available: https://www.iso.org/standard/78743.html.

[96] A. Wortmann, *Digital Twin definitions*, https://awortmann.github.io/research/digital_twin_definitions/, Accessed on March 8, 2024, 2024.

[97] D. Kundisch *et al.*, "An update for taxonomy designers: Methodological guidance from information systems research", *Business and Information Systems Engineering*, vol. 64, no. 4, pp. 421–439, 2022, ISSN: 18670202. DOI: 10.1007/s12599-021-00723-x.

[98] F. Foldager, O. Balling, C. Gamble, P. G. Larsen, M. Boel, and O. Green, "Design Space Exploration in the Development of Agricultural Robots", in *AgEng conference*, Wageningen, The Netherlands, 2018.

[99] G. Lumer-Klabbers, J. O. Hausted, J. L. Kvistgaard, H. D. Macedo, M. Frasheri, and P. G. Larsen, "Towards a digital twin framework for autonomous robots", in *Proceedings - 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021*, IEEE, 2021, pp. 1254–1259, ISBN: 9781665424639. DOI: 10.1109/COMPSAC51774.2021.00174.

[100] E. Gamma, R. Johnson, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.