Journal Title
XX(X):??–??
©The Author(s) 2024
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Towards a Systematic Reporting Framework for Digital Twins: A Cooperative Robotics Case Study

Santiago Gil[1], Bentley J. Oakes[2], Cláudio Gomes[1], Mirgita Frasheri[1], and Peter G. Larsen[1]

Abstract

Digital Twins (DTs) can be constructed for many different applications, leading to substantial differences between different case studies. To be able to learn from the challenges and lessons learned by other DT practitioners, it is important that experience reports be consistent to facilitate comparisons. In this paper we merge three reference description frameworks for DTs, one generated from a systematic mapping study, one generated from an analysis of experience reports, and one from a systematic literature review, to come up with a unified characterization of DT applications. This analysis has identified six non-overlapping and three cross-cutting characteristics in the reference frameworks. This paper showcases the unified characterization with 21 characteristics to report on a DT case study called the *Flex-cell*, a manufacturing cell with two robotic arms used for cooperative assembly. The generalizability of this unified characterization is validated using a multi-case approach with another case study in robotics and another in the food industry. We call on the DT community to integrate these systematic reporting principles in their future DT experience reports such that other practitioners can learn from each other more effectively.

Keywords

Digital Twin, Experience report, Robotic system, Mobile Robotics, Taxonomy, Co-simulation, Modeling and simulation

1 Introduction

Digital Twins (DTs) are virtual representations of their physical counterparts where bi-directional communication is enabled[?]. This technology has attracted considerable research attention[?] as a modern approach for improving processes through monitoring, simulation, and virtual execution, achieving applications supported by virtual commissioning mechanisms[?]. Although development and operation of DTs may still have certain barriers due to complexity and ensuring a sufficient representation of the physical system, it is a promising technology to support the transition into the digitalized world that can be used for experimentation in a risk-free environment[?].

DT examples are already found in the robotics domain to represent and interact with the components of robotic systems with value-added services[?]. They can represent the entire robot and/or internal parts of it and generate outputs based on optimization goals. Due to the complexity of robotic systems, simulation models that are contained in DTs can cover one or multiple modeling aspects, such as kinematics, dynamics, rigid body, and control, among others[?]. In general, DT applications in robotics require the orchestration of the DT constellation[?] to incorporate different models, tools, and services to fulfill its particular purpose and scope.

When dealing with the DT engineering of cooperative and collaborative robots, composition can be used to approach the multiplicity of the independent sub-components that compose the larger system[??]. However, this is not an easy task and a current challenge in DTs for robotics[?]. Moreover, some aspects of composition, such as vertical composition, composition for different perspectives, composition of heterogeneous twin implementations, and composition of interfaces, are current challenges in relation to DT model-driven engineering methods[??]. All these challenges, plus a lack of guidelines for DT engineering and reporting, make it difficult to precisely describe the process and implementation of DT case studies.

Although there are several case study reports of DTs in literature, these reports have been mostly described based on individual case-specific requirements rather than on general requirements for DTs, making the reporting procedure difficult to digest by the readers from a broader audience[?]. No previous study has used a well-defined reporting framework (or a combination of reporting frameworks) for describing essential DT characteristics and their findings in relation to challenges and lessons learned. DT case study

[1]Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
[2]Polytechnique Montréal, Montréal, Canada

Corresponding author:
Bentley Oakes, Polytechnique Montréal, Montréal, Canada
Email: bentley.oakes@polymtl.ca

experience reports often lack detailed descriptions, and focus mostly on the system under study and its bi-directional connections. This hides relevant characteristics that i) may be relevant for the readers to understand the DT engineering process and ii) highlight the complexity behind such systems, which are then needed to describe the findings, challenges, and lessons learned.

Since DT technology is a modern concept, there is still lack of consensus on aspects, e.g., definitions[?], standardization[?], and reporting. The recent ISO-23247 standard[?] and IEC-63278-1[?] are pioneer standards in the DT technology[?], the former focusing on the general principles for DTs for manufacturing and the latter focusing on the use of Asset Administration Shell (AAS) representation for digital assets that can be used to exchange data in DTs. However, there is no standard or specification on how to report DT case studies.

To bridge the aforementioned gaps, we propose a merge of three existing description frameworks, namely, Oakes et al.[?], which presents 14 fundamental DT characteristics; Dalibor et al.[?], which presents a four-dimensional feature model describing DT features; and Jones et al.[?], which presents 19 themes, including 13 characteristics and seven knowledge gaps for DTs. The purpose of merging the three reference frameworks is to propose a more complete and unified reporting framework with 18 fundamental characteristics and three cross-cutting characteristics for reporting DT case studies.

The merged framework is then showcased to report on a sufficiently-complex case study in robotics, the *flex-cell*, a manufacturing cell composed of two robotic arms on a working plate for cooperative assembly. Also, with the aim of showing the generalizability of the reporting framework, two more case studies, one with a miniature agricultural robot and one with an incubator are briefly showcased.

The flex-cell case study, reflecting on similar setups in our industrial partners, addresses the challenge of non-trivial composition of DTs with coupled behavior. The example implementation of the flex-cell DT in the Digital Twin as a Service (DTaaS) platform proposed by Talasila et al.[?] is publicly available on the INTO-CPS Association GitHub repository[*]. Through reporting the resulting characteristics of the flex-cell, we show that this case study provides a representative and sufficient exemplar to be utilized as a reference in future DT experience reports.

The motivation behind reporting the flex-cell DT on a merge of three different frameworks is twofold; first, for the readers to easily identify the engineering requirements, process, and terminology in a structured way, and second, with the intention of presenting the challenges, lessons learned, and limitations of this case study regarding the DT engineering and reporting aspects, based on the guidelines provided by the unified characterization. Moreover, we foresee the potential of our systematic reporting framework as a basis for subsequent research and industrial reports and assessments in the DT domain.

Contributions Our main contributions are (1) systematically merging three reference description frameworks for reporting DT case studies to identify non-overlapping characteristics and provide a combined framework that covers a consistent set of characteristics, (2) reporting a representative exemplar DT case study in cooperative robotics using the resulting characteristics, (3) briefly reporting on two additional DT case studies with the proposed framework, one for mobile robotics and another for a food incubator, and (4) detailing the challenges and lessons learned from the experience we gained throughout the DT engineering and reporting processes.

Structure The remainder of this paper is structured as follows: **??** presents relevant background and related work for this case study report. **??** presents the systematic merging of the reference reporting frameworks presented by Oakes et al.[?], Dalibor et al.[?], and Jones et al.[?]. **??** elaborates on the flex-cell and its components by reporting it with the resulting unified characteristics. **??** describes two more case studies to argue for the generalizability of the reporting framework. **??** discusses the main challenges and lessons learned from this case study report and lists the identified limitations. Finally, **??** provides the concluding remarks and potential research directions.

2 Background and Related Work

This section will introduce concepts related to DTs, the composition of DTs, the reporting frameworks, robotic arms, and the used tooling. We also detail some related work of previous DT case studies and experience reports.

Background

Digital Twins (DTs) were proposed as a virtual representation of a physical asset, usually called the Physical Twin (PT), with enabled bi-directional communication[?], in comparison to their downgraded versions, Digital Models (with no communication at all) and Digital Shadows (DSs) (with unidirectional communication from the physical counterparts to the DSs). Although there are multiple definitions for DTs[??], these may require different components, ranging from models, data, tools, and services, depending on the business goals. In terms of modeling, DTs may cover different aspects of the physical system, including but not limited to geometric, physical, behavioral, rule, assembly, verification, and management modeling[?].

DT Engineering Some of the pioneer initiatives assisting the DT engineering process are "DT platforms"[?], which are mainly built using a meta-model approach, and enable the creation of DTs from properties and operations contained in a data model or schema. Nevertheless, some requirements for running simulation are not necessarily covered by these platforms, creating a gap between the DT and its behavioral components, which are essential to run experiments[?] and have virtual commissioning mechanisms[?]. Other tools, such as co-simulation frameworks, can also be used for the DT engineering and operation process when integrated with communication interfaces to achieve bi-directional communication, complementing the simulation and behavioral requirements[??]. Another survey of open-source frameworks for realizing DTs is presented by Gil

[*]https://github.com/INTO-CPS-Association/DTaaS-examples/tree/main/digital_twins/flex-cell

et al.[?], which focuses on tools and categories for the application of DTs in different domains.

The DT engineering process can also benefit from reusable components, such as models, tools, or services. Reusing components helps with reducing the implementation effort of some engineering tasks and increase the maturity level of the DTs and their components in comparison to creating DTs from scratch [??]. On the same agenda, reusable components can be hosted in a platform, such as a DTaaS platform[?], where all the infrastructure and services for DTs can be orchestrated.

Other implementations for DT deployment have used a model-based design approach to increase the reusability of DT components from a high-level perspective[?]. The reusable components can, for instance, be hosted in a DTaaS platform[?], where along with other tools and services, DTs have a different lifecycle and can be used for different purposes with less design and implementation effort.

Composable Modeling of DTs Following with the idea of reusable components, DTs can also be engineered with a composable approach. Composability in DTs is highly beneficial since it can lead the DTs to effectively reuse their internal components, such as structure, models, and tools, in a hierarchical way[?]. Hierarchical (dis)aggregation can also be used to group individual DTs and their components into larger DTs[?].

Although composable DTs are highly suitable due to their contribution to increase the reusability of their components, aspects, such as internal dynamics, heterogeneity, and relationships make the composition process complex.

In the robotics domain, the composition of DTs should also mirror the composition and constraints of the physical system[?], and therefore, both hardware and software interfaces need to be compatible.

The study on the composition of DTs[?] proposes a modeling approach that supports composition in cooperative systems and its extension to integrate the skill-based engineering concept[?], which have been assessed on the flex-cell case study. The base modeling approach is designed on top of an ontological model that defines four main components, namely, *Attributes* containing the state of the PT, *Operations* containing the direct and indirect operations of the DT, *Behaviors* containing the behavioral models of the DT, and *Relationships* containing the internal relationships between smaller DTs. As part of the *Relationships*, the property *isComposedOf* enables the composition of larger DTs from smaller DTs. Moreover, the relationships can be extended to other kind of properties, such as *cooperateWith*, *requires*, and so on. As a result, the composed DTs contain the attributes, operations, behavioral models, and relationships of their smaller DTs.

Tooling As for the tooling, several tools specific for the DT engineering process are used throughout this paper. The architectural extension for integrating behavioral models with DT platforms presented by Lehner et al.[?], called the *DT Manager*, is used for administrating the DTs and their interfaces, i.e., with the physical system and simulations. The DT Manager supports the integration of behavioral models through Functional Mock-up Units (FMUs) via the Functional Mock-up Interface (FMI[†])[?]. It also enables the administration of PT and DT at the same abstraction level, i.e., as *twins*, having access to DT services at the *DT service layer* seamlessly.

The DT Manager can be wrapped as a reusable tool, which is then embedded into the DTaaS platform proposed by Talasila et al.[?]. The DTaaS platform is a complementary approach across different design patterns for developing DTs. It provides an infrastructure to host and maintain DTs and their assets with a particular asset configuration definition. Within the platform, it is possible to convey existing DT platforms and multiple engines to run simulations of the DTs, and integrate them with third-party services. Nevertheless, the DTaaS platform may add additional overhead and requirements for the design of the DT to be hosted. The DTaaS platform defines a DT by the components *Data*, *Tool*, *Function*, and *Model* and the orchestration of infrastructure and DT services, therefore, it enables the *DT constellation*[?]. A DT can be initialized and administrated throughout its lifecycle with the DTaaS platform, in this case, using the DT Manager. The DTaaS platform also contains *Maestro*[?], a co-simulation engine to run coupled simulations based on the FMI interface.

Finally, some of the models in this case study use the Robotics Toolbox by Corke and Haviland[?]. Namely, we use the kinematic models available in the Robotics Toolbox and their solvers for finding the inverse kinematics and trajectory generation. These models are then wrapped as FMUs using the UniFMU tool[?], enabling their integration with the FMI interface.

Robotic Arms Also called robotic manipulators, robotic arms are electronically controlled mechanical devices with multiple links and joints[?], which can be controlled to complete certain tasks. These robotic manipulators can be classified into rigid and flexible[?]. The modeling and control of robotic manipulators may not be trivial tasks and these require some level of expertise[?]. In particular, behavioral aspects including kinematics and dynamics are important when considering simulation of robotic arms[?]. These models can be used at the control level or at the monitoring level, enabling behavioral representations in DTs[?].

DTs can be used along with robotic arms as an integral complement for control and monitoring of the physical robots and their application in collaborative robotics[?]. DTs can also assist to set up collaborative tasks in human-robot shared workspaces[??]. The survey on DTs for robotics presented by Mazumder et al.[?] showed that the composition of DTs from modules for this domain is still a current challenge due to high development expenses and lack of model or architecture generalization.

Reporting Frameworks With the aim of better describing DT engineering processes for experience reports, Oakes et al.[?] proposed a description framework composed of 14 essential characteristics of DTs, as described in **??**. This framework was then used to report on a smart drilling system and compared against the Asset Administration Shell (AAS) standard[?].

[†]https://fmi-standard.org/

Table 1. DT Characteristics proposed by Oakes et al.[?].

Characteristic	Description
C1: System-Under-Study (SUS)	The PT, its environment, and any agents present
C2: Acting Components	Any additions and modifications to the SUS which enables communication from the DT to the SUS
C3: Sensing Components	Any additions and modifications to the SUS which enables communication from the SUS to the DT
C4: Multiplicities	How many systems and DTs are involved in the DT ecosystem and their relationships
C5: Data Transmitted	The data transmitted from the SUS to the DT
C6: Insights/Actions	The information from the DT to agents in the SUS, or the automatic controlling actions from the DT to the SUS
C7: Services	The activities that the DT is used for. This could also be termed the *capabilities* or *usages* of the DT
C8: Enablers	Computational components which take models and data, and support the services of the DT
C9: Models and Data	The input and output for the enabler components, with some data coming from the SUS
C10: Constellation	The conceptual relationships within the DT of the models/data, enablers, and services
C11: Time-scale	The time-scale of the communication between the DT and SUS, and the computation within the DT. Includes data, insights, actions, and any simulations
C12: Fidelity Considerations	For each DT service, the considerations for fidelity (how the DT represents the SUS)
C13: Life-cycle Stages	The stages of the SUS (*ideation*, *realization*, *utilization*, etc.) which the DT is used for. If the scope of the SUS changes, this should also be reported
C14: Evolution	The evolution of the DT throughout its development (milestones, publications)

Similarly, Dalibor et al.[?] carried out a systematic mapping study on software engineering for DTs based on requirements, where they proposed a feature model to assist the engineering and operation of DTs made of four dimensions and subfeatures. This feature model (refer to[?] for more information on feature models) also helps to identify and report the steps involved in the DT engineering process. The dimensions are described in **??**.

Jones et al.[?] also proposed a set of *themes*, resulting in 13 characteristics and seven knowledge gaps/topics in the domain, as presented in **??**, which were the result of the analysis of a systematic literature review.

Related Work

Although there are relevant related work in relation to case studies in DT engineering, experience reports, and DTs of robotic arms, none has used the reference reporting frameworks mentioned in this study nor a combination of them to more accurately report their findings, challenges, and lessons learned. Additionally, no previous work has merged reporting frameworks for reporting based on unified and

Table 2. DT Dimensions proposed by Dalibor et al.[?].

Dimension	Description and subfeatures
Requirements Dimension	This dimension covers the basic constituents and characteristics of the DT under study. It is characterized by the subfeatures *Counterpart*, *Multiple Representation*, *Usage Phase*, *Representation Phase*, *Asset Interaction*, *Optimization*, and *Consist Of*
Realization Dimension	This dimension reports on how the DT is implemented and which tools and processes are used for the DT development. It is characterized by the subfeatures *Implementation*, *Tools*, and *Process*
Deployment Dimension	This dimension reports on hosting the DT and its connection to the real world. It is characterized by the subfeatures *Hosting* and *Connection*
Operation Dimension	This dimension reports on the operational features of the DT while it is running. It is characterized by the subfeatures *Horizontal Integration*, *Decision Making*, *Inputs and Events*, and *Outputs*

general requirements of DT engineering instead of focusing on individual, case-specific requirements. This leads the reports to mainly describe the aspects in relation to the SUS, DT services and usages, tools and enablers, models, data transmitted, and asset interaction, whereas the other aspects usually remain hidden from the reader.

As for previous works of the authors, there are three relevant case studies to relate to this work, namely, the *Desktop Robotti DT*[?], the *Incubator DT.*[?], and the DT for a manufacturing pilot line[?].

The Desktop Robotti is a small replica of the mobile field robot Robotti[?], which is used as a test-bed for experimenting at a low-cost with different DT services, such as visualization and monitoring, prediction, among others. Lumer-Klabbers et al.[?] proposed the FMI-based DT running along-side the Desktop Robotti, where both DT and PT were provided the same control commands. The Desktop Robotti DT reports on the SUS, own requirements, tools/enablers, models and data, DT services/usages, connection, constellation, insights and actions, sensing and acting components, data transmitted, validation process, and evolution. This case study is used to argue for the generalizability of our approach later on in the paper.

The *Incubator DT*[‡] by Hao et al.[?] is the DT for a thermal chamber with temperature control, monitoring, *what-if* analysis, and formally verified self-adaptation[??], which has been used for the incubation of Tempeh, an Indonesian fermented soybean food[?]. The incubator has also served as a case study in Lehner et al.[?] for its application integration with the DT Manager and the FMI interface, and in Gil et al.[?] to explore the capabilities of open-source DT frameworks based on the services provided by the *Incubator DT*. The incubator DT reports on the SUS, sensing and acting components, models and data, data transmitted, connection and hosting, tools/enablers, calibration and testing process,

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

Table 3. DT Themes proposed by Jones et al.[?].

Theme	Description
Characteristics	
Physical Entity	A real-world artifact, i.e., a PT
Virtual Entity	A computer generated representation of the physical artifact, i.e., a DT
Physical Environment	The measurable real-world environment within which the PT exists
Virtual Environment	Any number of virtual worlds or simulations that replicate the state of the physical environment and designed for specific use-case(s)
Fidelity	The number of parameters transferred between the physical and virtual entities, their accuracy, and their level of abstraction
State	The current value of all parameters of either the physical or virtual entity/environment
Parameters	The types of data, information, and processes transferred between entities
Physical-to-Virtual Connection	The connection from the physical to the virtual environment. Comprises of physical metrology and virtual realization stages
Virtual-to-Physical Connection	The connection from the virtual to the physical environment. Comprises of virtual metrology and physical realization stages
Twinning and Twinning Rate	The act of synchronization between the two entities and the rate with which synchronization occurs
Physical Processes	The physical purposes and process within which the physical entity engages
Virtual Processes	The computational techniques employed within the virtual-world
Knowledge Gaps	
Perceived Benefit	The envisaged advantages achieved in realizing the DT
DT across the Product Life-cycle	The life-Cycle of the DT – (whole life cycle, evolving digital profile, historical data)
Use-Cases	The applications of the DT
Technical Implementations	The technology used in realizing the DT
Levels of Fidelity	The number of parameters, their accuracy, and level of abstraction that are transferred between the virtual and physical twin/environment
Data Ownership	The legal ownership of the data stored within the DT
Integration between Virtual Entities	The methods required to enable communication between different virtual entities

and DT services/usages. This case study is used to argue for the generalizability of our approach later on in the paper.

The DT for a manufacturing pilot line by Kakavandi et al.[?] is a data-driven DT with integrated data analytics for high-volume production of medical devices. The case study reports on the SUS, models and data, DT services/usages, data transmitted, and tools/enablers.

Regarding other case studies on DT engineering, the DT for a vessel presented by Zhang et al.[?] provides services for early warning, lifecycle service support, system behavior prediction, and verification of the operational performance. The vessel DT relies on FMUs and co-simulation. The case study reports on the SUS, data collection and transmission, models and data, constellation, tools/enablers, DT services/usages, validation process, and lifecycle support (although not specific which stages).

More specifically to the robotics domain, Malik et al.[?] presented a case study of DT engineering for a human-robot collaboration assembly system for flexible automation solutions. Their case study uses a UR5e robotic arm and reports on the SUS, models and data, DT services/usages, actions and insights, and the joint engineering development process of the DT. Guerra-Zubiaga et al.[?] presented two case studies of DTs for manufacturing cells with robotic arms for telemetry monitoring and control. Their case studies report on the SUS, asset interaction, actions and insights, models and data, tools/enablers, inputs and outputs, connection and hosting, and DT services/usages.

In relation to experience reports on DTs, David et al.[?] report on a DT for controlled environment agriculture along with challenges and lessons learned, from requirements leveraged for DTs of cyber-biophysical systems coming from strategies for generalization of software engineering theories. Onaji et al.[?] report on three different case studies within the manufacturing domain and their corresponding benefits/lessons learned. The reported six characteristics come from the resulting conceptual framework of a literature review in DT applications within the manufacturing industry, which are used for describing the DT creation process.

Contrast to Previous Works

The previous works on which this paper is based upon are Gil et al.[??] where the flex-cell case study has been used to demonstrate and evaluate methods for a modeling approach for composable DTs and the integration of robot skills into this approach; the DT Manager by Lehner et al.[?] as an architectural framework for realizing DTs; the DTaaS Platform by Talasila et al.[?] as a hub for administrating DT components and tools; and the conceptual structure for DTs by Oakes et al.[?].

The differences of this research in comparison to the previous works listed above are i) this paper elaborates on a *fully-orchestrated* flex-cell DT, whereas the previous contributions either used the same case study, but only approached the flex-cell DT from the particular method's scope, or proposed a method for DTs without using the flex-cell DT as a case study. For this contribution, the approaches are combined and the report is based on those challenges that arose during combining the approaches. Additionally, the structure to use for the report is further analyzed and extended with two additional existing description frameworks for DTs. As a result, this work reports on and discusses a more complete flex-cell DT (which now contains experiments with coupled behavior via co-simulation and can be executed on the DTaaS Platform) and its engineering process, based on an extended set of characteristics.

3 A Unified Reporting Framework

This section presents our contribution of a unified reporting framework for DTs. We discuss the systematic procedure for merging three previous reference frameworks, and present the list of 21 characteristics and their description.

Rationale

The rationale for selecting these three reference frameworks by Oakes et al.?, Dalibor et al.?, and Jones et al.? is as follows: First, we selected Oakes et al.? because it was explicitly constructed for the description of DTs based on an analysis of experience reports. Then, we selected the feature model proposed by Dalibor et al.? because it provides a comprehensive and recent survey in the field of model-based/simulation DTs that covers the phases for engineering and realizing DTs. Finally, we selected the themes/characteristics by Jones? because it is a well-known, recent and top-cited review in the field of DTs? and provides a broader view of the characteristics of DTs.

Therefore, by having these three reference frameworks, we expect that the resulting characteristics come from more general definitions for DTs, producing a set of fundamental requirements to be reported on DT case studies. Using three frameworks also allows us to triangulate the characteristics, and thus, the merge of elements is clearer as we focus on those aspects which appear in at least two of the three frameworks. As a consequence, this helps improve the generalizability of the resulting framework.

Methodology

To systematically merge the characteristics presented in the three frameworks, we follow the methodology and recommendations presented by Kundisch et al.? to create a new set, in this case a taxonomy, based on the existing elements of the three reference frameworks/taxonomies.

Thus, we identify the meta-characteristics of the phenomenon based on the provided definitions and a collective perception, and then, cluster the elements based on similarities and differences, obtaining groups of objects, which are then merged into a resulting characteristic.

Since the terminology adopted in the three frameworks is similar and the concepts are within the same domain, the clustering is performed based on the similarity or proximity between the elements. Therefore, whenever there is proximity or equality between two elements, these are grouped into the same resulting characteristic, which represents the broader term for the merged elements.

In our procedure, the first two authors of this article each proposed (independently) a candidate taxonomy following the definitions in each reference framework. Then, a discussion was held between the two authors to solve inconsistencies and align terminology. This continued until the characteristics were agreed upon by both authors. The candidate result of the merge was then shared in the working group of this article's authors and discussed to obtain more general opinion and reduce individual bias.

We have assigned each element in the taxonomy a name which covers the broader concept, and also categorized the characteristics into four phases, similar as the dimensions proposed by Dalibor et al.?. These phases help to identify the order of the resulting characteristics from conceptualization to operation of DTs.

Merged DT Reporting Characteristics

?? shows the resulting merged characteristics (***MCi***) to be reported based on the results of the taxonomy update of the three reference reporting frameworks. The categorization of the characteristics into the phases of *Requirements, Conceptualization, and Design*, *Realization*, *Deployment*, and *Operation* is shown in **??**.

We propose that a total of 18 fundamental characteristics are reported (in bold). We also propose three cross-cutting characteristics (in italics), namely, *data ownership and privacy*, *standardization*, and *security and safety considerations*. These points are mentioned in the reference frameworks, but are not necessarily considered as characteristics. Although we do not propose them as *fundamental* characteristics, these three *cross-cutting* characteristics are worth mentioning since they are discussed in at least one of the three reference frameworks and focus on existing gaps in the DT domain, and are especially relevant for industrial case studies and scalable long-term applications. These cross-cutting characteristics are not applicable to all DT case studies, however, their inclusion in our reporting framework encourages authors to report on them when applicable.

During the merging process, the characteristics were grouped based on their meta-characteristics. Identifying the meta-characteristic was easy for most of the characteristics, since there is a quasi one-to-one relationship between the characteristics provided in the frameworks based on their descriptions. For those characteristics difficult to group up, we used a broader term covering the features of the three reference frameworks. In some of the characteristics, especially in Jones et al.?, the concepts are more abstract and general, and therefore, can cover more than one aspect and can be grouped into more than one element. In those cases, where there was a partial coverage of the base characteristic for the resulting characteristic, the key aspect is highlighted with italics and underlined in **??**. Moreover, the merge also required some considerations, which are described as follows:

- There is a distinction between technical and conceptual implementations for the connections at different levels, including physical and logical aspects, technology, and design considerations. Therefore, *MC2: Physical acting components* and *MC5: Virtual-to-Physical Interaction*, *MC3: Physical sensing components* and *MC4: Physical-to-Virtual Interaction*, *MC15: Digital Twin Technical Connection*, and *MC16: Digital Twin Hosting/Deployment* belong to different categories.
- *MC6: Digital Twin Services* differ from *MC17: Insights and Decision Making* in the sense that, the former refers to the expectations of what the DT is intended to offer (and so, how the DT is designed) and the latter to the insight generation during operation.
- The characteristic *Representation phase* in Dalibor et al.? is extra information regarding how the DT represents the PT in *MC9: Life-cycle stages*.
- The characteristic *Technical implementation* in Jones et al.? is split into several categories since it covers several technological aspects, i.e., it is not restricted to *MC11: Tooling and Enablers* only, which is the most similar category.
- For the characteristic *MC13: Twinning Process and Digital Twin Evolution*, *Twinning* refers to the methodology (which helps for the replicability), while

Table 4. Merge of the reporting frameworks by Oakes et al.[?], Dalibor et al.[?], and Jones et al.[?]. *In bold*: Fundamental characteristics. *In italics*: cross-cutting characteristics.

Oakes et al.	Dalibor et al.	Jones et al.	Resulting 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.
		Data Ownership	*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?*

Evolution refers to the milestones (which helps for the traceability).

- *MC18: Horizontal Integration* refers to integrations with other information systems not limited to other DTs, whereas *MC8: Multiplicities* focuses on the multiple DTs and their integration/orchestration.

Based on the merge, we identified six gaps with at least one of the three frameworks missing a description for such a resulting characteristic, and three cross-cutting characteristics. The gaps are **MC2-MC3**, **MC7**, **MC15-MC16**, and **MC18** and the cross-cutting characteristics are **MC19-MC21**. Two out of the six fundamental characteristics are only covered by one framework, whereas the remaining four are covered by two frameworks. Additionally, even though the descriptions by Jones et al.[?] have a criterion for each of the resulting characteristics,

Table 5. Categorization of phases for the resulting characteristics. *In bold*: Fundamental characteristics. *In normal text*: characteristics that also belong to this category but has been defined in an earlier phase. *In italics*: cross-cutting characteristics.

Requirements, Conceptualization, and Design	Realization	Deployment	Operation
MC1	**MC10**	**MC15**	**MC17**
MC2	**MC11**	**MC16**	**MC18**
MC3	**MC12**		
MC4	**MC13**		MC4
MC5	**MC14**		MC5
MC6			
MC7	*MC20*		*MC19*
MC8			*MC21*
MC9			

it does not necessarily mean that such a criterion covers sufficiently the corresponding characteristic. This is because the definitions are more general, and therefore, fit in more than one category. As a result, the resulting framework highly complements the one provided by Jones et al.[?] by making the characteristics more specific, and hence, easier to report.

4 Flex-cell Case Study

This section presents our case study of the flex-cell, including a brief overview and a thorough report through the unified reporting framework presented in **??**.

The flex-cell case study is a robot cell composed of two different robotic arms for cooperative assembly (see **??**), that has been used for exploratory and descriptive case study research within the phenomenon of DTs in robotics[?,?]. The research conducted on this case study follows the methodology proposed by Runeson and Höst[?], as a case study that partially falls into the software engineering domain.

The flex-cell is 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 connectivity for each asset is different; the robots must be connected through TCP sockets and the grippers through ModbusTCP. The connectivity for the UR5e robot is performed through the Python module URInterface[§] and for the Kuka robot through the module Kukalbrinterface[¶].

The flex-cell has provided sufficient complexity to seek insights and ideas for new research as an exploratory case study, and also to portray the challenges in this domain and provide a solid foundation for research, obtaining lessons learned, as a descriptive case study[?]. The scope and phenomena for which this case study research is conducted are in relation to i) composable DTs; ii) cooperative systems with synchronous motions and coupled models; iii) multiple abstraction levels of attributes and operations of both robots and grippers; 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, optimization in assembly, payload estimation, online recalibration, and collision detection, among others.

The particular applications that are run on the flex-cell case study in this work are based on a 2D cooperative assembly where the scope is related to robot positioning. The real workspace (x, y, z) of the flex-cell plate is discretized to (X, Y, Z), where the discrete positions use the plate holes, creating a grid of 16x24 holes separated by 5mm each.

?? shows two shapes, the square shape and the cross shape, which are assembled on the plate with pegs/pins. The robots take turns to pick a peg and place it in the corresponding hole. Before a motion is performed, a reachability analysis is done to check that the robot in turn can perform the movement; if not possible, the other robot takes over the movement to place the peg in the hole. The control sequences are deterministically generated based on a list of discretized vectors associated to the shape.

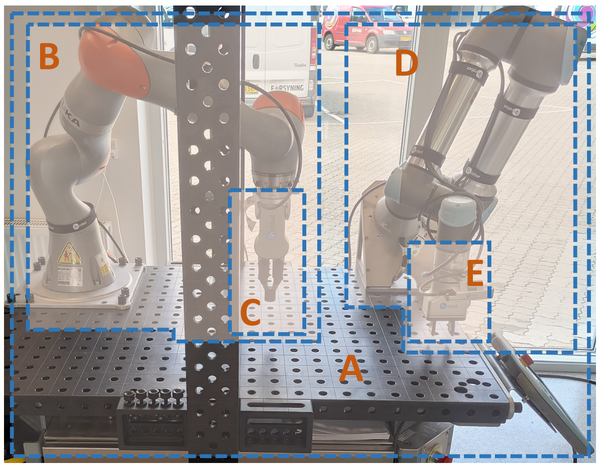

Figure 1. Flex-cell (A) and its components: robotic arms Kuka lbr iiwa 7 (B) and UR5e (D), and grippers OnRobot RG6 (C) and 2FG7 (E). The black surface (A) represents the discretized workspace.

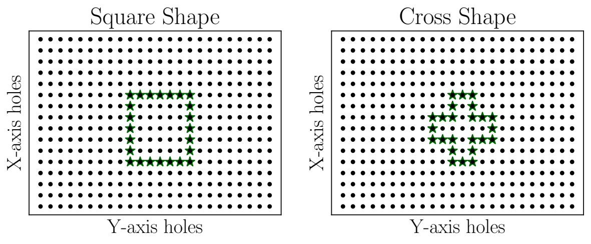

Figure 2. 2D assembly shapes carried out on the flex-cell plate.

Flex-cell DT Overview

The flex-cell DT is a constellation of models, tools, and services, which enable the representation and interaction mechanisms for the flex-cell system, considering its individual components. Since there are several individual components, the flex-cell DT is approached using the modeling approach proposed in[?], where each of the assets in the system, i.e., robotic arms and grippers, are individually represented and then composed into the flex-cell DT. **??**

[§] https://gitlab.au.dk/clagms/urinterface
[¶] https://github.com/sagilar/kukalbrinterface

shows an overview of the DT composition of the flex-cell using the modeling approach described in[?].

The components of the flex-cell DT are defined by data models, which describe the *Attributes* and *Operations* in form of schemas. The *Relationships* are managed at a higher abstraction level in an ontology. Finally, only the robotic arms - not the grippers - contain simulation models, i.e., *Behaviors*, as described in[?].

The individual FMUs are then used in co-simulations to generate the results of the coupled behavior of the flex-cell, i.e., the execution of the cooperative routines. This is required since the simulators of the UR5e and Kuka lbr iiwa 7 are different, and we need them integrated in the virtual space to represent a more similar setting to the real system.

For the co-simulation of the flex-cell we use the co-simulation orchestration engine *Maestro*[?] and the *RabbitMQFMU* (RMQFMU)[?]. The purpose of Maestro and the RMQFMU is to enable synchronization in the simulation when the physical robots are executing cooperative motions. In this sense, RMQFMU passes the commands of the physical controller to the simulated robots during runtime. **??** shows a representation of the co-simulation setup of the flex-cell to execute synchronous motions on the DT and PT using the RMQFMU.

We use the *DT Manager* approach proposed in[?] to handle the connectivity between PT and DT and the game engine Unity[‖] along with Unified Robotics Description Format (URDF) models to generate the visual representation of the flex-cell DT.

We use the DTaaS Platform proposed in[?] for the realization of the flex-cell DT and the orchestration of its constellation. The environment provided in the platform enables the combination of the components and methods described above.

Description with the Proposed Description Framework

We use our unified description framework presented in **??** to summarize the flex-cell with all characteristics in **??**.

The following details each framework characteristic, from a flex-cell DT implementation perspective:

MC1: System-under-Study

The flex-cell itself is the manufacturing cell described above with the two robotic arms with attached grippers. The *environment* is characterized by a plate with 16×24 holes where there are objects like Lego Bricks and pins/pegs to place on the holes. The environment also has a safety system, which is activated in case of a collision or an abnormal condition, e.g., one of the robots has the emergency stop active or the wiring for the safety signals is wrong. The safety system also has a proximity sensor, which is enabled when carrying out demonstrations for visitors to stop the robots in case anyone gets closer to the working space. Each robotic arm has a human-machine interface, where the *agent*, a human operator, can manipulate the robots or set them to work with remote control for automatic execution. The sequences to assemble a shape are executed remotely from a PC by the human operator.

MC2: Physical acting components

The acting components in the system are the two robotic arms, the two grippers, and the safety system. The commands sent to the robotic arms from the DT are processed by their internal controllers to perform different motion types and update their parameters, such as speed, acceleration, etc.

Similarly, the commands from the DT to the grippers are processed by their internal controllers, where the human operator can update the width/opening of the gripper and the gripping force. The safety system stops the robots if there is a collision or someone gets too close to the working space.

MC3: Physical sensing components

Regarding the sensors, the robotic arms contain several internal sensors from which it is possible to collect data, such as current position, target position, torque, current, voltage, speed, acceleration, and so on. Some of the variables for the robot contain their angular and cartesian value, and there are some variables that are specific for each joint and some that are for the whole robotic arm. We collect 117 and 31 variables for the UR5e robot and the Kuka lbr iiwa 7 robot respectively, although only data related to robot positioning are used in the flex-cell DT case study. For the grippers, we collect force and width/opening values. The safety system has a proximity sensor and internal flags in case any of the robotic arms collide.

MC4: Physical-to-Virtual Interaction

The transmission of data between PT and DT is managed by the DT Manager. It is possible to use the method `getAttributeValue` to get the value of attributes in the PT.

Inputs and events of the flex-cell DT are managed by the DT Manager as *attributes*, which refer to the *streaming* data coming from the PT to the DT[?].

The messages coming from the PT are fed into the flex-cell DT every time there is a new message in the MQTT queue. The attributes can also be updated on demand, in this case, based on the last element in the queue.

MC5: Virtual-to-Physical Interaction

The virtual-to-physical interaction for the flex-cell DT is twofold, namely, directly and indirectly. For direct interaction, the DT can execute *move* and *grasping/releasing* commands on the PT. For indirection interaction, the DT can either raise an alarm or set a parameter on the PT, such as the motion speed.

The transmission of data between DT and PT is managed by the DT Manager. It is possible to use the methods `setAttributeValue` and `executeOperation` provided by the DT Manager. The former allows to set a value on the PT, for example, a parameter, in case of the output of *insights*. The latter allows to execute direct actions on the PT, for example, move commands.

Outputs from DT to PT are managed as *attributes* for parameter update and as *operations* for direct actions.

MC6: Digital Twin Services

The services that the flex-cell DT provides are as follows:
What-if simulation[?] that can perform moving commands on both robots on both physical and virtual versions

[‖] https://unity.com/

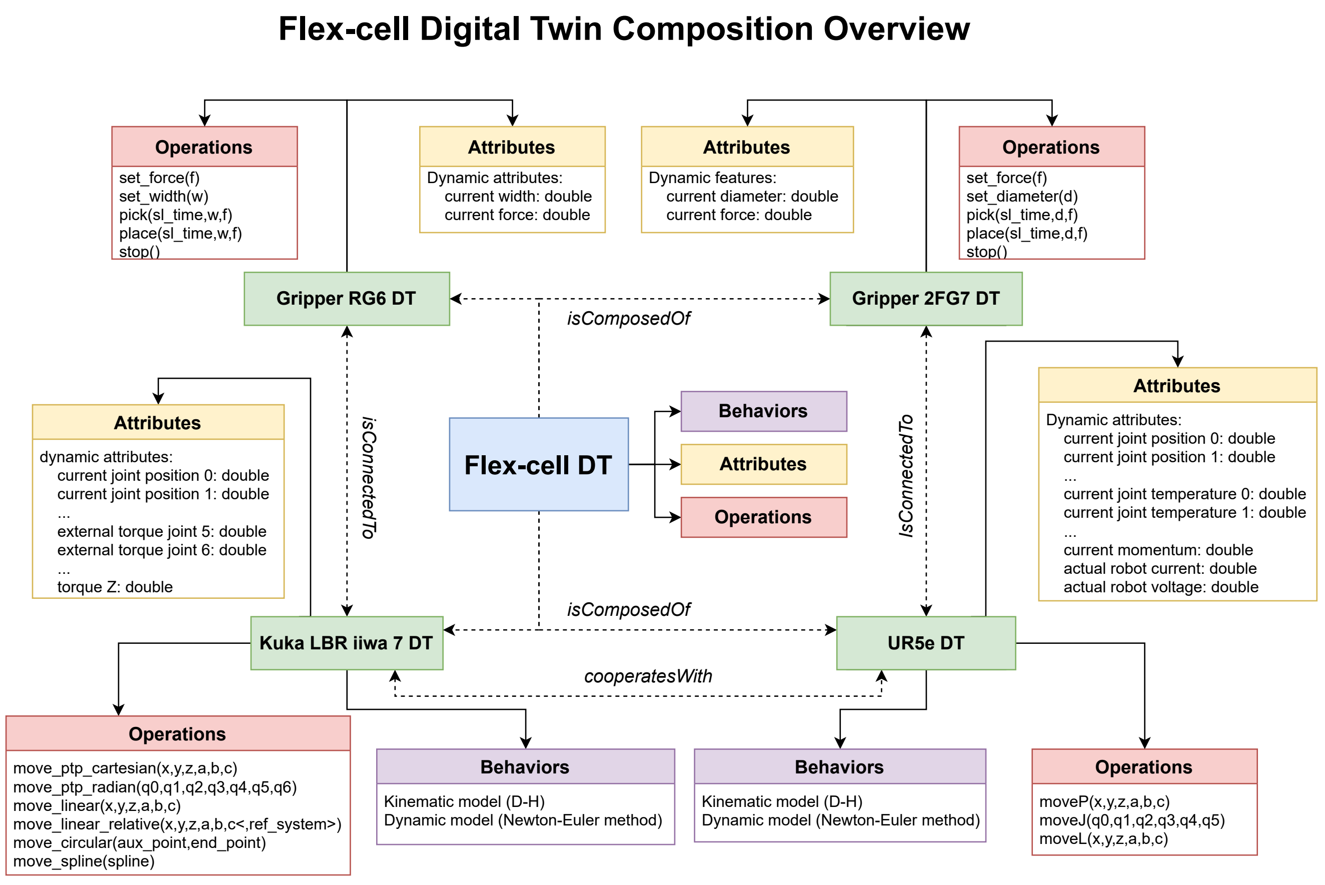

Figure 3. DT composition for the flex-cell case study (blue) using the modeling approach proposed in Gil et al.[?]. The flex-cell DT is composed of four smaller DTs (green), which are defined by attributes (yellow), operations (red), behaviors (purple), and relationships (dashed lines).

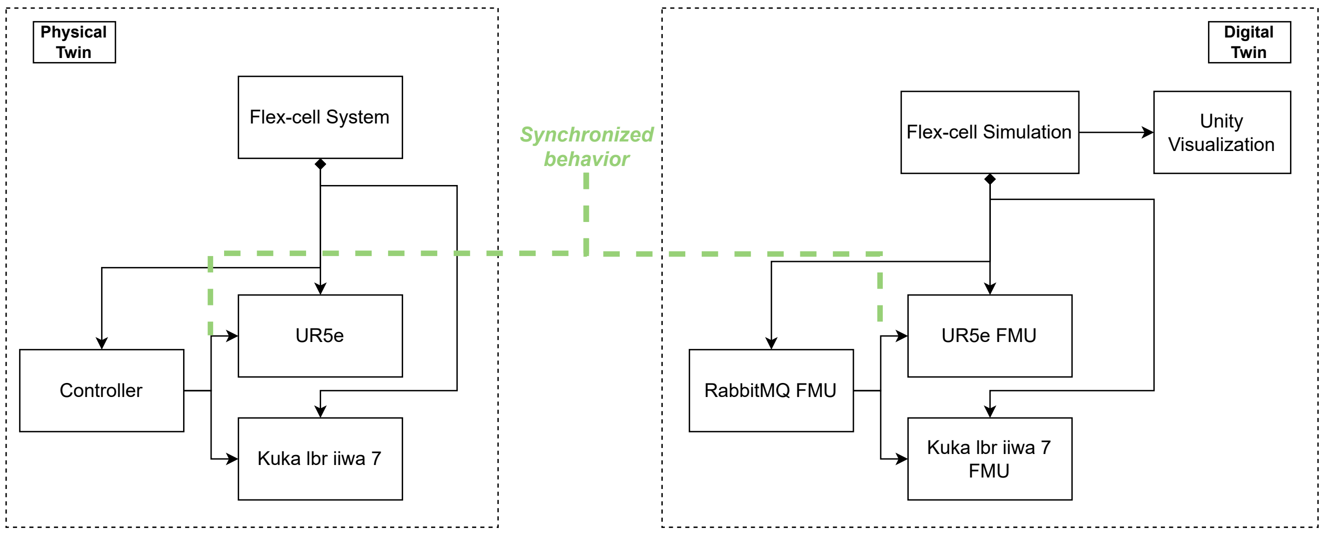

Figure 4. DT representation of the flex-cell DT in a co-simulation setting. Both real and simulated flex-cell are a composition of the three lower blocks *Controller/RabbitMQ FMU*, *UR5e/UR5e FMU*, and *Kuka lbr iiwa7/Kuka lbr iiwa 7 FMU* respectively.

synchronously in a cooperative setting to explore different scenarios. The DT can also be decoupled from the PT to perform certain experiments and validate on the virtual flex-cell before deployment to the physical one.

Trajectory visualization to have an idea of how the robots are going to move for a certain move command in the flex-cell working space (in connection with *what-if* simulations).

Discrete flex-cell working space-related commands that translate the desired (X, Y, Z) position in the flex-cell to joint angles of the robots. We use this feature as a service of the DT since this is case-specific and is not provided by manufacturers of industrial robots. To achieve this service, we use the kinematic models and some transformations in relation to the flex-cell working space to select the desired hole and height for the assembly tasks, as shown in **??**.

Table 6. Summary of the flex-cell DT case study through the characteristics of our proposed DT description framework.

Merged Characteristic	Flex-cell case study
MC1: System-under-Study	Manufacturing cell with independent assets (2 robotic arms, 2 grippers).
MC2: Physical acting components	Controllers of the robotic arms, grippers, and safety system.
MC3: Physical sensing components	Sensors of the robotic arms and grippers, including 117 observations for the UR5e, 31 for the Kuka lbr iiwa 7, and two for each gripper.
MC4: Physical-to-Virtual Interaction	The PT to DT interaction is managed by the DT Manager with the methods `getAttributeValue` on either a periodic basis or on event.
MC5: Virtual-to-Physical Interaction	The DT to PT interaction is managed by the DT Manager with the methods `setAttributeValue` for parameter update and `executeOperation` for direct actions.
MC6: Digital Twin Services	The flex-cell DT provides services for *what-if simulation*, *trajectory visualization*, *discrete working space commands*, and *deviation checking*.
MC7: Twinning Time-scale	The DT-to-PT synchronization is on demand, on a periodic basis, or on incoming events. The DT supports slower-than-real-time, real-time, and faster-than-real-time services.
MC8: Multiplicities	Each independent asset has its own DT and composition is enabled. There is no multiplicity of the same DT class.
MC9: Life-cycle stages	The services provided by the DT include the *design*, *manufacturing*, and *service* life-cycle phases. Within the *service* phase, it supports creating, executing, analyzing, saving, and terminating. The DTs cover the system as designed and as operated.
MC10: Digital Twin Models and Data	The flex-cell DT is provided with data models initialized from DT schemas using the AAS meta-model, structural models for visualization with the URDF format, and behavioral models wrapped as FMUs for the kinematics of the robotic arms. Data in the DT are related to robot positioning and handled as *attributes*, which hold the state of the twins. The number of attributes depends on the number of available observations in the model or provided by the sensors.
MC11: Tooling and Enablers	The elements supporting the services of the flex-cell DT include the DT Manager for interfacing and access to DT services; the Robotics Toolbox and UniFMU to encapsulate the behavioral models of the robotic arms; the URInterface, the Kukalbrinterface, ModbusTCP, MQTT, and RabbitMQ for connectivity; Maestro and RMQFMU to run the co-simulation scenarios; URSim to emulate the UR5e robot; Unity, ZeroMQ, and URDF for visualization; and The DTaaS Platform to execute the DT on the cloud.
MC12: Digital Twin Constellation	The orchestration of the system-as-a-whole, including the models and data, tools and enablers, services, and physical-to-virtual and virtual-to-physical interaction is defined. The constellation also describes how the DT behaves in the multiple scenarios for the provided services. Additionally, some of the components are initialized from configuration files and scripts.
MC13: Twinning Process and Digital Twin Evolution	The DT was engineered based on an existing manufacturing cell with a set of own requirements. The evolution presents 12 milestones.
MC14: Fidelity and Validity Considerations	The DT contains sufficiently accurate models for the robotic arms and overall execution. Low coverage of the models for the grippers. The flex-cell DT has been experimentally validated and provides mechanisms for consistency checking.
MC15: Digital Twin Technical Connection	Connection to the physical assets needs to be done on a LAN. Several communication protocols are used for the whole system deployment.
MC16: Digital Twin Hosting/Deployment	The flex-cell DT can be deployed on a LAN or on the DTaaS platform in the cloud.
MC17: Insights and Decision Making	The flex-cell DT can provide insights in form of simulation-based analysis and semantic reasoning.
MC18: Horizontal Integration	There is horizontal integration with the flex-cell PT and infrastructure services of the DTaaS Platform. The flex-cell DT is able to exchange information with other information systems not limited to other DTs.
MC19: Data Ownership and Privacy	Not considered in case study.
MC20: Standardization	Behavioral models conform with the FMI Standard Version 2. Twin schemas conform with the AAS meta-model (IEC-63278-1).
MC21: Security and Safety Considerations	Security aspects inherited from the DTaaS TLS. Safety aspects regarding remote operation for accidents and collisions.

Deviation checking compares that the physical robots are moving as expected. We use the simulations to check if there is any kind of deviation. The DT Manager is used as the interface to check whether physical and virtual robots are moving similarly.

MC7: Twinning Time-scale

The DT-to-PT synchronization has two ways to be implemented. First, one twin can be synchronized to the values of another twin on demand by using the one of the methods provided by the DT Manager. Second, a twin can be synchronized through its attributes and operations on demand either on a periodic basis or on incoming events (applicable for the MQTT and RabbitMQ endpoints).

Additionally, the flex-cell DT operates at different rates:

Slower-than-real-time These services are adapted to the interfaces of the DT Manager in the service layer to generate an output, e.g., for deviation checking, cooperative motions, or collision warnings, based on PT and simulation input data. These services have an additional overhead which introduces delays, and therefore, they cannot run precisely in real-time.

Real-time Control commands, position calculation, data logging (from the robot interfacing libraries), and visualization.

Faster-than-real-time For virtual commissioning and what-if simulations, and so the DT is decoupled from the PT to run independent simulations and *what-if* scenarios.

MC8: Multiplicities

The flex-cell DT is addressed with a composable modeling approach, and thus, the flex-cell DT is composed of more than one DT instance. There are four individual DTs corresponding to four assets in the flex-cell system, which are then composed into robotic arm + gripper, and so composed into the flex-cell DT, such as illustrated in **??**.

Additionally, the DT for the flex-cell can have multiple representations, i.e., multiple twins for the same physical system. This is because 1) the system may have different models featuring different functionalities with different scopes; 2) the implementation through the DT Manager enables having multiple twins for the same physical system; and 3) due to the modeling approach with composition, it is possible to compose smaller sub-components which are related somehow. The current state of the flex-cell DT does not utilize multiplicities of the same DT class.

MC9: Life-cycle stages

The services provided by the flex-cell DT include the *design*, *manufacturing*, and *service* life-cycle phases presented in Liu et al.[?]. Moreover, we adopt the life-cycle phases presented in Talasila et al.[?] as a reference for the internal phases during *service*, namely, *create, execute, save, analyze, evolve, and terminate*.

The *service* phase is administrated by the DT Manager, which can be deployed from the DTaaS Platform or manually given the case. For the *(re-)design* phase, the virtual commissioning service of the DT is used, i.e., decoupled of the PT, to validate changes to the applications to be run in the physical system. The other life-cycle phases that are covered in the flex-cell DT during *service* are the *create, execute, analyze, save,* and *terminate* life-cycle phases.

As for the representation phase of the DT, it is as designed (when working at the individual level) or as operated (at the cooperative level).

MC10: Digital Twin Models and Data

The flex-cell DT contains different models, all wrapped and described in the composed model as described in[?].

Data models There is a data model for each asset, i.e., each robotic arm and gripper, which contains *Attributes* and *Operations*; these are initialized from DT schemas using the AAS meta-model. These models contain only static data and are used to initialize the DT on the DT platform, with the interfaces to act on the PT/simulation (i.e., DT outputs) and to store the state of the PT/simulation (i.e., DT inputs).

Structural models Each asset, i.e., robotic arms and grippers, has also a corresponding URDF model, which are used uniquely for visualization. These models contain the descriptions of the rigid bodies, such as links and joints.

Behavioral models Additionally, the robotic arms contain behavioral models, mapped to *Behaviors*, which are i) kinematic models using the Denavit-Hartenberg parameters and ii) dynamic models with the Newton-Euler formulation (not being used). The behavioral models are wrapped as FMUs. There are no behavioral models for the grippers.

Data The relevant data in the flex-cell DT is related to robot positioning on a 2D space, as it is the scope of the DT. These data include cartesian and angular/joint positions. The data available in the flex-cell DT are administrated as *attributes*, which hold the state of the Twins. The number of attributes depends on the observations given in the models (for the DT) and in the *streaming* data[?] coming from the physical sensing components (for the PT - see **MC3** for more information). The models provide observations for the cartesian and angular/joint positions for each robot plus any linearly dependent transformations/combinations based on these positions.

MC11: Tooling and Enablers

The following elements are *tools and enablers* for supporting the services of the flex-cell DT:

- The DT Manager to provide the access and interfaces to PT and DT, and to the DT services.
- The Robotics Toolbox and UniFMU to encapsulate the kinematic models of the robotic arms.
- The URInterface, Kukalbrinterface, ModbusTCP, MQTT, and RabbitMQ to access the physical robots and grippers and bridge the communication to external networks.
- Maestro and the RMQFMU to perform the synchronous and coupled co-simulations.
- URSim to emulate the UR5e robot and run experiments on it.
- Unity, ZeroMQ, and URDF for displaying visualizations.
- There is handcrafted code written on Python and Java to bind some of the interfaces and set up the threads/initializers for the services.
- The DTaaS Platform[?] is used to orchestrate the DT constellation and the execution of scripts from configuration files when hosted there.

MC12: Digital Twin Constellation

?? illustrates the *constellation* of the flex-cell DT with the models/data, enablers, and services that are coupled with the disaggregrated representation of the flex-cell system.

Additionally, some of the components in the DT constellation are initialized from configuration files. That is, the communication interfaces for DT and PT are initialized from configuration files using the DT Manager. The skeletons to hold the state, that is, the data of the twins, are initialized from the DT schema files, which in this case is based on AAS. The co-simulation experiments are also initialized from configuration files, defining the inputs and outputs in the co-simulation blocks, the time-steps, and the parameters to be passed to the involved FMUs. When using the DTaaS Platform, scripts that are associated to life-cycle phases automate the execution for those phases.

Regarding the *slices*[?] of the flex-cell DT, these represent the multiple scenarios for the existing services as follows:

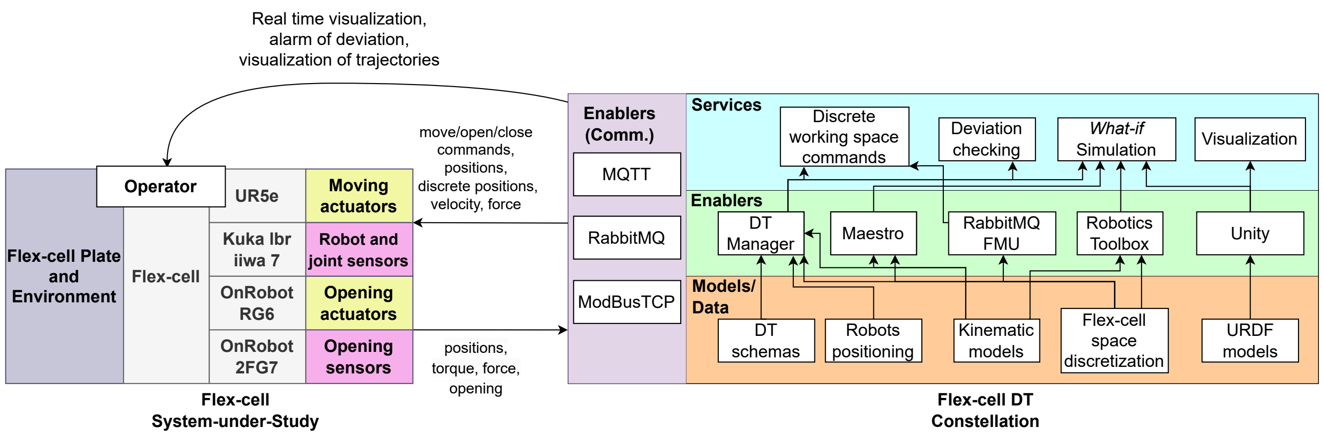

Figure 5. Constellation? of the flex-cell DT, detailing the composition of the DT and the data flow.

In the scenario for *what-if simulation*, *deviation checking*, and implicitly *discrete working space commands*, **??** shows one of the experiments of running coupled cooperative motions along with the co-simulation; the figure shows the joint positions over time of the two robots. The experiment consists of moving the robots between two different discrete waypoints (UR5e/Kuka lbr iiwa 7) (1) $(X, Y, Z) = (5, 14, 2)$ / $(X, Y, Z) = (3, 10, 1)$, and (2) $(X, Y, Z) = (4, 18, 1)$ / $(X, Y, Z) = (4, 7, 2)$. From the experiments, it is possible to see two kinds of delays between PTs and DTs. The first delay is due to executing the command. Although the move commands are sent to PT and DT at the same time, there is a communication and processing latency in the actual controllers to start with the execution of commands. The second delay is due to behavioral differences between PT and DT model. The real robots take some time to perform a motion, and the models of the DT set an approximation for the time it takes to perform the motion. This parameter can be updated online and has a precision of 0.05 seconds. While the motion time is experimentally validated, there may still be minor errors, negligible for the scope of this work.

In the scenario for *what-if analysis* and *trajectory visualization*, **??** shows the graphical representation of the flex-cell system in Unity. The Unity application embeds the URDF models and the incoming messages from the virtual or real flex-cell are translated into motions, enabling the visualization of the system as a DT (data from simulation) or as a DS (data from real robots) respectively. When using the virtual flex-cell decoupled from real flex-cell, this setup is also used for virtual commissioning where it is possible to see the trajectories of the robotic arms before deployment.

In the scenario for *deviation checking*, a monitoring function that checks for deviations at periodic intervals is attached to the DT Manager. The function takes four arguments, namely, `PhysicalTwin` object, `DigitalTwin` object, `VariableName` string, and `Tolerance` percentage, and returns *true* when there is a deviation in the variable between the two objects given the tolerance, otherwise *false*.

MC13: Twinning Process and Digital Twin Evolution

The flex-cell DT is created with a subsequent engineering approach since the DT is adapted to an existing product(s), and so, it inherits its constraints and knowledge.

The particular requirements for the flex-cell DT are as follows: (i) it needs to comply with some real-time capabilities for control commands, data recording, and visualization; (ii) it needs to comply with some consistency requirements, such as that a composed DT cannot be a composition of a robotic arm attached to the end-effector of another robotic arm, i.e., it needs to be a gripper instead; and (iii) the internal components of the flex-cell DT need to be reusable.

As for the evolution, the flex-cell DT has evolved considerably during development. Some of the steps are reported in??. The most representative milestones M_i are as follows:

M_1 The DT schema-based object models for robotic arms and grippers were designed.
M_2 The object models were coupled to the DT Manager architecture.
M_3 The visualization with Unity and the URDF models were created.
M_4 The composable representation of the flex-cell was established.
M_5 The kinematic and dynamic models for the robotic arms were designed.
M_6 The space and mapping model for the flex-cell was created.
M_7 The execution of online PT-to-DT movements with individual non-cooperative motions was achieved.
M_8 The first experiments of applications for the flex-cell discrete working space using the DT as the virtual commissioning mechanism were successful.
M_9 The skill-based engineering approach was integrated to disaggregate the *Operation* into *device primitives*, *skills*, and *tasks*.
M_{10} The kinematic models were embedded into FMUs.
M_{11} The co-simulation with Maestro and RMQFMU was added to provide simulation capabilities of the coupled cooperative motions.
M_{12} Synchronous cooperative experiments were successful and used for deviation checking.

MC14: Fidelity and Validity Considerations

As for quality assurance, the validation of the flex-cell DT has been carried out through experimental validation as follows: The motion speed of behavioral models has been tuned so they approximate to the actual motion trajectory. The discrete to cartesian mapping has been tuned so the poses for the end effector are approximately centered in the holes of the flex-cell plate for all the feasible (X, Y, Z) poses. The visualization in Unity has been tuned so the

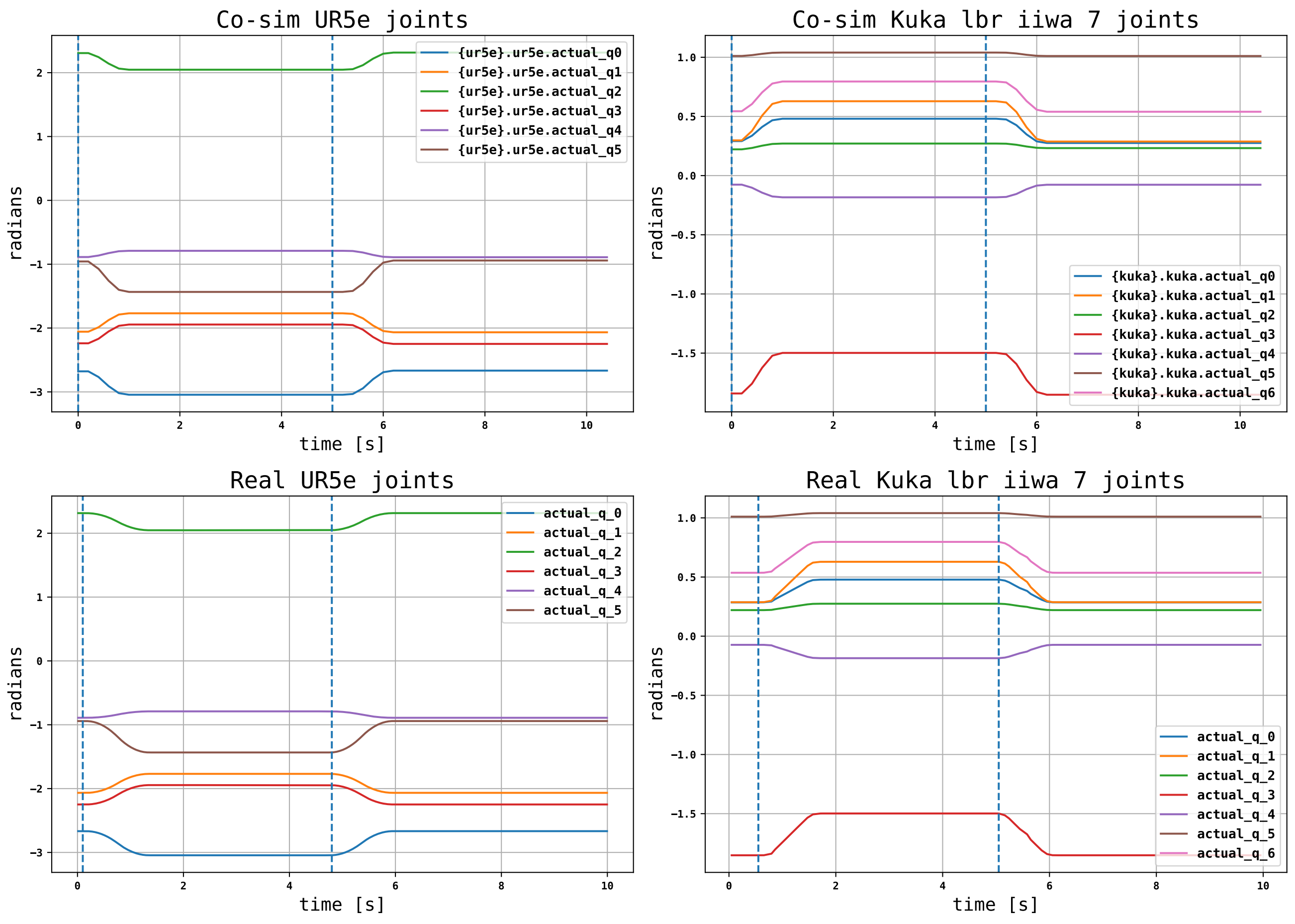

Figure 6. Plot of the joint positions of the robotic arms in the cooperative co-simulation setting. The vertical dashed lines indicate the invocation of moving commands, considering the delays in the real robots.

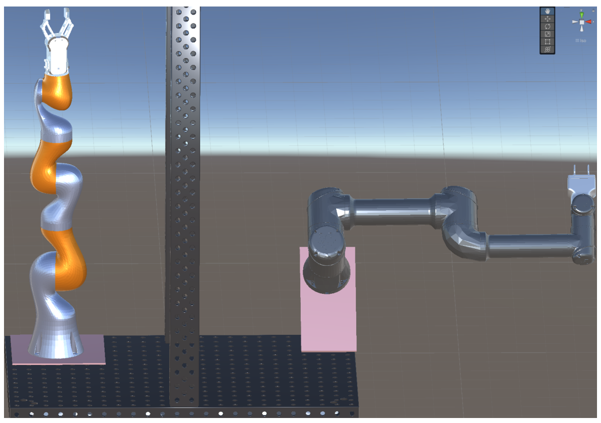

Figure 7. Flex-cell graphical representation in Unity.

poses for the robots and the motions are similar to how they look in the physical setup. The data models, including the composed representation and the functional requirements for bi-directional connections have been validated through a case study demonstration in[?][?].

Additionally, the representation of the flex-cell DT on top of the ontological model presented in[?] enables consistency checking to validate that structural constraints are complied. the flex-cell DT uses simulation and some features of consistency checking on top of the ontological model and its constraints that are imposed. On the downside, the simulation models do not consider the actual dynamics of the system nor are calibrated to the physical conditions.

Overall, the flex-cell DT requires a certain level of fidelity in its models to perform reasonably well with respect to the provided services[?], especially the ones related to the positioning of the robotic arms. Note that non-sufficient fidelity may cause the robots to collide with static objects like the flex-cell structure, with other dynamic objects like the other robotic arm or grippers, and with humans. This can lead to physical damage to equipment and injuries.

Some of the limitations in terms of model fidelity are as follows:

- The kinematic models do not include the kinematics for the grippers.
- The trajectory generation with the kinematic models provides certain time behavior based on an assumed motion speed (which can be tuned during operation). However, it does not consider the actual dynamics of the robots, and so, it can have time offsets. Using dynamic models may overcome this problem.
- The motions performed by the simulations may differ from the motions performed by the actual controllers as the algorithms that operate in actual controllers are closed source. A more accurate representation of the real controllers in simulation and visualization can mitigate this problem.

- The co-simulation executions for cooperative motions perform a given number of steps before updating since executing stepped simulations is computing-demanding. Thus, the synchronization with the PT takes longer, which can lead to additional errors in time and accuracy in the deviation detection. This problem can be overcome by injecting the PT state to the co-simulation with RMQFMU every step.

MC15: Digital Twin Technical Connection

Each asset in the flex-cell has its connection requirements, although all connections are established over TCP and have bidirectional connection enabled. The UR5e robot is controlled with the URInterface, the Kuka lbr iiwa 7 is controlled with the Kukalbrinterface, and both grippers are controlled with ModbusTCP. The robots' particular protocols are then bridged to MQTT (RabbitMQ is also available) to be accessible from the DT Manager and the DTaaS Platform given the case.

MC16: Digital Twin Hosting/Deployment

The hosting of the DT is done twofold. The first option is hosting the flex-cell DT on a local computer where all the network services and infrastructure can be easily deployed. The connection to the PT is then much easier since it can be achieved on a Local Area Network (LAN) with fewer technical complications.

The second option is hosting the flex-cell on a external server. In this case, we use the DTaaS platform and the example is available on GitHub**??**. Additionally, when hosting the flex-cell DT on the DTaaS platform, it needs to stick to the structure of the platform for reusing the network and architecture services, and the structure to run the multiple life-cycle phases available on the platform. On the downside, the connection to the PT is not straightforward since it needs to be bridged between the DTaaS platform to a LAN due to security and safety reasons. For this, a script forwarding messages from the particular robot's interface to and from MQTT is used on a Raspberry PI connected to both the LAN and the DTaaS platform brokers.

MC17: Insights and Decision Making

In terms of the decision making, although the flex-cell DT is mostly behaving according to pre-programmed sequences, there are two scenarios where the DT can make predictions or decisions, namely, simulation-based analysis and semantic reasoning. Simulation-based analysis is possible due to including the models of the flex-cell components, for example the kinematic models of the robotic arms, which are used to detect anomalies or deviations and raise an alarm. These models also enable the visualization with Unity, which provides insight into how the trajectory will be performed, as shown in **??**. Semantic reasoning is enabled by the underlying ontological model that is used for the composed flex-cell DT model, which enables semantic reasoning through queries and inferences, although these are mostly used for model/structure consistency checking of the system than for the actual analysis in comparison to the PT.

MC18: Horizontal Integration

In the horizontal interaction of the flex-cell DT, we identified the communication with its PT (and its sub-components) and with some services running on the DTaaS platform (when deployed from there), which handle the life-cycle of the flex-cell DT. Although the flex-cell DT contains several smaller DT sub-components, they are deployed in a centralized way, and therefore, there is no horizontal integration among the smaller DTs.

Overall, the flex-cell DT is able to exchange information with other systems, including external information systems and other DTs if required. This would require an adaptation of the algorithms for handling messages though.

MC19: Data Ownership and Privacy

There is no consideration regarding this characteristic in the flex-cell DT as it is an academic case study.

MC20: Standardization

The behavioral models of the flex-cell are designed in such a way they conform with the FMI standard**??** Version 2, and thus, the models are wrapped as FMUs. The data models are imported from AAS schemas that conform with the IEC-63278-1 standard.

MC21: Security and Safety Considerations

There are two safety considerations in the flex-cell case study. First, the physical flex-cell is provided with a proximity sensor which disables the execution coming from any source to avoid accidents with nearby humans. Second, the connection to the PT is achieved on a LAN and not directly from the cloud due to some critical operations, such as, for example, movement commands on the robotic arms cannot be executed unless there is someone at the facilities checking that the environment is safe to work to avoid accidents in the infrastructure or even with humans. In case the flex-cell is required to work remotely, an in-situ operator needs to enable the connection with the cloud by initializing the forwarding bridge and ensuring that the environment is safe to operate.

There are no security considerations more than the existing security layers on the DTaaS Platform[?] when the DT is being hosted there. The current security functionality of the platform is based on signed Transport Layer Security (TLS). No security considerations are currently considered for the local connection.

5 Towards the Generalizability of the Reporting Framework

With the aim of improving the generalization of the resulting reporting framework presented in **??**, we use the *Desktop Robotti DT* case study and the *Incubator DT*, presented in **??**, to approach a theoretical generalization of the proposed framework by a multi-case design, as it provides a stronger basis for theoretical generalization than a single-case design[?]. The generalization of our conceptual framework through using these two additional case studies is considered under architectural similarity[?].

To do so, we use the proposed framework to briefly report on the Desktop Robotti DT and the Incubator DT, as shown in **??** and **??** respectively. It is worth noticing that the Desktop Robotti DT is a case study of DT engineering within the robotics domain, as a mobile robot, but the Incubator DT is a case study in a very different domain (food industry). Therefore, this multi-case approach for assessing the resulting framework allows to investigate the phenomena in the context of robotics, which would support its generalizability in this domain, but also in a different context, arguing for the generalizability in other domains.

Since we are not providing a detailed description for each characteristic as we did for the flex-cell case study, we refer the readers to references[?][?][?] and the example implementation with the DTaaS Platform[**], where more information about the Desktop Robotti DT is found; and to references[?][?][?][?][?][?] and the publicly available GitHub repository**??**, where more information about the Incubator DT is found.

As a remark, although the Incubator DT does not come with an implementation on the DTaaS Platform, the specific sub-phases within the *service* phase have been reported based on those proposed for the implementation on the DTaaS.

Regarding the procedure to report these two additional case studies was as follows: The third author, who has previously worked on the Incubator DT, performed the report of this case study in a table with assistance of the first author. Similarly, the fourth author, who has previously worked on the Desktop Robotti DT, performed the report of this case study in a table with assistance of the first author. Both procedures follow the resulting framework's characteristics and their descriptions step-by-step.

Both authors agree that the description framework is easy-to-use and descriptive, which provides light validation of our description framework. The description for each of the case studies, contained in the table, took approximately 50 minutes. However, it is worth mentioning that the DTs had already been engineered and the authors have a good understanding of their case studies and the description framework, which speeds up the reporting process.

6 Discussion

This section discusses the main findings of this work in relation to DT engineering and its reporting process. It also discusses the challenges, lessons learned, and limitations for both fronts, namely, the DT engineering process of the flex-cell DT and the reporting of this case study in our proposed description framework. This discussion aims to provide insights from our experience to readers about both the engineering and reporting process to assist with conducting DT case study research.

The discussion on the case study research front intends to present the common problems that may appear in any case study, including models, synchronization, accuracy, and orchestration and reuse of components, among others. This differs from common problems in reporting any modeling or simulation research, as the DT is intended to have live connections and multiple components and services running at the same time, which can have the same or different time rates and periodicity.

Similarly, the discussion on the reporting front intends to present the problems that may appear when reporting a case study research of a technology that still lacks standardization and consensus. It also discusses the benefits of following a reporting framework as a complementary resource for conducting DT case study research as a mechanism for internal feedback.

It is also worth discussing the relevance of using a reporting framework as a guideline, and at some point as a standard, to report DT case study research. Such a guideline, in this case, our approach, encourages practitioners to report on *all* the characteristics, including those that may not be applicable or relevant for the particular scope. Hence, the case study research becomes more transparent and understandable to a broader community, enabling readers to easily identify all the features having core descriptions as a reference. This aspect also benefits authors of DT case studies, so they can identify and further elaborate on characteristics they are not aware of when going through the DT engineering process, reporting the DT case study research, or upgrading their existing DTs.

Discussion on Gaps Found

Throughout conducting this combined research for merging taxonomies to describe DT case study research and reporting the flex-cell DT case study, we found relevant gaps that require further discussion and research.

First and foremost, there is a lack of standardization for conducting and reporting DT case study research and in the DT domain in general, which is essential for the establishment and consolidation of DT technology[?]. The ISO-23247[?] and IEC-63278-1[?] are the two pioneering initiatives for this technology, but they are still framed in particular scopes, manufacturing and AAS, respectively. Since there is no standard on how or what to describe in a DT case study research, there are substantial differences between different case studies and the way they are presented, which difficults the understanding and objective comparisons. Supporting this are the multiple definitions for the DT concept[?][?] and the found six non-overlapping characteristics after the merge, showing that there is still lack of consensus in this domain.

Second, and extending from the previous gap, it is difficult to objectively assess a DT case study research without measuring its benefits/improvements to a given process or physical system, that is, assessing how complete, mature, and complex a DT approach is, including its strengths and weaknesses in a case-independent manner. Such an assessment method can help authors to identify strengths and weaknesses of their case studies, and readers to quickly spot whether a DT is relevant for their interests and what they are after. A consensual or standardized reporting framework is a good starting point for creating an assessment framework, where each characteristic can be weighed based on certain criteria, and thus, the assessment framework can be easily attachable to the reporting framework.

Finally, there are multiple cross-cutting characteristics that may be relevant to report on DT case study research, such as the ones presented, namely, data ownership and privacy, standardization, and security and safety considerations. These characteristics may be highly relevant to be reported on industrial and commercial DT applications, where ethical and technical aspects matter for the long-term support. While

[**]https://into-cps-association.github.io/DTaaS/version0.4/user/examples/drobotti-rmqfmu/index.html

Table 7. Brief description of the Desktop Robotti DT case study with our proposed DT description framework.

Merged Characteristic	**Desktop Robotti case study**
MC1: System-under-Study	Small prototype of a field (agricultural) robot. A mobile robot.
MC2: Physical acting components	Motors for each wheel.
MC3: Physical sensing components	RPLidar A1, IMU (Inertial Management Unit), and wheel encoders.
MC4: Physical-to-Virtual Interaction	The PT sends location data at a periodic basis over RabbitMQ.
MC5: Virtual-to-Physical Interaction	The DT sends emergency stops and parameter updates (e.g. by constraining the speed).
MC6: Digital Twin Services	The Desktop Robotti DT provides services for *monitoring: distance-to-obstacle*, *collision avoidance for two cooperative Desktop Robottis*, *Parallel operation: comparing real and predicted location data*, *Fault-injection with hardware in the loop*, and *Runtime model swapping*: swapping FMUs during operation to extend functionality.
MC7: Twinning Time-scale	The DT-to-PT synchronization is on a periodic basis. The DT supports best-effort real-time.
MC8: Multiplicities	When deployed in a cooperative setting of two Desktop Robottis, there is one DT for both, so there is no multiplicities.
MC9: Life-cycle stages	The DT provides services within the *service* phase. In this phase, it supports creating, executing, and terminating. The DT covers the system as designed.
MC10: Digital Twin Models and Data	There are a kinematic model of the robot, specifically a bicycle model with the virtual wheels placed a the center of the front and rear axles, and an actuation model for the DC motors expressed as a first-order system. The data of interest are related to robot positioning and velocity.
MC11: Tooling and Enablers	RabbitMQ and the Robot Operating System (ROS)[?] for communication and interfacing. Maestro and RMQFMU to run the co-simulation scenarios. The Model Swap and Fault Injection plugins to run the DT services related to fault injection[?] and runtime model swapping[?]. RViz for visualization.
MC12: Digital Twin Constellation	The orchestration of the system-as-a-whole, including the models and data, tools and enablers, services, and physical-to-virtual and virtual-to-physical interaction is defined. The constellation also describes how the DT behaves in the multiple scenarios for the provided services. Additionally, some of the components are initialized from configuration files and scripts.
MC13: Twinning Process and Digital Twin Evolution	The DT was engineered based on an existing prototype of a large-scale agricultural robot (Robotti[?]) with a subsequent engineering approach. The evolution presents five milestones: the setup of the parallel operation, enhancement with fault-injection, time discrepancy detection (between real and simulated/DT time), runtime model-swapping, collision zone detection for a fleet of DRs.
MC14: Fidelity and Validity Considerations	The DT contains medium to high fidelity models. The Desktop Robotti DT has been experimentally validated.
MC15: Digital Twin Technical Connection	The PT-to-DT connection is done over Wi-Fi on a laptop using RabbitMQ and ROS.
MC16: Digital Twin Hosting/Deployment	The Desktop Robotti DT can be deployed locally on a LAN or on the DTaaS platform in the cloud.
MC17: Insights and Decision Making	The DT can provide insights into visualization of the robot's location through RViz coupled to the PT (as a DS) or to the DT.
MC18: Horizontal Integration	There is horizontal integration with the Desktop Robotti PT and infrastructure services of the DTaaS Platform. The Desktop Robotti DT is able to exchange information with other information systems not limited to other DTs over RabbitMQ.
MC19: Data Ownership and Privacy	Not considered.
MC20: Standardization	Communication is carried out using AMQP standard via RabbitMQ. Behavioral models conform with the FMI standard version 2.
MC21: Security and Safety Considerations	Security aspects inherited from the DTaaS TLS. Safety aspects regarding collision zones avoidance.

our scope did not focus on exploring these cross-cutting characteristics, we encourage further research on these, so they can be integrated to the reporting framework presented here as optional characteristics.

Challenges and Lessons Learned: Case Study

Along the development of the flex-cell case study, we identified some relevant challenges and lessons learned that are worth mentioning and can help practitioners who are interested in the DT engineering process for their case studies.

The most challenging aspect of this case study so far is related to the models and their integration in a cooperative-synchronized setting. This is due to having multiple modeling approaches for robotic arms for different components, e.g., kinematics, dynamics, visualization, programming, etc. Therefore, it is difficult to integrate all of them into a unique DT if the scope is a bit loose. Thus, it is better to narrow down the scope according to the applications that are needed, and so, choose the appropriate models for that scope. In other words, one should start from the expected services which come from the requirements, and from there, plan the modeling and orchestration tasks accordingly. The

Table 8. Brief description of the Incubator DT case study with our proposed DT description framework.

Merged Characteristic	**Incubator case study**
MC1: System-under-Study	A Styrofoam box containing a lid, a heating element, and fan, controlled by a Raspberry Pi, for incubating tempeh.
MC2: Physical acting components	Heating element and fan.
MC3: Physical sensing components	3 temperature sensors (2 inside and 1 outside).
MC4: Physical-to-Virtual Interaction	The controller in the PT sends sensor and actuator data on a periodic basis over RabbitMQ
MC5: Virtual-to-Physical Interaction	The DT sends new parameters of the controller, or desired temperature, to the controller in the PT.
MC6: Digital Twin Services	*Heater state estimation*, *real-time (and historical) visualization*, *anomaly detection*, *what-if simulations*, *reconfiguration according to state of the lid*, *controller parameters optimization*.
MC7: Twinning Time-scale	The DT-to-PT synchronization occurs every time the PT sends a message to the DT (on a periodic basis).
MC8: Multiplicities	The current implementation has no multiplicities, however, it is possible to deploy multiple DTs for the same PT as proposed in[?].
MC9: Life-cycle stages	The DT supports the *design* and the *service* phases. In the *service* phase, it supports creating, executing, saving, analyzing, evolving, and terminating.
MC10: Digital Twin Models and Data	There are plant models (2-parameter and 4-parameter ordinary different equations, and artificial neural network models for plant and state estimator), controller models (state machine), environment models (room temperature prediction), CAD models for 3D visualization, and various couplings between the plant, controller, and environment, models. The data of interest are related to temperature, actuation (on/off state of the actuators), and state of the controller.
MC11: Tooling and Enablers	RabbitMQ for communication, InfluxDB for storing timeseries data, Docker for containerization, HOCON for storing configuration files, Godot for 3D visualization, and various Python libraries (e.g., python-control, SciPy).
MC12: Digital Twin Constellation	The orchestration of the system-as-a-whole is carried out by micro-services. These micro-services set up the multiple components from configuration files, including the models and data, tools and enablers, services, and physical-to-virtual and virtual-to-physical interaction. It is possible to leave aside some of the micro-services when initializing the DT for testing purposes.
MC13: Twinning Process and Digital Twin Evolution	The DT was engineered based on a joint engineering approach. For the evolution, 10 milestones have been defined: identifying the physics for the PT, building the plant models with, characterizing the heating power, building the first physical prototype, experimentally refining the parameters for the plant model, creating the controller model, the deploying the controller code into the physical controller, deploying the visualization service, providing services for state estimation and anomaly detection, and providing the service for optimizing the control policy.
MC14: Fidelity and Validity Considerations	The models have been calibrated against experimental data and the predictive accuracy of the best model is within 2°C. The models have been validated in a controlled environment.
MC15: Digital Twin Technical Connection	The PT-to-DT connection is done over Wi-Fi on a laptop using RabbitMQ.
MC16: Digital Twin Hosting/Deployment	The Incubator DT is deployed locally on a LAN.
MC17: Insights and Decision Making	The DT can provide visualization of the current state and historical data, what-if analysis for future behavior under different controller configurations, and control policy updates to the PT.
MC18: Horizontal Integration	There is horizontal integration with the micro-services of the Incubator DT. The DT is able to exchange information with other information systems over RabbitMQ.
MC19: Data Ownership and Privacy	Datasets have been provided online to the public. No privacy-related data are stored.
MC20: Standardization	Communication is carried out using AMQP standard via RabbitMQ. Behavioral models have been produced following the FMI standard version 2.
MC21: Security and Safety Considerations	Communication can be TLS encripted through the RabbitMQ broker. The physical controller counts with a safety consideration that turns off system if the temperature read is above 60°C or if the network connection is unstable.

current scope of the flex-cell is related to positioning, but it may be broadened to cover other relevant aspects.

The details of the challenges found are as follows:

Model accuracy The higher the accuracy in the models behind the DT, the better the DT can represent and interact with the PT[?]. However, creating accurate models can be tricky, especially because it may need time and several iterations between the DT and PT to fine-tune them. A high-accuracy model can also slow down the computation process, therefore, the accuracy level needs to be aligned with the expectations and scope.

Reusable/generalizable modules for optimization It is important that the DT has some direct actions based on improved plans and inferred sequences and some indirect actions on the PT in terms of optimization, improvement, self-adaption, etc. However, these kinds of services are usually designed case-specific and are hard to reuse, making the incorporation of them a challenging task.

Synchronous and asynchronous messaging Even though bidirectional communication between PT and DT is a must, it can be complex to administrate synchronous and asynchronous messaging, especially when the DTs are created without the support of any existing Internet of Things or DT frameworks that already provide a communication middleware. Even so, when processing asynchronous events, it is challenging to manage how the DT responds/reacts to them in a smart and easy-to-deploy way.

Formally verifying the DT for its scope It can be the case that we know exactly what the DT is supposed to do, but the software, models, and their integration, are not formally verified. Formal verification can add extra overhead and extra engineering effort, but it may be worth it if the scope requires additional safety.

Environment and uncertainty These aspects play an important role in the DT development and execution. The DT engineering is much easier when the environment is controlled, such as the case of the flex-cell system, where the uncertainty is considerably reduced, and thus, not considered. In other cases, where the DT is to run in unknown environments, the uncertainty increases and needs to be approached as a critical factor of the DT.

As for lessons learned, these are as follows:

Multiplicity Although the concept of multiplicity in DTs is controversial[?], we find it better to use multiple DTs for representing multiple components instead of a big DT to cover everything. It may also be useful in the case a physical counterpart has more than one DT, each featuring different models that are comparable.

Orchestration Creating and running DTs is a task that requires the orchestration of several components, namely, models, enablers, and services, i.e., the *DT constellation*[?]. This also involves the use of infrastructure services related to communication that are required to bind the PT to the DT and vice-versa. It is suitable to have strategies for the orchestration of these multiple components to lift the system from configuration files that are generalizable for different DT use cases.

Simulation models Although there is no consensus or standards providing the guidelines for the right models to be used in a DT, simulation models that support coupled and synchronized behavior are highly recommended. This can help the DT in the dimension of horizontal integration, being able to estimate, predict, or compute the behavior of the PT and its relationships with the environment or external independent systems.

Challenges and Lessons Learned: Reporting

Similarly, we also identified relevant challenges and lessons learned regarding the reporting process of a DT case study. This is especially evident for some characteristics that are difficult to describe in a textual description. However, some kind of exemplification helps to complement the answers. On the other hand, having a guideline for the reporting process assists practitioners in their writing, and makes the report understandable to a broader audience. Moreover, the more complete the set of characteristics to report on, the more clear the report becomes, but the more challenging it becomes to write.

The unified characteristic framework proposed in this paper is a strong guideline for reporting a DT case study, coming from the merge of three very complete description frameworks. Therefore, we encourage readers to adopt these systematic reporting principles presented here to report their DT case study research.

The details of the challenges found are as follows:

Describing characteristics in words There are some characteristics that are difficult to explain in words. The explanation can be accompanied by exemplification, such as diagrams, figures, and results, which can provide more meaning and make the descriptions clearer.

Reporting all the characteristics This point can be complex for some case studies, where there is no clear view of all aspects of the DTs. Nevertheless, these characteristics come from generalized requirements for DTs, which may provide a more complete and reproducible description of the DT and its engineering process.

Common understanding Since the DT is a modern concept and is approached by audiences/communities of multi-disciplinary knowledge background, it is difficult to have a common understanding for some terminology, which may differ from one technical society to another. There is also lack of standardization[??], where the ISO-23247[?] and the IEC-63278-1[?] are pioneer standards for DTs.

As for lessons learned, these are as follows:

Trivial vs non-trivial characteristics There are some characteristics that can be answered straightforwardly, such as describing inputs and outputs, whereas there are others that require more elaboration, such as describing the DT constellation. This provides a sense of where should the report (and also the DT implementation) place more effort. A clearer description of the non-trivial characteristics provides better traceability of the DT engineering process, and thus, the process can be more easily reproducible by practitioners.

Using a reporting framework for internal feedback Most DT case studies are reported on individual requirements, and therefore, they dismiss reporting on dimensions that can be relevant to other practitioners. Using a reporting framework helps to overcome this problem. A reporting framework can also be used for internal feedback during the engineering process to find out the maturity level of the DT and which of its dimensions require further work.

Limitations of This Study

The primarily identified limitations of this case study are in relation to the case study and the reporting process.

Case Study

Internal validity

Different controllers The controllers of the simulated robots may slightly differ from how the real controllers of the physical robots behave since the source of the real controllers is closed. It also applies to the visualized motions in Unity since the simulation engine has its own settings, which are not necessarily identical to the physical controllers. A potential solution for this problem is using a more accurate representation of the real controllers in both the simulation and visualization.

Model fidelity and validation Although the models used in the flex-cell DT have been experimentally validated to be sufficient for robot positioning, these still introduce error for accurate movements. These models do not include time, and therefore, timing does not necessarily correspond to how the real robots will move. The parameter for motion timing has a precision of 0.05 seconds and can be updated online, which has been experimentally validated to be sufficient for the flex-cell DT case study. Nevertheless, upgrading the models to include dynamics where time is considered would improve this aspect. Additionally, there are no behavioral models for the grippers.

Observations The kuka lbr iiwa 7 does not provide certain observations for speed and acceleration, which increases the complexity of including a calibrated dynamics model for this robot. Similarly, the number of observations obtained from the grippers is low, which limits the variables where the DT can act/react over.

Delays The execution of commands in the real flex-cell can have some latency since it is prone to network and computation delays which are not further analyzed in this study.

External validity

Use of the running example for external validity Although there are publicly available examples of the flex-cell DT and the Desktop Robotti DT on the DTaaS platform, readers may have limited access to use our hosted version of the DTaaS Platform and its services to run experiments for external validity. Readers can refer to the open-source code of the DTaaS[††] which they can use to set up their own environment and services as needed.

Reporting

Internal validity

Bias in the report The case studies presented in this work have been executed and reported from the empirical and personal experiences of the authors, which could lead to bias.

Selection of reference reporting frameworks This study selected three reference frameworks for the merging process based on the rationale described in **??**, which may have led to the exclusion of other relevant dimensions when reporting DT case studies. This work is not performing a systematic tertiary survey of other frameworks since we believe the field is not yet sufficiently mature for this. Thus, the focus of this work is more on the practical explanation of DTs (in particular our DT cases) rather than the evaluation of the academic literature side, which finds a balance between idealization and practice[?]. This selection is also a trade-off because the more reference frameworks we consider, the more difficult the merging becomes.

Merging process Although we followed the methodology by Kundisch et al.[?] for the merging process, the result may still be prone to different interpretation since it comes from our subjective perception. This potential limitation is mitigated by the fact that we have extensive experience in DT implementation and description. We tried to reuse terms and descriptions from the three reference frameworks and triangulate among them.

External validity

Open data There are no open data to compare the results presented in this report. However, the reader can refer to the publicly available examples for extra information on the technical details of the case studies.

Generalizability The generalizability of the reporting frameworks was approached by a multi-case design of three case studies, two within the same domain and one from a different domain. This fact may limit the generalizability of the framework to be only applicable to the domains of the presented case studies or domains with architectural similarity.

7 Concluding Remarks

This paper presents an approach to unify the characteristics to be reported in DT case study research, by systematically merging three description frameworks for DTs. As a result of the merging process, we have identified eighteen fundamental characteristics and three cross-cutting characteristics that are relevant to be reported on DT case studies. This approach is proposed as a solution for a reporting framework in a domain that is lacking standardization and guidelines in this regard. Therefore, we encourage the DT community to adopt this reporting mechanism to report their DT case studies.

The main outcomes of this work are in relation to i) a reporting framework for DT case studies that encourages authors to report and elaborate on a set of fundamental and consistent characteristics of DTs, so the reports are more transparent and understandable to readers; ii) the use of a sufficiently complex DT case study in cooperative robotic arms to showcase the framework, reporting on the 18 fundamental characteristics and two cross-cutting characteristics (since one cross-cutting characteristic is not considered); iii) the use of two additional case studies, one within robotics and one within the food industry, which are briefly showcased to elaborate on the theoretical generalizability of the framework; and iv) the provision of the main gaps found, challenges, and lessons learned that arose throughout conducting this research in relation to the DT engineering and reporting processes.

Future Work

Both the implementation and reporting fronts still face technical challenges and limitations that are worth addressing as potential research directions in the DT domain.

First, regarding the case study, it is continuing to evolve, mostly in terms of applications and services, such as collision detection, optimization of assembly, and flexible and reconfigurable sequences. The ongoing work of the flex-cell DT focuses on the generation of sequences for flexible manufacturing in a cooperative setting by integrating the

[††] https://github.com/INTO-CPS-Association/DTaaS

skill-based programming approach?. Another future work is in relation to integrating different models with a similar scope in the flex-cell DT with the purpose of analyzing the advantages of including control software, dynamics, and model verification that are part of the *RoboStar* tools?.

Second, regarding the reporting framework, we foresee a potential to extend the reporting framework with a case-independent mechanism for assessing DT case study research. The framework can also be extended to support the identification of supplementary material, such as figures, diagrams, and so on, in characteristics that require it. Additionally, as we did not perform any systematic review on existing frameworks for reporting DT, we call on the research community to go farther, perform these surveys and elaborate on a standardized reporting framework for DTs.

Acknowledgements

This work has been partially funded by the Ringkøbing-Skjern Municipality, Denmark, under the Framework Collaboration Agreement for Aarhus University Digital Transformation Lab-Skjern. The authors are grateful to the Poul Due Jensen Foundation, which has supported the establishment of the Center for Digital Twin Technology at Aarhus University. The authors also sincerely thank Dr. Istvan David (McMaster University) for his insightful recommendations concerning our methodology, and Dr. Benoit Combemale (University of Rennes) for his suggestion on representing the communication enablers in the constellation figure.

Santiago Gil is a PhD candidate in the Department of Electrical and Computer Engineering at Aarhus University, Denmark. He completed his Bachelor's and Master's degrees in Colombia and moved to Denmark to pursue his PhD studies. His research interests include digital twins, cyber-physical systems, Internet of Things, and digital transformation.

Bentley James Oakes is an Assistant Professor in the Department of Computer Engineering and Software Engineering at Polytechnique Montréal, Canada. His research interests include digital twins, verification of cyber-physical systems, model-driven engineering, knowledge representation, and model transformations. Please visit https://bentleyjoakes.github.io/ for more information.

Cláudio Gomes is an Assistant Professor at the Department of Electrical and Computer Engineering at Aarhus University, Denmark. His research interests include co-simulation and digital twin engineering. His email address is claudio.gomes@ece.au.dk.

Mirgita Frasheri conducted her PhD studies at Mälardalen University, Sweden, on the topic of adaptive autonomy. After her PhD, Mirgita continued as a Postdoc at Aarhus University, Denmark, where she is currently an Assistant Professor, and conducts research on Digital Twins, with robotics as the main application. Her research interests include multi-agent systems, autonomous systems, AI ethics, digital twins, and co-simulation.

Peter Gorm Larsen is a professor and deputy-head of section at the Department for Electrical and Computer Engineering at Aarhus University. He currently leads the AU DIGIT Centre, the AU Centre for Digital Twins, as well as the research group for Cyber-Physical Systems. His research areas range from formal methods over cyber-physical systems to digital twins.