Received: 1 August 2022 | Revised: 6 November 2023 | Accepted: 4 December 2023

DOI: 10.1002/spe.3305

SURVEY ARTICLE

WILEY

Survey on open-source digital twin frameworks–A case study approach

Santiago Gil | Peter H. Mikkelsen | Cláudio Gomes | Peter G. Larsen

Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark

Correspondence
Santiago Gil, Department of Electrical and Computer Engineering, Aarhus University, Finlandsgade 22, 8200 Aarhus, Denmark.
Email: sgil@ece.au.dk

Funding information
Ringkøbing-Skjern Municipality under the Framework Collaboration Agreement for Aarhus University Digital Transformation Lab-Skjern

Abstract
Digital twin (DT) technology has been a topic with academic and industrial coverage in recent years. DTs are intended to be a virtual high-fidelity representation of a physical counterpart. Its complex nature requires several components to create and run a DT, and that is why many DT frameworks have been proposed in the literature. There are also many surveys of DTs, but none that is bottom-up with concrete examples and focused on open-source software. This survey analyzes 14 open-source DT frameworks in 10 different dimensions, which are then categorized in six different groups according to their modeling and technological domain, to present the reader different options for creating and managing DT applications, and to understand potential combinations, uses, and limitations of the tools. It also presents a case study with five of the explored DT frameworks, describing the process on how the DT is set up and comparing their capabilities based on the services to be provided by the DT. Finally, it discusses advantages and limitations of the tools according to domain, requirements, and scope, relevant aspects regarding built-in simulations and data analytics, theory-to-practice transition, and advantages/disadvantages of using open-source software instead of commercial. Main limitations of the study due to its narrow niche, conclusions, and opportunities for future research regarding the potential room for improvement in terms of *out-of-the-box* features and services for DTs, are also shown.

KEYWORDS
cyber-physical system, digital twin, open-source software, software framework

1 | INTRODUCTION

Digital twin (DT) is the concept of a digital replica of a physical entity (PE), also called physical twin (PT), that adapts to it through the PE's service life. Such a DT will provide opportunities to simulate scenarios that would otherwise be too

Abbreviations: AAS, Asset Administration Shell; AML, automation ML; AMQP, advanced message queuing protocol; API, application program interface; CPS, cyber-physical system; DS, digital shadow; DT, digital twin; DTCC, Digital Twin Cities Centre; DTDL, Digital Twin Definition Language; FMI, functional mock-up interface; FMU, functional mock-up unit; GUI, graphical user interface; HTTP, hypertext transfer protocol; IoT, internet of things; MQTT, MQ telemetry transport; NSW, New South Wales; PE, physical entity; PT, physical twin; SDK, Software Development Kit; SME, small and medium enterprise; VAB, virtual automation bus.

expensive, create models based on data aggregation from PEs in operation, provide a reference for detecting changes, and so on. The concept originally coined by Grieves,[1] has increased attention in both academic and industrial communities, as shown in Figure 1. The increase in both papers and patents has increased exponentially, except during the period of Covid-19. The DT concept has been adopted in domains such as aerospace, manufacturing, and civil infrastructure, where it is crucial to predict failures and continuously refine the ability to do so. In manufacturing, the concept is also used as a way to capture cross-domain knowledge across product life-cycle stages in order to provide insights, which would otherwise easily be lost in the individual domains or stages.[2]

Recent technological advancements, such as high-performance edge computing, high-bandwidth cellular networks and high-performance streaming and storage facilities are enablers for DT technologies. They allow the DT to be updated by high-bandwidth, low-latency data, that enables very accurate modeling. The same technological advancements has also given rise to the concept of internet of things (IoT). This concept has a wider perspective, but share many similarities.

A lot of research is being made to suggest DT concepts, architectures, and application domains. Recently there has been an explosion of surveys of the DT concept.[3–20] Among these, it is worth highlighting Kritzinger et al.,[4] where the term digital shadow (DS) was introduced, and Macchi et al.[20] where multiple cases of abuse of the term DT are described. Common to all these surveys is that they try to cover a wide portion of the state of the art, opting for breadth rather than depth. Unfortunately, this makes them less suited for practitioners who wish to set up a DT on their premises. Though a relatively new idea, it also sees consolidation and is now being standardized in the manufacturing domain, through the ISO-23247 standard[21] and the upcoming IEC-63278 standard.

On the tool side, several software DT frameworks are emerging, which were not considered in the above surveys. They are primarily built around IoT technologies, but typically adapted to a specific domain. There are both, commercial and open-source DT frameworks, but this survey focuses only on open-source contributions. It is relevant to mention that the community and the open-source contributions supplied by persons and organizations have driven the DT technology and its adoption under common foundations.[22] The rational of this survey intends to cover the feasibility for small and medium enterprises (SMEs) to get on board actively in the community of DTs. This way, SMEs with low budget can start developing in-house DT projects in case they cannot afford a complete commercial DT solution.

In order to perform an in-depth survey of open-source DT frameworks, we selected the case study proposed in Reference 23 and detailed in Reference 24, that is representative of a cyber-physical system (CPS) for evaluating DT frameworks. We then collected 14 DT frameworks and analyzed their documentation. From those, we selected five based on their maturity, and conducted our case study with those five DT frameworks, while contacting the corresponding authors to clarify features. We have selected this case study for its simplicity, minimal background requirements, ability to allow researchers to perform modeling and simulation at multiple levels of fidelity, and ability to accept multiple variants of DTs.[25]

With the findings and analyses of this survey we show practitioners a practical guide through different alternatives for setting up DTs in different domains. Then, the reader can discover which option fits best according to domain, requirements, and scope. Advantages and disadvantages are also discussed as well as identified limitations and challenges are stated.

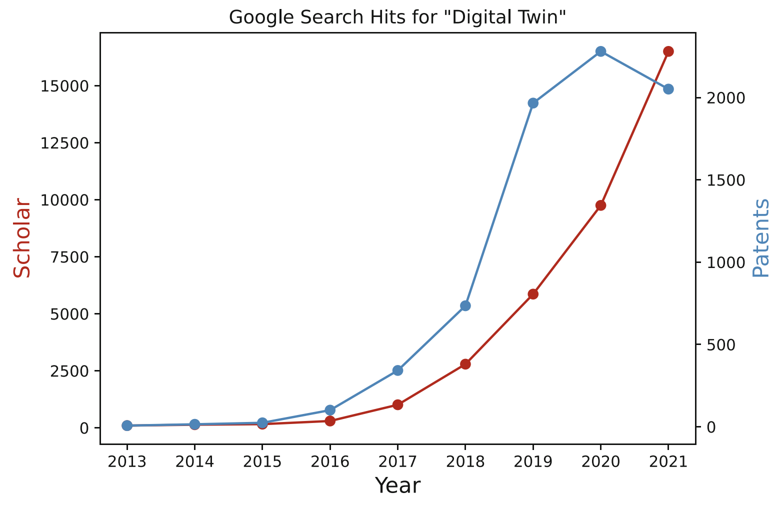


FIGURE 1 Google search on digital twin (DT).

The remainder of this paper is as follows:

1. Section 2 presents the methodology, criteria, and analysis on the explored DT frameworks.
2. Section 3 Presents a bottom-up study of selected candidates and reports how they can be leveraged to implement a specific DT case.
3. Section 4 discusses the findings, challenges, limitations, and opportunities for improvement in relation to tools and the theory-to-practice transition for DTs.
4. Section 5 presents a brief state of the art on surveys, reviews, and approaches for DT frameworks and tools.
5. Finally, section 6 presents the main conclusions and future work opportunities.

2 | MATERIALS AND METHODS

2.1 | Methodology

For this bottom-up survey, we selected an appropriate methodology to select, analyze, and explore the DT frameworks and implement a demonstration for the readers with a simple case study. The proposed methodology implements the following steps:

1. Collect candidate open-source DT frameworks (further described in section 2.1.1).
2. Discard the frameworks that could not be explored and evaluated because of lack of documentation for installation and setup.
3. Classify them according to the taxonomy defined in section 2.2. This is described from sections 2.3.1 to 2.3.14 for each framework respectively.
4. Propose the case study, described in section 3.
5. Exclude the ones that lacked documentation, enough releases, or complete open-source environment or the domain did not fit to the proposed case study.
6. Implement the DT case study (selected for its simplicity) using the remaining ones, and give a more detailed report of the experience. This is described from sections 3.1 to 3.5 for each of the selected frameworks respectively.
7. Analyze the capabilities of each of the selected frameworks in relation to how well they suit the requirements of the proposed case study. This is described in section 3.6.

2.1.1 | Candidate collection

For searching and collecting the initial candidates, we used three different channels, namely, (i) Google Search engine, (ii) GitHub repository and topic explorer, and (iii) Google Scholar. The search included tools up to July 2022. The searching keywords were associated to *Digital Twin*, *Digital Twin Framework*, *Digital Twin Platform*, *Digital Twin Tool*, *Open-source Digital Twin Framework*, *Open-source Digital Twin Platform*, and *Open-source Digital Twin Tool*.

The sample ended with a number of 14 candidates that complied with our filtering criteria. A summary of the filtering process is shown in Figure 2. The DT frameworks that are analyzed in this survey are shown in Table 1.

Proceeding with the details of the filtering process (and considering the notation ***Fx*** of Figure 2), we first filtered the commercial tools (mostly appearing on Google Search engine) and kept the ones claiming to be an open-source Digital Twin framework/platform with a valid public repository or webpage (**F1**). Then, from the results on Google Scholar, we inspected a total of 47 papers, from which we focused on the papers claiming to have developed or reviewed an *open-source Digital Twin framework/platform* with a valid and public repository (**F2**). Finally, from GitHub explorer we only considered the repositories claiming to be an *open-source Digital Twin framework/platform* rather than DT implementations or complementary tools (**F3**).

A quick search on GitHub with the topic *digital-twin* retrieves more than 100 repositories, however, many of them are DT implementations, undocumented projects, tools that are complementary but not specific to DTs, or the maturity of the documentation available is not enough to be used.

The general filtering criteria applied to select a final candidate from the pool of candidates were (1) the candidate claims to be an *open-source Digital Twin framework/platform* even though in reality it only covers *digital models*, *digital*

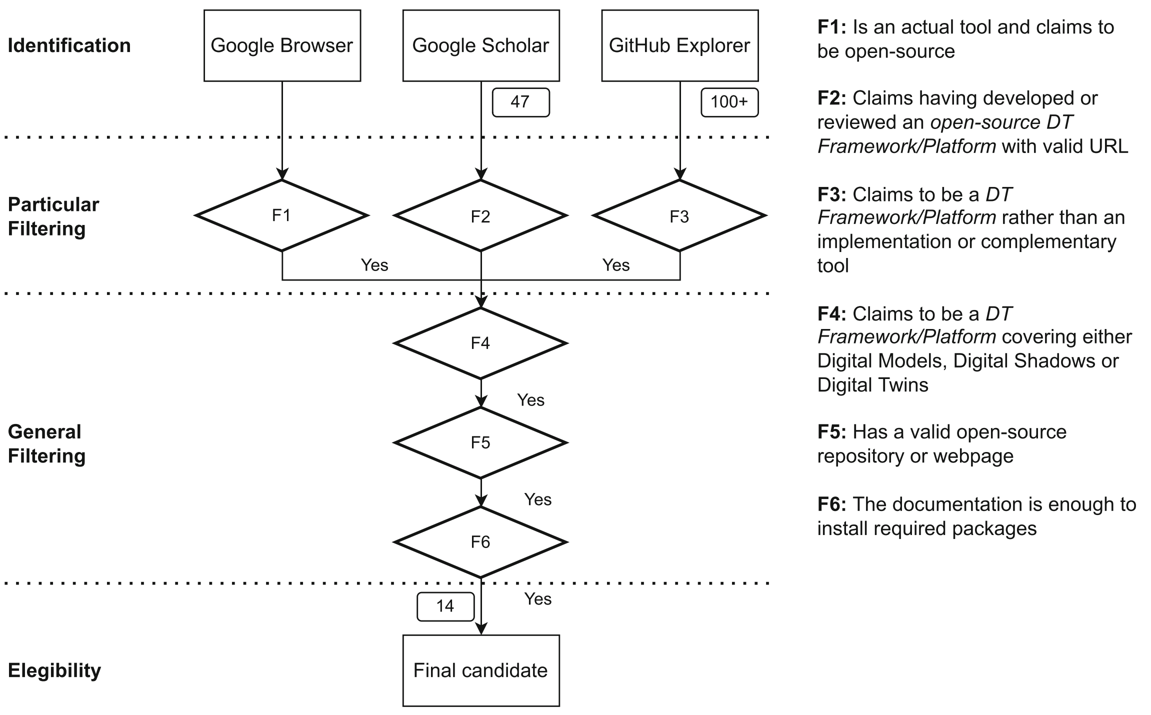

Identification
Google Browser
Google Scholar
GitHub Explorer
47
100+
Particular Filtering
F1
F2
F3
Yes
Yes
F4
Yes
General Filtering
F5
Yes
F6
14
Yes
Elegibility
Final candidate
F1: Is an actual tool and claims to be open-source
F2: Claims having developed or reviewed an open-source DT Framework/Platform with valid URL
F3: Claims to be a DT Framework/Platform rather than an implementation or complementary tool
F4: Claims to be a DT Framework/Platform covering either Digital Models, Digital Shadows or Digital Twins
F5: Has a valid open-source repository or webpage
F6: The documentation is enough to install required packages

FIGURE 2 Filtering process for the collected candidates.

TABLE 1 List of analyzed DT frameworks.

Name	Author/Organization	Link
Eclipse Ditto	Eclipse Foundation & Bosch	https://www.eclipse.org/ditto/
Equinox	Murat Artim	https://github.com/muratartim/Equinox
AASX Package Explorer	Industrial Digital Twin Association (IDTA)	https://github.com/admin-shell-io/aasx-package-explorer
PYI40AAS	RWTH Aachen	https://git.rwth-aachen.de/acplt/pyi40aas
SAP I4.0 AAS	SAP	https://github.com/SAP/i40-aas
Eclipse Basyx	Eclipse / Bosch / Fraunhofer Institute	https://projects.eclipse.org/projects/technology.basyx
NOVA AAS	NOVA School of Science and Technology (NOVA University Lisbon)	https://gitlab.com/gidouninova/novaas
CPS-Twinning	SBA Research	https://github.com/sbaresearch/cps-twinning
Twined	Octue Ltd	https://github.com/octue/twined
Azure Digital Twins Definition Language (DTDL)	Microsoft Azure	https://github.com/Azure/opendigitaltwins-dtdl
iTwin.js	Bentley Systems, Incorporated	https://www.itwinjs.org/
Digital Twin Cities Centre Platform (DTCC)	Chalmers University of Technology	https://gitlab.com/dtcc-platform
TerriaJS (NSW Digital Twin implementation)	New South Wales State, Australia	https://nsw.digitaltwin.terria.io/about.html, https://github.com/TerriaJS/terriajs
INTO-CPS Co-simulation Framework	INTO-CPS Association	https://into-cps-association.readthedocs.io/en/latest/

shadows, or actual *digital twins* (**F4**), (2) the candidate has a valid open-source repository or webpage (**F5**), and (3) the documentation of the candidate is mature enough to at least install the required packages (**F6**).

2.2 | Criteria to analyze the DT frameworks

The proposed criteria is designed based on inputs collected from analyzing some contributions in the field of DTs,[26,27] to classify whether the tool provides capabilities, such as bi-directional integration, interoperability, integrated intelligence and analytics, and the community that sustains the tool or framework. Nevertheless, some of the points were abstract and subjective to grade the tools, and thus, an extension was performed according to common notation of the concept of DT[28–31] and informed essential characteristics for DT experience reports, as shown in Bentley et al.[32]

Additionally, it is proposed to assign levels of compliance instead of classifying with binary attributes. This way, it is possible to discretize categorical qualitative variables into numeric ones. Therefore, the levels intend to cover from non-compliant to fully-compliant. In different words, *level 0* means non-compliant or uncovered by the framework, and *level max*, where *max* can have a different number depending on the criterion, means fully-compliant or fully-covered by the framework. This numeric handling is with the purpose of making the attributes clearer for subsequent comparison between frameworks.

Some of the criteria are inspired by the ISO 23247:2021–*Automation systems and integration–Digital twin framework for manufacturing*.[21]

Hence, the proposed set of criteria to grade the tools is as follows:

1. Communication. We use the categorization at the level of integration of DTs given by Kritzinger et al.[4]
   a. Level 0 (None)–Digital Model support only.
   b. Level 1 (PT2DT)–Digital Shadow support only.
   c. Level 2 (DT2PT)–Generator support only.
   d. Level 3 (PT2DT2PT)–Full Digital Twin support.
2. Storage. DTs need to store data in various formats, methods, and locations:[33]
   a. The tool supports data storage mechanisms.
3. Support for Analytics. DTs should provide some services in relation to optimization, decision making, task allocation, and so forth,[32,34] where some of these services should come from simulation, analysis, artificial intelligence, or computation models:[33]
   a. Level 0–No support for analytics.
   b. Level 1–application program interface (API) available for deploying analytics.
   c. Level 2–Automated analytics performed from metadata. (e.g., identifying an alarm or failure state for a given variable).
   d. Level 3–Learning metadata for given analytics. (i.e., understanding the metadata of models to provide a specific analysis of the PE, such as a failure state of any variable).
   e. Level 4–Learns analytics and all needed metadata. (i.e., understanding the metadata of models to provide an inferred analysis that is not requested by the user but learned by the DT itself).
4. Compositionality. Composable DTs are essential for their effective reuse and the reuse of its sub-components.[34] Building DTs from composable DTs introduces the term *Multiplicity*.[32,34] If not fully composable, hierarchical aggregation of DTs[35] can still be beneficial:
   a. The tool support compositionality of DTs. If two DTs are composed, will the composition behave as a single DT?
   b. If composition is not supported, is hierarchical aggregation of DTs and their sub-components feasible?
5. Support for Physical Interventions. Continuous engineering and fault-tolerance is highly relevant in software-based systems such as DTs.[36] Can the tool still be alive and running if the PE enters in an intervention or failure state?:
   a. None.
   b. Expected.
   c. Unexpected.
6. Scalability. Since IoT being one of the core technologies of DTs, it is important to study the scalability of the DTs in relation to their platforms:[37]
   a. The tools support adding an increasing number of DT instances over time while keeping an efficient computation performance.

7. Standardization. The framework, tool, or implementation follows and complies with one or more standards that allow the interpretation of its components with other tools or entities at different levels.
   a. At the communication level–Standard(s) for communication protocols or interfaces.
   b. At the metadata level–Standard(s) for data representations or interfaces.
   c. At the behavior level–Standard(s) for behavioral or functional modules or interfaces.
8. Steps to Configure/Reproducibility. DTs are usually based on *constellations* of models, enablers, and usages,[32] which may require the integration of internal or external services and infrastructure, and a proper guidance:
   a. The tool describes the steps taken to setup the DT for a given PE. The tool provides a clear guidance from installation to setup to implementing and running DTs
9. Interoperability. The tool provides interoperable communication interfaces to exchange data at different levels. According to Reference 38, interoperability is a current challenge of DT engineering:
   a. None.
   b. Syntactic–The interfaces offer a structured message exchange mechanism or skeleton.
   c. Semantic–The interfaces offer a structured message exchange mechanism that can be interpreted under a common schema.
10. Community Support. The tool is supported by a certain reliability and foundation for the long-term usage:
   a. Who is the responsible for the tool development? (Individual, Research group, Company, Association or Industrial Consortium).
   b. Number of releases/commits from its first appearance.
   c. Summary of where the documentation is.

2.2.1 | Rationale

It is important to highlight that some of the criteria are inspired by the ISO 23247:2021–*Automation systems and integration–Digital twin framework for manufacturing*,[21] which provides a consensual set of services for DT frameworks regarding communication, data acquisition, data analysis, simulation, among other features.

This standard was chosen due to it being a pioneer of standards for DT technology that may provide harmonization within the domain,[34] a domain where there are still challenges in relation to standardization,[29,38] and it is hopefully transferable to other fields and domains.

2.3 | Comparison of DT frameworks

This subsection presents the outcomes of the analysis of this DT framework comparison where the 14 frameworks were analyzed systematically according to the criteria proposed above. Tables 2 and 3 present a summary of the analysis of the frameworks and further details are presented from sections 2.3.1 to 2.3.14, each containing a description and extended information to the tables.

2.3.1 | Eclipse ditto

This framework provides a full set of tools to adapt DT in the context of Industrial IoT. It provides the infrastructure/cluster for communication and modeling of assets as things, as well as other microservices for connectivity. It is possible to adapt the system to receive/send data from/to the PE or the DT. It has a well-defined semantics and is intended to be integrable with other standardized tools through the definition of standard structures for the DTs. It proposes Eclipse Vorto as the way to define DT structures. Currently, the connectivity can be achieved through the communication protocols advanced message queuing protocol (AMQP), MQ telemetry transport (MQTT), hypertext transfer protocol (HTTP), WebSockets, and Apache Kafka. The security policies are an advantage compared to the other DT frameworks. It provides client Software development kit (SDKs) for Java and JavaScript, where the user can integrate the Ditto cluster and the twins with customized applications. It has a robust documentation and it is supported by the Eclipse foundation. This framework also offers compatibility with Eclipse Vorto and Eclipse Hono for the connectivity, semantics, and modeling.

Complementary information for Tables 2 and 3 of this framework are:

TABLE 2 Comparative analysis for the different DT frameworks (part 1).

Name	Communication	Storage	Support for analytics	Compositionality	Support for physical twin interventions
Eclipse Ditto (2.3.1)	PT2DT2PT	No-SQL database	Level 1	Aggregable features	Expected
Equinox (2.3.2)	None	Files and SQL database	Level 1 & Level 2	None	None
AASX Package Explorer (2.3.3)	PT2DT	Files	Level 0	Aggregable submodels	None
PYI40AAS (2.3.4)	None	Files	Level 0	Aggregable submodels	None
SAP I4.0 AAS (2.3.5)	PT2DT2PT	SQL database & No-SQL database	Level 1	Aggregable submodels	Expected
Eclipse Basyx (2.3.6)	PT2DT2PT	SQL database, No-SQL database, & InMemory	Level 1	Aggregable submodels	Expected
NOVAAS (2.3.7)	PT2DT2PT	Files and Internal	Level 1 & Level 2	Aggregable submodels	Unexpected
CPS-Twinning (2.3.8)	None	Files	Level 0	Theoretically composable	None
Twined (2.3.9)	None	Files	Level 0	Aggregable features	None
Azure DTDL (2.3.10)	None	Files	Level 0	Aggregable features & composition via relationships	None
iTwin.js (2.3.11)	None or PT2DT	iModels & Files	Level 1 & Level 2	Partially composable	Expected
DTCC (2.3.12)	None	Files	Level 0	None	None
TerriaJS (NSW Digital Twin implementation) (2.3.13)	None or PT2DT	Files and APIs	Level 2	None	None
INTO-CPS Co-simulation Framework (2.3.14)	PT2DT2PT	Files	Level 0	Aggregable FMUs	None

1. **Analyzed version**: 2.3.
2. **Storage**: It provides a No-SQL database (MongoDB) and model-based storage for DT data (at the structure level).
3. **Support for analytics**: It offers APIs to deal with Things, Features, Policies, Things-Search, Messages, and CloudEvents. There is also the option to use the Ditto Protocol.
4. **Compositionality**: The way of defining the DTs does not allow direct composition of DTs, but the internal features can be aggregated into a larger DT.
5. **Scalability**: It is possible to generate several DTs and integrate them at a large-scale.
6. **Reproducibility**: Easy to install and easy to set up DTs. There are good documentation and examples.
7. **Interoperability**: Syntactic (by using structured communication formats, such as JSON) and semantic (by following the Ditto/Hono/Vorto metamodels).
8. **Community support**: *Creator*: Association. *# of releases*: 39 releases have been made from 09-2019 to 02-2022. *Documentation* at its own web page.

This framework is considered for the case study implementation.

TABLE 3 Comparative analysis for the different DT frameworks (part 2).

Name	Scalability	Standardization	Reproducibility	Interoperability	Community Support
Eclipse Ditto (2.3.1)	Yes	None	Yes	Syntactic & semantic	Association / 39 releases / Web page
Equinox (2.3.2)	None	None	No	Syntactic	Individual / 1 release, 7 commits / GitHub repository
AASX Package Explorer (2.3.3)	None	Communication & Metadata	Yes	Syntactic & semantic	Industrial Consortium / 23 releases / GitHub repository
PYI40AAS (2.3.4)	Yes	Metadata	Yes	Syntactic and semantic	Research Group / 4 releases / GitLab repository
SAP I4.0 AAS (2.3.5)	Yes	Metadata	Yes	Syntactic and semantic	Company / 1 release, 589 commits / GitHub repository
Eclipse Basyx (2.3.6)	Yes	Metadata	Yes	Syntactic and semantic	Association / 3 releases / web page
NOVAAS (2.3.7)	Yes	Metadata	No	Syntactic and semantic	Research group / 1 release, 139 commits / GitLab repository
CPS-Twinning (2.3.8)	No	Metadata & Behavior	No	Syntactic and semantic	Research group / 3 releases / GitHub repository
Twined (2.3.9)	Yes	None	Partially	Syntactic	Company / 30 releases / GitHub repository and ReadTheDocs
Azure DTDL (2.3.10)	Yes	None	Yes	Syntactic	Company / 2 versions-110 commits / web page
iTwin.js (2.3.11)	Partially	None	Yes	Syntactic	Company / 36 releases / web page
DTCC (2.3.12)	Partially	None	Partially	Syntactic	Research group / 1 release, 908 commits / GitLab repository
TerriaJS (NSW Digital Twin implementation) (2.3.13)	Partially	None	Partially	Syntactic	Public agency / version 8, 20140 commits / web page
INTO-CPS Co-simulation Framework (2.3.14)	No	Behavior	Yes	Syntactic	Association / releases for different tools / web page and ReadTheDocs

2.3.2 | Equinox

This is mainly designed for airplane/aircraft analysis. It works as a software to analyze and simulate airplane/aircraft features and behaviors. Visualization is enabled and integrated parameterized analytics are available. It is more a simulation software package rather than a DT framework (it can import data and models from different airplanes and do analyses on those datasets); therefore, it only covers digital models. It seems not to be currently maintained and fails when deploying the server-side infrastructure locally (which is based on Docker).

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: First release.
2. **Communication**: the tool runs offline.
3. **Storage**: It provides files and a SQL database (MySQL) built on Docker to store DT data.
4. **Reproducibility**: It provides some guidance for installation and setup, but there are no documentation or examples.
5. **Community support**: *Creator*: Individual. *# of releases*: 1 release, 7 commits have been made from 01-2020 to 02-2022. *Documentation* at GitHub repository.

This framework is not considered for the case study implementation because it is domain specific and the case study does not fit to it. It also lacks maturity and documentation for the setup.

2.3.3 | AASX

Asset Administration Shell (AAS) provides a reference standard and framework for DTs in the industrial context under the concept of Industry 4.0 or Industrial IoT. The standard was first introduced by Industrie Plattform 4.0, the German hub for Industry 4.0.[39] AAS has multiple independent implementations, including Eclipse BaSyx, PYI40AAS, AASX, NOVAAS, and SAP I4.0 AAS. The core of the AAS specification[40] is an information meta model to describe assets, such as machines, their components, capabilities and relations. It uses a structure of assets, submodels and properties, also know as submodel elements. The properties uses definitions from external repositories, such as ECLASS* and IEC CCD.† This use of well-defined semantics easens data exchange and provides a basis for automated reasoning.

In the case of AASX, this software provides a graphical user interface (GUI) client to manage AAS objects either AASX, JSON, or XML. It also provides connectivity through HTTP, MQTT, and OPC UA. Although the assets can be connected to a network, it is not able to manage dynamic assets natively, therefore, only works for AAS type 1. The interface is user-friendly and easy to deploy. It provides some examples to work with and for the users to learn and get used to it.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: v2021-08-18.
2. **Storage**: It provides files as storage mechanism for DT data (.aasx, .json, .xml).
3. **Compositionality**: Multiple submodels can be aggregated into a larger DT, but composition of DTs (*assets*) is not supported.
4. **Scalability**: None–this tool can run just one asset at a time with no chances of running several DT instances.
5. **Standardization**: OPC UA at communication level and AAS implementation at the metadata level.
6. **Reproducibility**: It provides reproducibility through documentation, demos, and examples.
7. **Community support**: *Creator*: Industrial Consortium. *# of releases*: 23 releases have been made from 08/2020 to 02/2022. *Documentation* at GitHub repository.

This framework is not considered for the case study implementation because of it being a GUI, it has limitations regarding the administration of the DTs.

2.3.4 | PYI40AAS

This SDK provides a Python module implementation for some features of the AAS standard. It contains the same modules as Eclipse BaSyx Python SDK. It also allows the creation, reading, modification, validation, and conversion of format of AAS packages (AASX, JSON, XML). It only works with static assets or files (AAS type 1).The module has basic documentation and five examples. It provides an easy way to create Assets and submodels from Python. It also provides a backend feature that allows the connection to databases to store, read, and update objects. It has a basic connection to AAS API clients through an AAS provider, but it cannot be binded to any AAS infrastructure services.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 0.2.2.
2. **Storage**: It provides files as storage mechanism for DT data (.aasx, .json, .xml).
3. **Compositionality**: Multiple submodels can be aggregated into a larger DT, but composition of DTs (*assets*) is not supported.
4. **Standardization**: AAS implementation at the metadata level.
5. **Reproducibility**: It provides some documentation and examples. Easy to install and set up the DTs.

*https://www.eclass.eu/.

†https://cdd.iec.ch/.

6. **Community support**: *Creator*: Research Group. *# of releases*: 4 releases have been made from 05/2020 to 02/2022. *Documentation* at GitLab repository.

This framework is not considered for the case study implementation since it does not provide a back-end to run it, but the AAS assets can be easily generated with it.

2.3.5 | SAP I4.0 AAS

This implementation is primarily implemented in Typescript and distributed in Docker containers. To get started, there are examples of how to create new Skills. Skills are analytics, that can access data from the RabbitMQ broker and the MongoDB database. There are several endpoints for registering assets and requesting information. These all comply with the RAMI 4.0 specification. The framework provides REST and GRPC ingress- and egress-endpoints for live data from/to physical assets. The asset registry adheres closely to the RAMI specification and supports the structure and semantics of the specification.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 25-08-2021.
2. **Storage**: It provides an SQL database (PostgresSQL) for registering assets and a No-SQL database (MongoDB) for storing events and data. It has an adapter registry that allows additional storage adapters.
3. **Support for analytics**: It provides APIs and analytics can be built using Node Red, but must be implemented manually.
4. **Compositionality**: Multiple submodels can be aggregated into a larger DT, but composition of DTs (*assets*) is not supported.
5. **Scalability**: It uses a micro-service architecture on Docker with a RabbitMQ broker for asset data exchange and databases for storage. Also supports for new interface adapters.
6. **Standardization**: AAS implementation at the metadata level.
7. **Reproducibility**: It provides reproducibility through documentation and examples.
8. **Community support**: *Creator*: Company. *# of releases*: 1 release in development mode, 589 commits have been made from 11/2019 to 09/2021. *Documentation* at GitHub repository.

This framework is considered for the case study implementation.

2.3.6 | Eclipse Basyx

This software provides a Java/.NET implementation of the AAS standard. It provides infrastructure for AAS Server and Registry Server (both as services), either deployed as Docker containers or as JAR executables. It supports creation, reading, integration, modification, update, and deletion of AAS objects, submodels, and submodel elements. It also supports AAS types 1, 2, and 3. The implementation is designed as an SDK with multiple functionalities. The integration allows different networks and protocols that need to be coded and orchestrated through the virtual automation bus (VAB). This mechanism offers a common semantics to work with AAS elements. Currently, it supports the protocols TCP, HTTP/REST (APIs), and MQTT; the OPC UA implementation is in progress. Other protocols can be integrated through Java or C# independently. A secure HTTP connection can be configured. The assets can embed interfaces to invoke operation through HTTP/REST and the VAB, although it can undergo some synchronization problems. It also offers extended services that require external/third-party software, such as device integration, network to network integration with gateways, integration with plant strategy and optimization services, process control, and monitoring. Among the possibilities, there are Graphana and Node-red. It has a robust documentation, several basic and advanced examples, and the support of Eclipse. It also provides some limited SDKs for Python, C++, and RUST.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: basyx-sdk-1.0.1, BaSyx AAS Server 1.0.1, BaSyx Registry Server 1.0.3.
2. **Storage**: It provides a SQL database (PostgresSQL) for registering assets and a No-SQL database (MongoDB) for storing events and data. InMemory option is also available.

3. **Support for analytics**: It provides APIs and additional analytics can be developed using Java or C# code manually.
4. **Compositionality**: Multiple submodels can be aggregated into a larger DT, but composition of DTs (*assets*) is not supported.
5. **Scalability**: It provides the infrastructure to run several DTs at the same time and a micro-service architecture on Docker or on JAR executables.
6. **Standardization**: AAS implementation at the metadata level.
7. **Reproducibility**: It provides reproducibility through documentation, demos, and examples.
8. **Community support**: *Creator*: Association. *# of releases*: 3 releases have been made from 01/2021 to 02/2022. *Documentation* at its own web page.

This framework is considered for the case study implementation.

2.3.7 | NOVAAS

This software offers a complete and fully integrated nore-red-based implementation for AAS deployed using Docker containers. It has integrated connectivity of assets, communication, and dashboard. Within the platform, it is possible to subscribe/unsubscribe to/from topics for communication purposes. It supports static and dynamic environments. It also allows to track the status of the variables of interest. It has an embedded dashboard to create and update graphs and alarms in runtime. It offers a fully integrated implementation that is easy to install and easy to use, but in-detail documentation to work with this tool is not provided.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 2.4.
2. **Storage**: It provides files as storage mechanism for DT data and integrated SQL storage for events and in-platform settings.
3. **Support for analytics**: It provides APIs and integrated in-platform analytics, additionally, it runs upon Node-red, making this tool modular to set extended analytics manually.
4. **Compositionality**: Multiple submodels can be aggregated into a larger DT, but composition of DTs (*assets*) is not supported.
5. **Scalability**: It provides a micro-service architecture on Docker and infrastructure to run several DTs at the same time.
6. **Standardization**: AAS implementation at the metadata level.
7. **Reproducibility**: It provides some guidance for installation and setup, but there are no documentation or examples.
8. **Community support**: *Creator*: Research group. *# of releases*: 1 release, 139 commits have been made from 12/2020 to 02/2022. *Documentation* at GitLab repository.

This framework is not considered for the case study implementation because it lacks documentation to extend the examples or create diverse scenarios.

2.3.8 | CPS-twinning

It has the main purpose of creating state replication for PEs into DTs in simulated networks based on AutomationML (AML) documents. The documentation is poor and the current version is not going to have any update. It works currently on Python2. There will be a new open-source version. The second version is expected to have these features (described by the author): “The idea is to have a modular architecture that consists of an engine that delegates commands to plugins that implement the DTs. We will provide a few plugins out-of-the-box, such as QEMU/KVM via libvirt, LXC via libvirt, 4diac (FORTE), and FMU/FMI to attach physical simulations.”

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: CPS-Twinning 0.0.1, CPS-state-replication 0.0.1.
2. **Storage**: It manages the DT from AML files.
3. **Compositionality**: Theoretically, two different DTs represented by AML files can be composed into one larger DT.
4. **Standardization**: Implements AML at the metadata and behavioral levels.

5. **Interoperability**: The models use XML format and AML metadata.
6. **Reproducibility**: It provides some guidance through the installation and setup, but there are no documentation or examples.
7. **Community support**: *Creator*: Research group. *# of releases*: 3 releases have been made from 01/2018 to 02/2022. *Documentation* at GitHub repository.

This framework is not considered for the case study implementation because it is not yet a stable and mature tool.

2.3.9 | Twined

This is a general-purpose DT definition language framework (not especially designed for industry). It was developed by the private company Octue, but it is open-source. The DTs are launched from JSON files with a specific schema provided by the Twined framework, consisting of: Configuration Values, Configuration Manifest, Input Values, Input Manifest, Output Values, Output Manifest, Credentials, Children (other DTs), and Monitors. The frameworks offers the creation and validation of DTs that are static and require a back-end to work in dynamic environments. The back-end is not explained or specified regarding its requirements. So, it is difficult to check the functionality of the framework. The documentation is maintained by Octue and so the back-end to run the DTs. Any other back-end that understands the schema could work with the definition of the DTs in this framework.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 0.1.2.
2. **Communication**: None (if back-end is not considered). PT2DT2PT if back-end is considered.
3. **Storage**: The DTs are stored as JSON files.
4. **Support for analytics**: The limited open-source part of the framework does not provide APIs or connectivity services.
5. **Compositionality**: The way of defining the DTs does not allow direct composition of DTs, but the internal features can be aggregated into a larger DT.
6. **Reproducibility**: It provides some guidance to create and validate DTs. No information is given regarding the integration with the platform.
7. **Interoperability**: It is achieved thanks to JSON structures with a common schema.
8. **Community support**: *Creator*: Company. *# of releases*: 30 releases have been made from 09/2019 to 02/2022. *Documentation* at GitHub repository and ReadTheDocs.

This framework is not considered for the case study implementation because it does not allow you to run the whole DT framework for free.

2.3.10 | Azure digital twins definition language

This is a general-purpose DT definition language framework (not especially designed for industry). It was developed by the private company Microsoft, but it is open-source. The framework provides a schema to create DTs under a common definition through JSON-LD files, and can be easily integrated with RDF. The schema for the fields, also called *interfaces*, consists of: Identifier, Type, Context, Comment, Contents (*with inner embeddable schemas–Telemetry, Property, Command, Relationship, Component*), Description, DisplayName, Extends (for inherited interfaces), and Schemas (for reusable schemas). The frameworks offers the creation of DTs that are static and require a back-end to work in dynamic environments. The available back-end is Microsoft Azure, which is not open-source nor free-to-use. Any other back-end that understands the schema could work with the definition of the DTs in this framework. It also has known feasible integrations with third-party software, such as its integration with iTwin.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: v2.
2. **Communication**: None (if back-end is not considered). PT2DT2PT if back-end is considered.
3. **Storage**: The DTs are stored as JSON files.
4. **Support for analytics**: The limited open-source part of the framework does not provide APIs or connectivity services.

5. **Compositionality**: The way of defining the DTs does not allow direct composition of DTs, but the internal features can be aggregated into a larger DT. Additionally, the framework provides relationships, where properties like *isComposedOf* can be implemented at the syntactic level.
6. **Reproducibility**: It provides reproducibility through documentation and examples.
7. **Interoperability**: It is achieved thanks to JSON structures with a common schema.
8. **Community support**: *Creator*: Company. *# of releases*: 2 versions, 110 commits have been made from 05/2019 to 02/2022. *Documentation* at its own web page.

This framework is not considered for the case study implementation because it does not allow you to run the whole DT framework for free.

2.3.11 | iTwin.js

This framework provides a set of tools for DT in the infrastructure industry. It was developed by the private company Bentley Systems, but it is open-source. It is required to use an iModel, which is a composed 2D or 3D model with additional integrated information, as the base to create and run a DT. Supported files extensions can be generated on commercial software for infrastructure and CAD modeling, especially by using the company's software packages. There is also the option to create models, objects, and shapes using the iTwin nodejs module and the iTwin Sandbox. Documentation and support are highly available. The deployments use meta-information of the models in the cloud, which can be deployed locally or remotely. The machine running the applications requires certain graphical capabilities to display the visualizations. The framework does not provide an explicit way to integrate real-time data, as this framework is targeted to infrastructure industry, but DT data can be stored locally for further integration at the communication level and a complete DT can be obtained by extra coding.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 3.0.0.
2. **Connectivity**: None by default. PT2DT or even PT2DT2PT can be obtained by writing extra code or using third-party integrations.
3. **Storage**: DTs are stored as iModels on the cloud and there is the option to store DT data in files.
4. **Support for analytics**: APIs and integrated analytics can be enabled in-platform.
5. **Compositionality**: Two DTs can be composed to create a new one, but the two objects need to be modeled in one file separately (reusing the models of the individual DTs).
6. **Scalability**: The framework is very computing-demanding, so it can fail with a limited amount of twins.
7. **Reproducibility**: It provides reproducibility through documentation and examples. It is easy to set up the tool, while it can be difficult to set up the models for nontechnical people related to infrastructure engineering.
8. **Interoperability**: It is achieved thanks to standard API connectivity.
9. **Community support**: *Creator*: Company. *# of releases*: 36 releases have been made until 02/2022. *Documentation* at its own web page.

This framework is considered for the case study implementation.

2.3.12 | Digital twin cities centre platform

This is a DT Platform for Smart Cities and city management. The development of the platform is still in process but is already functional. It provides a set of tools for data processing, modeling, and simulation of city models. It also provides visualization through their web browser viewer (https://view.dtcc.chalmers.se) or using external software (ParaView),[‡] which enables the recreation of the outputs generated by DTCC. It provides Docker images to run the framework. The tools are managed using a terminal. There is no GUI to interact with the tool yet. Documentation is still poor and only provides information for installation and setting up of the demos. No additional information for customized applications.

[‡]https://www.paraview.org/.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: beta version.
2. **Storage**: The models are stored as JSON and VTS files.
3. **Compositionality**: The way of defining the DTs does not allow merged entities.
4. **Scalability**: The framework is very computing-demanding, so it can fail with a limited amount of twins.
5. **Reproducibility**: It provides some guidance for installation, demos, and setup, but there are no documentation or examples on how to use.
6. **Interoperability**: It is achieved by JSON structures with a common schema.
7. **Community support**: *Creator*: Research group. *# of releases*: Beta version, 908 commits have been made from 02/2019 to 03/2022. *Documentation* at GitLab repository.

This framework is not considered for the case study implementation because it is domain specific and the case study does not fit to it.

2.3.13 | Terria JS (NSW digital twin implementation)

It is a DT framework, whose foundation is to provide geospatial data management for building web-based catalog explorers. TerriaJS, the core of the tool, is open-source§ and developed by New South Wales (NSW) state, Australia. The components mostly behave as Digital Models or Digital Shadows depending on how the data is integrated into the platform. It provides a tool to manage geographical, environment, and infrastructure information, including visualization and interrogation. This platform and data sources are mainly designed for NSW state, but it can be extended to other areas. The NSW DT implementation provides an open framework of geospatial data services that supports commercial and community innovation. The data sources are supplied by public and government agencies. It is not highly customizable unless developing almost everything from scratch. It offers a web client to interact with the map and see some features while adding data layers on the map with stored available datasets. There is also the option to run TerriaJS with a customized deployment instead of the web client.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: 8.
2. **Connectivity**: None (offline through stored files) or PT2DT (online if the resource is provided via APIs).
3. **Storage**: It obtains data from open data sources and APIs. Own sources can also be used.
4. **Compositionality**: The way of defining the DTs does not allow merged entities.
5. **Scalability**: The framework is very computing-demanding, so it can fail with a limited amount of twins. Although it supports several layers on the same map. It runs on a cloud server, so it should have some limitations to avoid Denial of Service. If deployed locally, the capacity would depend on hardware capabilities.
6. **Interoperability**: It is achieved thanks to standard geospatial data formats.
7. **Reproducibility**: The platform provides some documentation and tutorials. TerriaJS provides vast documentation.
8. **Community support**: *Creator*: Public agency. *# of releases*: TerriaJS version 8, 20140 commits have been made from 02/2019 to 03/2022. *Documentation* at its own web page.

This framework is not considered for the case study implementation because it is domain specific and the case study does not fit to it.

2.3.14 | INTO-CPS co-simulation framework

Analyzed version: It is a co-simulation framework presented by Thule et al.[41] (we refer the reader interested in an introduction to co-simulation to the work of Gomes et al.[42]), which allows users to download and use other modeling and

§https://github.com/TerriaJS/terriajs.

simulation tools. It allows the integration of digital and physical counterparts with continuous time or discrete-event models. It works upon functional mock-up units (FMUs) under the functional mock-up interface (FMI) standard.[43] It is mainly designed for PT2DT communication (unidirectional), but it is possible to achieve PT2DT2PT communication through the RabbitMQ FMU[44] which facilitates sending messages from the PT to the DT. The user still has to implement the method to send messages to RabbitMQ. Simulation results can be stored in a csv file, or streamed to some other place, but the user still needs to code this functionality. This tool, being a framework working with FMU blocks and thus, integrating a wide range of different models, offers a different perspective of DT from the co-simulation environment. It has strengths regarding the inclusion of more modeling methods and high-fidelity physical models.

Complementary information for Tables 2 and 3 of this framework are:

1. **Analyzed version**: INTO-CPS Application 4.0.5 (to setup co-simulation scenarios), Maestro 2.2.0 (to run co-simulations scenarios), UniFMU 0.0.7 (to export FMUs) and RabbitMQ FMU 2.1.3 (to stream data to from co-simulation using RabbitMQ).
2. **Storage**: None by default, storage in files can be achieved by writing extra code.
3. **Compositionality**: Multiple FMUs featuring multiple DTs can be aggregated into a larger DT. Additionally, there is a tool to join two FMUs into one, which can be used for composed DTs.[45]
4. **Standardization**: FMI at the behavior level.
5. **Reproducibility**: It provides reproducibility through documentation and examples.
6. **Interoperability**: It is achieved thanks to the FMI standard.
7. **Community support**: *Creator*: Association. *# of releases*: Not applicable because it is composed of different tools. *Documentation* at its own web page and ReadTheDocs.

This framework is considered for the case study implementation.

2.4 | Categorization of DT frameworks

From the analysis, we could identify similar patterns of the tools regardless of their maturity or features. Therefore, the tools were grouped according to their modeling and technological domain. They were categorized in six different groups, namely:

1. **Structured data DT framework:** This category is represented by frameworks that provide a common data structure to model DTs and the infrastructure to manage them along with structured interfaces. It is inspired from IoT-based frameworks providing bi-directional communication, like the so-called DT platforms.[46] From the theoretical definition of DT, this category focuses on fulfilling the bi-directional communication criterion, and it has strengths in terms of data modeling, synchronization between DT and PT, and security in the channels. On the downside, it does not focus sufficiently on simulation models, data analysis, and having the DT to fulfill a particular business goal. Its main strength is the use of metamodels and infrastructure for the deployment of DTs. This category includes Eclipse Ditto and the different AAS implementations (Eclipse BaSyx, SAP I4.0 AAS, NOVAAS, PYI40AAS, and AASX Package Explorer).
2. **Domain specific DT framework:** This category comprises the frameworks that provide a very specific and narrow domain to work with DTs. Its name is given due to the framework being able to provide solutions for a particular domain, that is, it is not extendable to other domains. From the theoretical definition of DTs, this category should be able to provide a more integrated and complete set of services to fulfill all the criteria of DTs for a certain domain. On the downside, in terms of tooling, the tools are not generalizable and extensible to other domains, and perhaps less reusable. Its main strength is the provision of specialized tools and methods for a certain domain. Equinox, which is focused only on aerospace and aeronautics, was the only one allocated to this category.
3. **Language specification DT framework:** This category is represented by frameworks that provide a language specification for DTs using a common data structure, but not necessarily the infrastructure to manage them along with structured interfaces. It is inspired from specification languages under a common metamodel to provide the so-called meta-level DTs.[47] From the theoretical definition of DTs, this category describes the core structure for DTs in IoT-based DT platforms, and if integrated with them, provides the interfaces for enabling bi-directional communication seamlessly. On the downside, these frameworks mainly focus on the data models or schemas rather than on the simulation

and behavioral aspects of DTs. Its main strength is the definition of a common language for modeling DTs with state-of-the-art technology. DTDL and Twined were categorized in this category.

4. **Geospatial data DT framework:** This category is represented by frameworks that provide a foundation for geospatial data management. It is inspired from the enabling technologies and tools for geospatial data management[48] for the specific niche of geospatial applications for cities and countries. From the theoretical definition of DTs, this category is the most distant to fulfill all the criteria to achieve functional DTs; however, it provides a set of tools that are interesting for the niche of geospatial DTs, which may be worth exploring for DT applications in, for instance, smart cities, integrated transportation, and weather analysis. Its main strenght is the management of DT above a geospatial information layer. DTCC and TerriaJS were categorized in this group.
5. **3D-based and infrastructure-oriented DT framework:** This category is represented by frameworks that provide features to work with 3D objects and infrastructure along with the visualization engine required for it. It is inspired from the enabling technologies and tools for geometric modeling,[49] such as 3D modeling software and Building Information Modeling software. From the theoretical definition of DTs, this category is strong in relation to integrating physical features and constraints based on geometric modeling and its interaction with the environment, and providing visualization for human interpretability and interaction. On the downside, the interfaces to achieve bi-directional communication may not be trivial and some features that do not require visualization or physics analysis may be non-viable to be integrated in this category. Its main strength is the provision of visualizations and 3D representations. iTwin was the only one allocated to this category.
6. **Co-simulation and model-based DT framework:** This category is represented by frameworks that provide modeling and simulation features as part of the core of the solution. It is inspired from the enabling technologies and tools for behavioral modeling.[49] From the theoretical definition of DTs, this category is strong in terms of providing DTs supported by behavioral models, also integrated synchronously with their PE; therefore, it enables having DTs that fulfill a particular business goal, for example, using *what-if* scenarios. On the downside, the interfaces to achieve bi-directional communication may not be trivial to implement and there is no clear boundary from one DT to another in composed DTs due to the them not being encapsulated in object artefacts. Its main strength is the inclusion of behavioral models and simulations for DTs. INTO-CPS and CPS-Twinning were allocated to this category.

Once the categories were created, some analyses were performed to show the readers a summary of the different attributes of the tools graphically. As the criteria were mostly categorical, some of the categories that could be discretized into levels were considered for the following charts. The values for the categories were assigned from zero to two, representing *uncovered* up to *highest rank/complied*. In the case of the support for analytics dimension, the highest rank means level 2, because this was the maximum reached level in the survey. Figure 3 shows a set of spider diagrams for the structured data DT framework group. Figure 4 shows the same information for the domain specific DT framework group, the language specification DT framework group, the geospatial data DT framework group, the 3D-based and infrastructured-oriented DT framework group, and, the Co-simulation and Model-based DT framework group respectively.

Besides, considering that a DT engineering process may require more than one tool or framework to accomplish its purpose, that is, it requires a DT *constellation*[32] made of *models and data*, *enablers/tools*, and *usages/services*, we also identified potential combination opportunities for the tools, where they can either be integrated with or be complementary to each other in order to achieve such a goal. This proposition is based on the authors' perspective and focuses only on the 14 surveyed frameworks and does not consider integrations with external tools. However, this does not mean that these 14 tools are sufficient to accomplish a DT constellation, and that DTs cannot or should not be integrated with other complementary tools, which is most likely the case, due to some other tools may be necessary for the completion of the DT and its constellation. Hence, the reader is encouraged to explore the integration with external tools that may be complementary.

Figure 5 presents an overview of how the tools can be combined for a common sake. Starting from left to right, AAS implementations can be integrated with and are also complementary to CPS-Twinning and vice versa since both are designed for the industrial domain and share some core foundations; both are targeted to cover automation engineering models with a IEC-based standard data structures. CPS-Twinning and INTO-CPS share some properties in common, such as the fact of being model-based frameworks, and both consider using the FMU blocks. INTO-CPS and Ditto can be integrated together, where Ditto provides more advanced connectivity interfaces to INTO-CPS, whereas INTO-CPS provides more advanced simulations and models. DTCC and TerriaJS are both geospational frameworks and their models

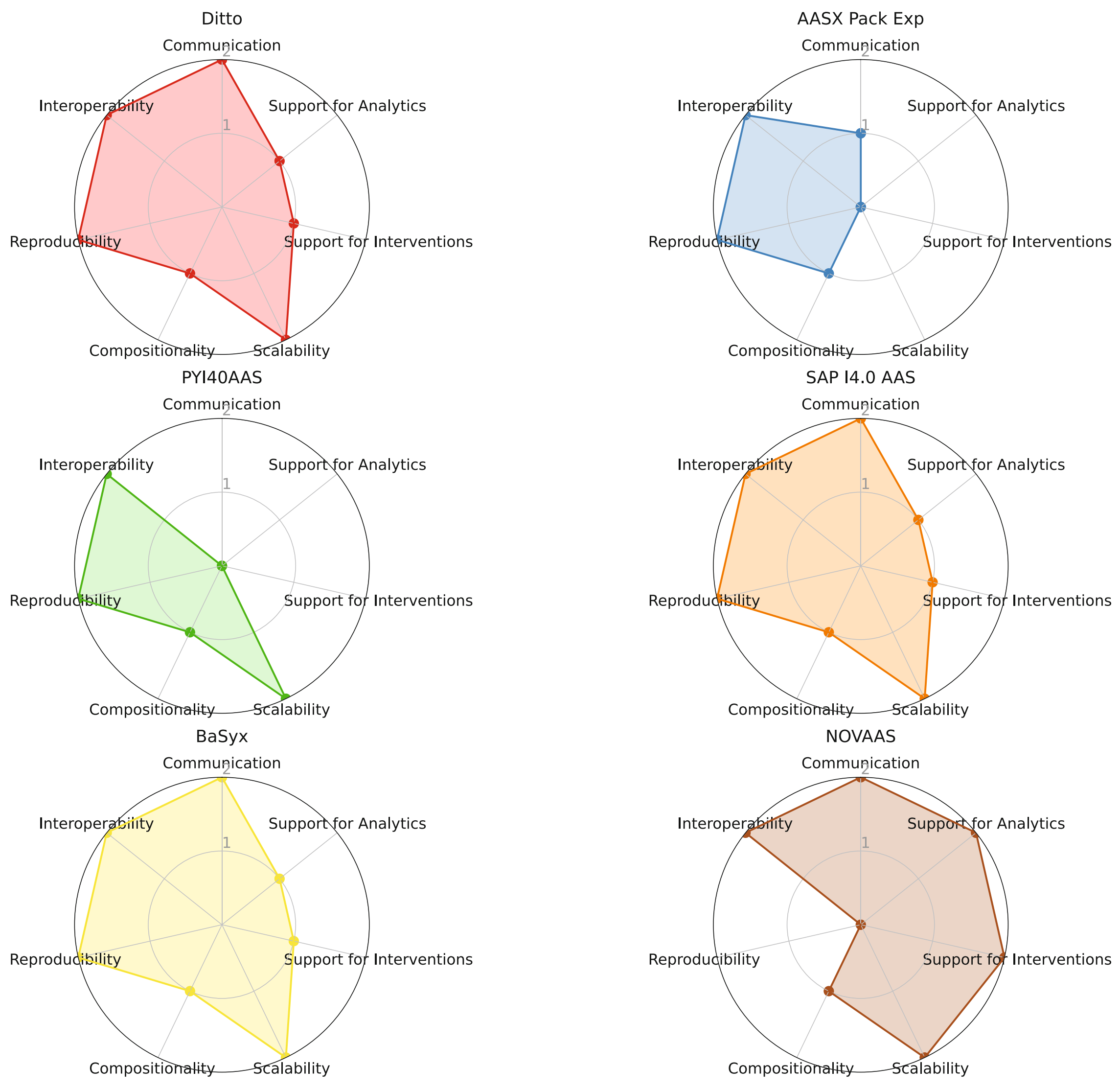


FIGURE 3 Comparison between the frameworks in the *Structured data DT framework* category.

can be or become compatible to a certain extent. They can also be integrated with Ditto at the connectivity layer to provide more options regarding data models and communication interfaces.

Similarly, Ditto and iTwin can be integrated together, where iTwin provides the 3D and visualization features. iTwin can be integrated with the Microsoft Azure platform and its DTDL language. There are use cases about the integration of iTwin with Azure that can boost the learning process of this potential combination. Both DTDL and Twined provide a specification language and their models can become compatible to a certain extent to migrate from one environment to another. Equinox, on the other hand, did not get any match to be combined with other tools because of its narrow domain and lack of documentation.

3 | CASE STUDY—INCUBATOR

In this section, in order to explore some of the capabilities and features of some of the tools and give the readers a notion of how these frameworks can be implemented for a particular use case, we present implementations using the Incubator case study, introduced in Reference 24, as a simple DT scenario of a heating/cooling system.

The system used in this work is the thermal incubator system that was originally proposed as a case study for DT engineering. An overview of the incubator is shown in Figure 6.

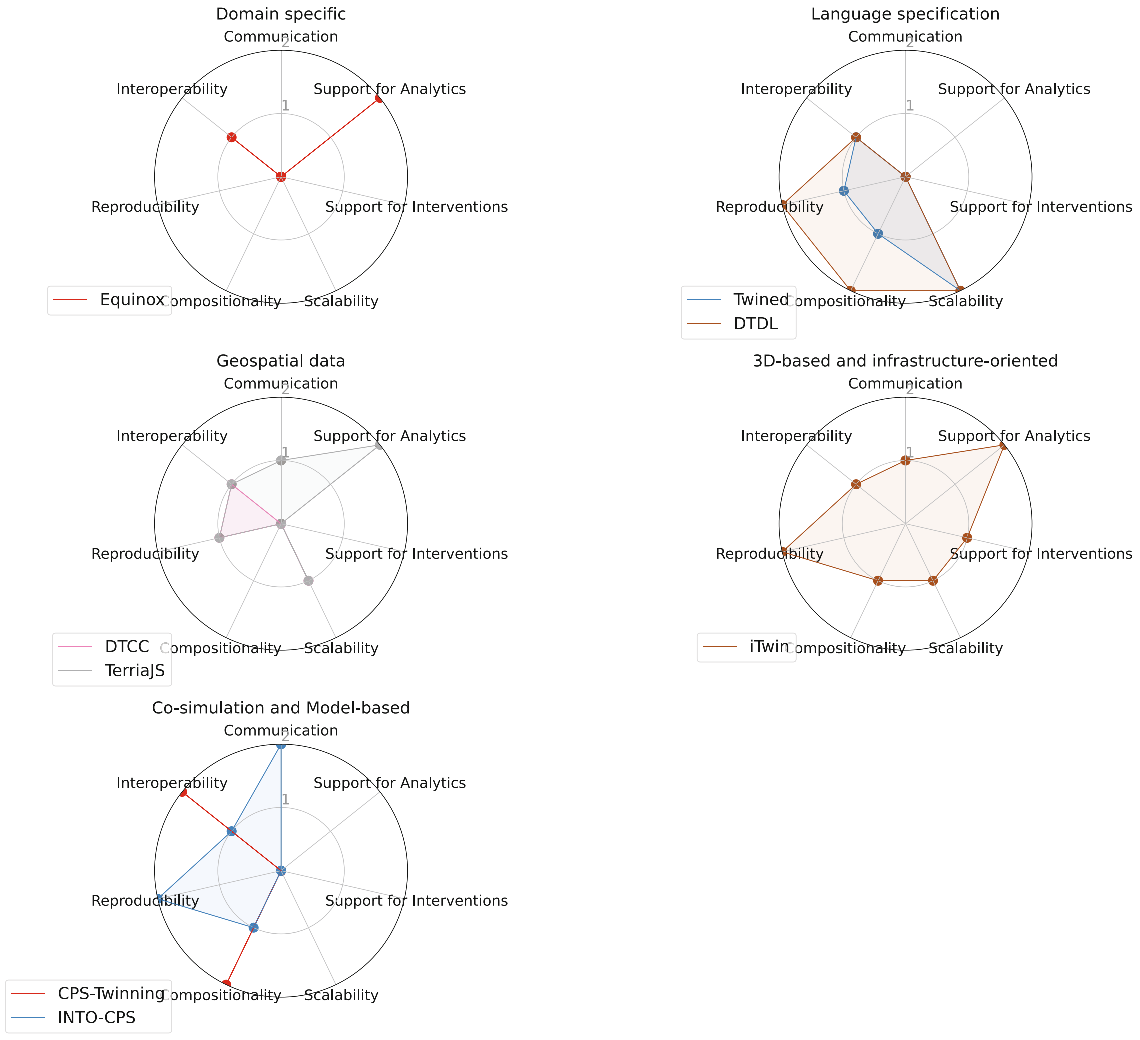


FIGURE 4 Comparison between frameworks in the categories *domain specific, language specification, geospatial data, 3D-based and infrastructure-oriented, and co-simulation and model-based* DT frameworks.

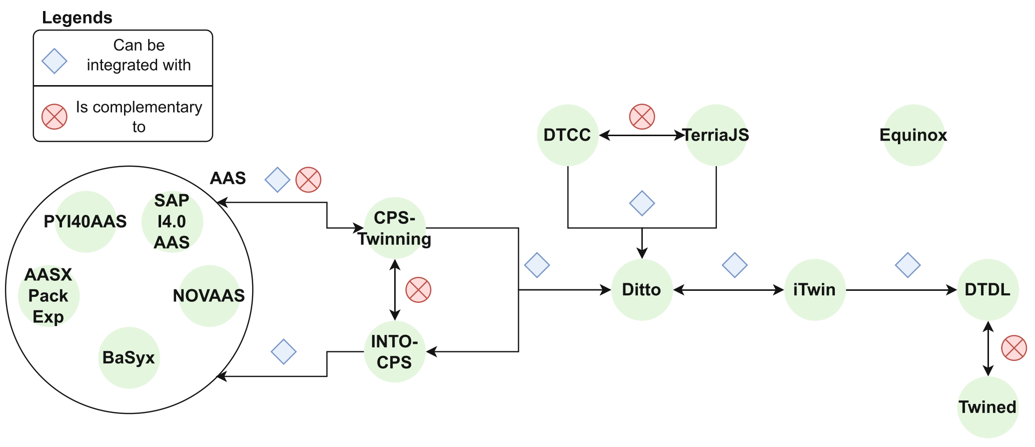
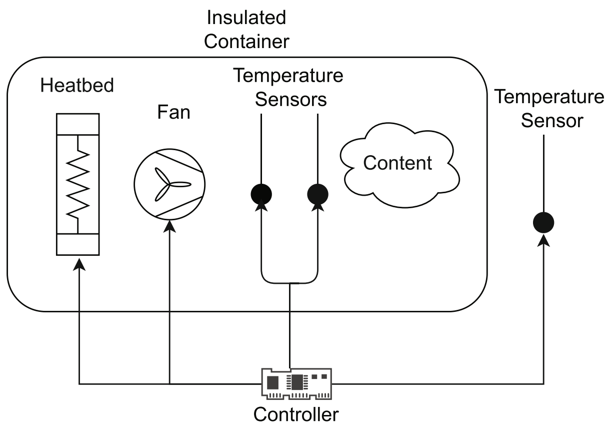


FIGURE 5 Network graph for potential combinations of the tools.



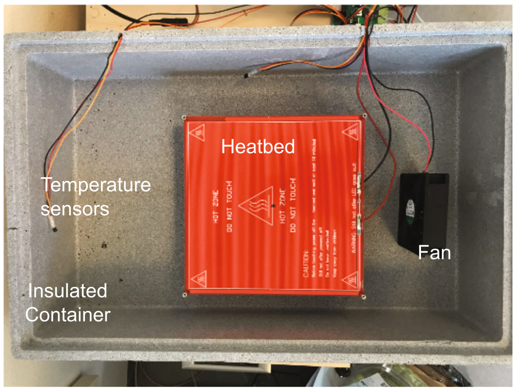


FIGURE 6 Left is the schematic overview of the incubator and right is the real incubator system. Adapted from Reference 24.

The incubator system is composed of a styrofoam box, a fan, three temperature sensors, a heating device called a heatbed, and a controller. The controller can set the fan and the heatbed to be turned on or off. The fan is always on when the incubator is turned on, while the heatbed can be switched on or off by the controller. For more details, we refer the reader to the technical report[24] and the online documentation.¶

3.1 | SAP I4.0 AAS

SAP's open-source implementation of the RAMI specification provides REST interfaces to register and interact with assets. As it follows the RAMI specification like Eclipse BaSyx, the configuration of assets, submodels, and submodel elements is similar to the one shown in Figure 7. When a storage adapter, such as PostgresSQL adapter, has been registered in the Adapter Registry, then submodels can be assigned to this. The content of the submodel is then sent as JSON to the Data Manager. Skills, such as detecting a temperature anomaly, interact with Ingress- and Egress services by means of an AMQT broker. AMQP queues are created and named in the Skills, and routing to/from Egress-/Ingress services is managed by a routing key based on semantic protocol, receiver role, name, and type. The semantic protocol is registered in the Data Manager. SAP provides an Onboarding Skill, written in Typescript, that uses xstate to create a state machine that reacts on data from the AMQP broker. This skill can easily be adapted to provide temperature anomily detecting and be deployed as a new micro service. The source code for the SAP framework can be downloaded from github (SAP/i40-aas) and prebuilt containers can be downloaded from dockerhub (sapi40).

The reader is encouraged to visit the available demo at.#

3.2 | Eclipse BaSyx

When working with Eclipse BaSyx, the approach is under the AAS concept, and so, mostly oriented to the manufacturing industry context. AAS and BaSyx provide the basic structure to model any system within the components of AAS, asset, submodel, and submodel elements. In this case, the incubator, can be represented by: the *Incubator AAS*, the shell that contains the *Incubator Asset*, which at the same time can be represented by just one submodel, the *Incubator Operational Data Submodel*, which includes different submodels elements, such as *Temperature*, which is a *Property* represents the current temperature of the incubator; *Heatbed activation* and *Heatbed deactivation*, which are *Operations* that activate/deactivate the physical heatbed of the incubator; and *Fan activation* and *Fan deactivation*, which are *Operations* that activate/deactivate the physical fan of the incubator. Figure 7 is also applicable for this implementation. The operations can also be represented by a *Heating operation*, which activates the heatbed and deactivates the fan, and the *Cooling*

¶https://github.com/INTO-CPS-Association/example-incubator.

#https://gitlab.au.dk/au698550/dtframeworksurvey_publiccasestudy/-/tree/master/Sap-i40-aas.

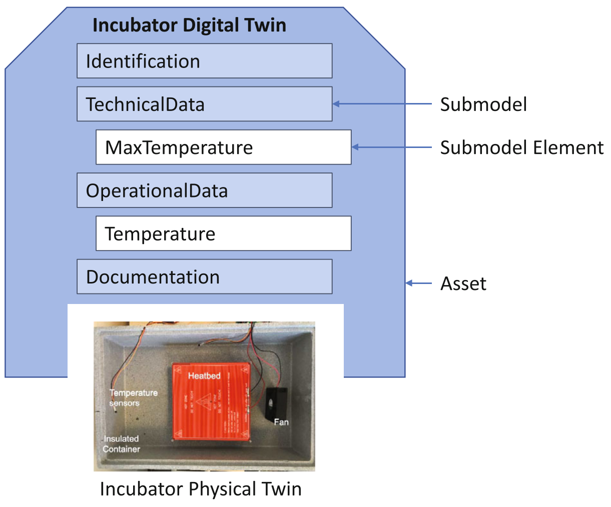


FIGURE 7 Incubator represented in AAS.

operation, which activates the fan and deactivates the heatbed. Other static parameters of the incubator can be added to the *Incubator Technical Data Submodel*, but they are not detailed in this example. The information of the asset and its component are available at the registry and the AAS servers and accessible through APIs. Both servers are run from the JAR files, but there is also the option to run them from Docker containers (Using the JAR files enables some personalization on the way of accessing the data in the machine). The operations are mounted on the VAB, where they can be executed via a remote call. The DT of the incubator is bound to the PT through RabbitMQ for PT to DT data and for DT to PT commands using the provided Client SDK. The development required manual coding for the PT to DT interfacing and setting up a controller/analytics service to set the desired temperature within a range of degrees.

The reader is encouraged to visit the available demo at.[‖]

3.3 | Eclipse ditto

The implementation with Ditto works similar to the implementation to Eclipse BaSyx with a different semantics. Ditto provides the infrastructure and servers where the Things and their properties are stored and available. In this case, the incubator is represented by the *Thing Incubator*, that has some properties, such as the creator, and the features *Temperature*, which contains the *properties* value of the temperature and the units Celsius and the *Desired property* value, which is the desired temperature of the system; *Fan*, which contains the *properties* automatic mode and state of the output; and *Heatbed*, which also contains the *properties* for automatic mode and state of the output. The *Thing* and the *Features* can be accessed through the provided API or any of the channels of the Ditto protocol. The controller/analytics services are also deployed through the client SDK. The development required manual coding for the PT to DT interfacing and setting up specific tasks. This implementation used the available Docker containers for the setup of the infrastructure. Figure 8 shows the implementation of the incubator with Eclipse Ditto. It requires the *Thing* to be modeled, such as in the diagram, and then interpreted as a JSON file to be stored in the infrastructure. Once the *Thing* is up and running, the different attributes and features can be accessed from the different communication mechanisms available in Ditto for reading and writing.

The reader is encouraged to visit the available demo at.[**]

3.4 | iTwin

iTwin uses TypeScript can be integrated to external IoT sources through APIs. As this framework is mainly oriented to 3D-based DTs, the 3D object becomes mandatory. It the case of the incubator, it is a box with some equipment inside.

[‖]https://gitlab.au.dk/au698550/dtframeworksurvey_publiccasestudy/-/tree/master/BaSyx.
[**]https://gitlab.au.dk/au698550/dtframeworksurvey_publiccasestudy/-/tree/master/Ditto.

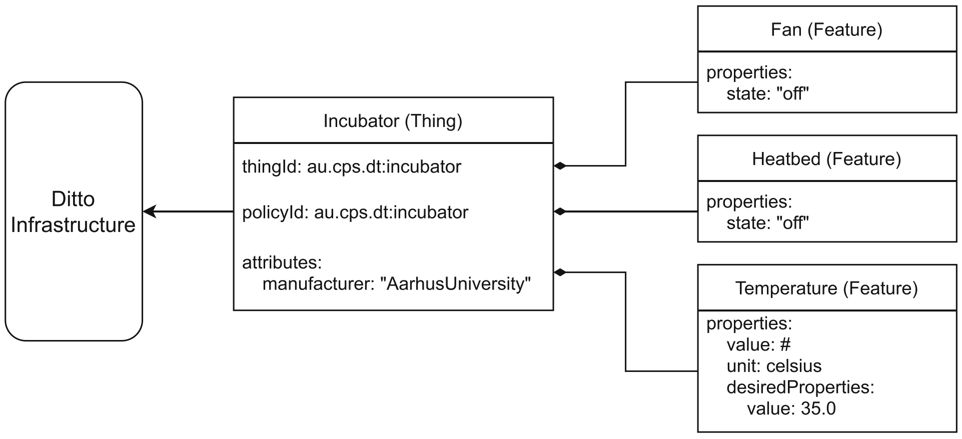


FIGURE 8 Incubator implementation on eclipse ditto.

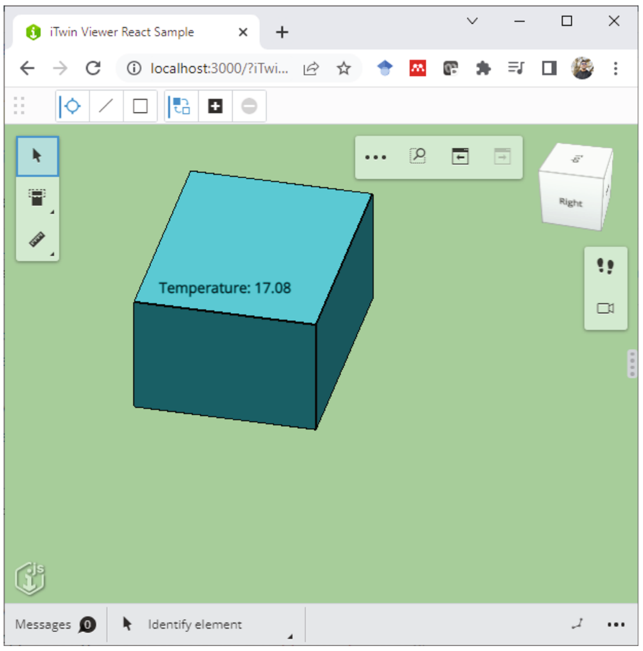


FIGURE 9 Incubator implementation on iTwin.

For this basic implementation, the incubator was set up as a box created with the iTwin Geometry-core toolbox. It was not required to set up the incubator as an iModel (but for complex models it is recommended). The incubator on iTwin also includes a method to update the color of the box based on the obtained temperature within a heatmap color range (blue to red) and a marker to show the temperature of the PT. The incubator setup is presented in Figure 9. Additional components, such as toggle buttons, toolbars, controllers, and so forth, can be used for activating/deactivating the outputs of the incubator, but they were not considered for this example. The integration between the twins was achieved through a set of steps, since the iTwin app runs on the browser (client-side because it is based on ReactJS), it cannot establish a direct server-side communication method, such as RabbitMQ. Then, an additional layer using WebSockets was required to bind the RabbitMQ messages to the iTwin implementation.

Certainly, iTwin offers interesting 3D visualization options for DT application, but naturally, it requires more lines of code and time to set up than a simple non-visual abstract setup.

The reader is encouraged to visit the available demo online.[††]

[††]https://gitlab.au.dk/au698550/dtframeworksurvey_publiccasestudy/-/tree/master/iTwin.

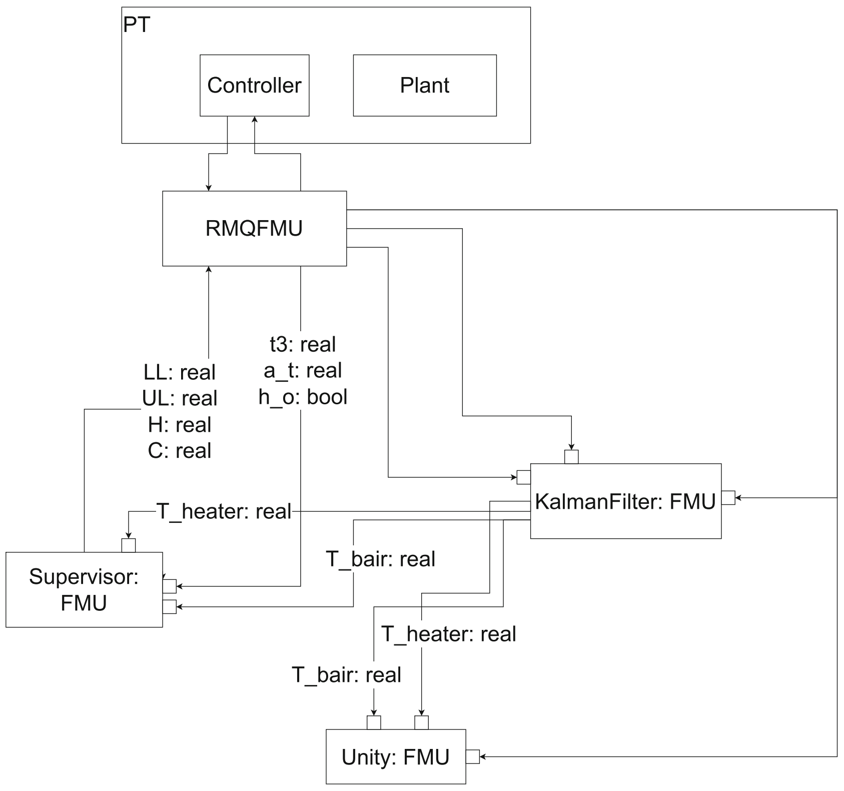


FIGURE 10 Co-simulation scenario for the incubator case study.

3.5 | INTO-CPS co-simulation framework FMI based toolbox

This framework leverages the FMI standard as the communication medium between the different DTs services. Because the FMI standard was created to enable time-stepped simulation of CPSs, it is expected that every DT service communicates with its neighbors continuously. This time-based communication paradigm is different in other frameworks, where services are expected to communicate when they are triggered (following an event-based paradigm). On the other hand, it makes reasoning about the DT easier, as the connections between services are determined statically and can be illustrated in a diagram, as presented in Figure 10.

This shows the different DT services, implemented as FMUs, and how they are connected. As can be seen, the Controller exchanges messages with the RabbitMQ FMU, who converts those messages into piece-wise constant signals that are exchanged in the co-simulation. Maestro2 is the tool that coordinates the co-simulation. When an anomaly is detected, the supervisor FMU changes one of its outputs (each output of the supervisor represents a parameter of the controller). That change causes a new message to be sent from the Rabbitmq FMU to the controller, which in turn updates the parameter. The RabbitMQ FMU is not constantly sending messages to the controller. It only sends a message when one of its inputs has changed more than a configured threshold. This is effectively converting the continuous signals of the co-simulation into quantized signals.[50] Other FMU blocks, such as the Unity FMU, which displays a 3D animation of the incubator, such as shown in Figure 11 or the KalmanFilter FMU, which implements a state estimator of the system, can be easily integrated to the system thanks to the FMI standard.

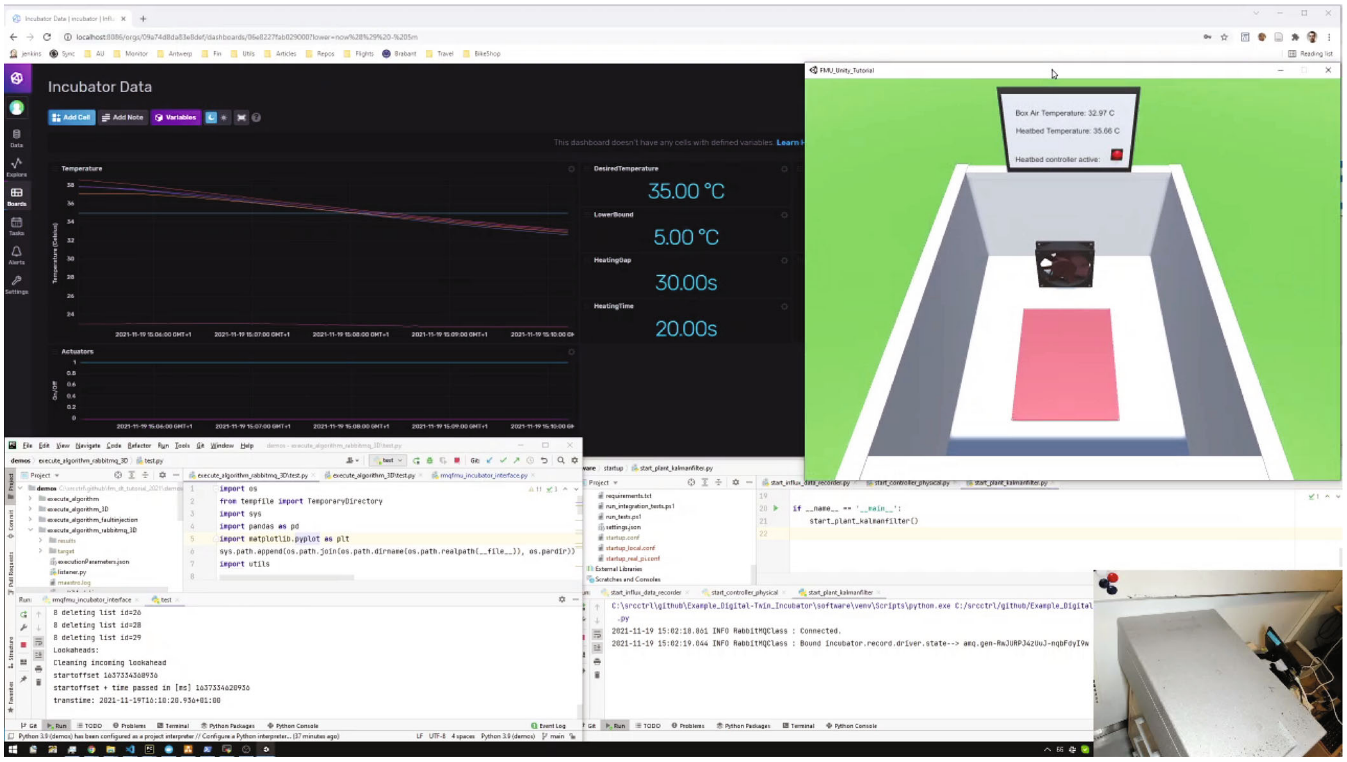
The reader is encouraged to visit the available demo online.[‡‡] [§§]

3.6 | Comparison to the incubator digital twin

To further analyze the selected five frameworks for the case study approach, we compare each of the implementations with the *Incubator DT*,[24,51] which is available at https://github.com/INTO-CPS-Association/example_digital-twin_incubator.

‡‡https://sites.google.com/view/fm2021tutorialdt/home.

§§https://github.com/INTO-CPS-Association/fm_dt_tutorial_2021.


Incubator Data
Temperature
Temperature (Celsius)
Actuators
On/Off
Box Air Temperature: 32.97 C
Heatbed Temperature: 35.66 C
Heatbed controller active:
FMU_Unity_Tutorial

DesiredTemperature 35.00 °C
LowerBound 5.00 °C
HeatingGap 30.00s
HeatingTime 20.00s

```python
import os
from tempfile import TemporaryDirectory
import sys
import pandas as pd
import matplotlib.pyplot as plt
sys.path.append(os.path.join(os.path.dirname(os.path.realpath(__file__)), os.pardir))
import utils
```

```
8 deleting list id=26
8 deleting list id=28
8 deleting list id=29
Lookaheads:
Cleaning incoming lookahead
startoffset 1637334368936
startoffset + time passed in [ms] 1637334628936
transtime: 2021-11-19T16:10:28.936+01:00
```

Python 3.9 (demos) has been configured as a project interpreter // Configure a Python interpreter... (37 minutes ago)

```python
if __name__ == '__main__':
    start_plant_kalmanfilter()
```

```
C:\srcctrl\github\Example_Digital-Twin_Incubator\software\venv\Scripts\python.exe C:/srcctrl/github/Example_Digital
 .py
2021-11-19 15:02:18.861 INFO RabbitMQClass : Connected.
2021-11-19 15:02:19.844 INFO RabbitMQClass : Bound incubator.record.driver.state--> amq.gen-RwJURPJ4zUuJ-nqbFdyI9w
```

FIGURE 11 Demo of the incubator case study using INTO-CPS co-simulation framework FMI based toolbox.

This DT is designed in such a way that all the models, tools, and services are specifically focused on the incubator case study using a service-oriented architecture.

To do so, we first describe the services that are integrated into the incubator DT, and then, we compare which of the services are fulfilled by the five candidates based on our experience.

The incubator DT services can be summarized as follows:

1. **Real-time mock-up (S1)**: This service implements a real-time plant simulation mock-up that represents the active PE.
2. **Real-time visualization (S2)**: This service provides a graphical interface for the user to understand the current state at a glance.
3. **Plant Kalman filter (S3)**: This service runs a Kalman filter to estimate the full state of the incubator system to be used in anomaly detection.
4. ***What-if*** *simulation (S4)*: This service runs simulations based on historical data with different conditions for the plant and controller models.
5. **Physical controller (S5)**: This service implements the controller of the incubator.
6. **Anomaly detector (S6)**: This service periodically monitors the incubator DT and alerts in case of any anomaly.
7. **Self-adaptation manager (S7)**: This service implements a self-adaptation process that checks whether the characteristics of the system have changed and triggers a re-calibration of the models and the controller.

Table 4 provides an analysis of the capabilities of the candidates, which were selected for the case study implementation, based on the services (***Sx***) to be provided by the *ideal* Incubator DT. Since most of the services are simulation-driven, namely, integration of the plant simulation, the Kalman filter for state estimation, *what-if* analysis, and monitoring and action based on deviations of the PE compared to the DT models, the framework implementation that better suits this case study is the INTO-CPS co-simulation framework. However, it is worth highlighting the advantages of the other framework implementations, such as the easy PE-to-DT interfacing with SAP I4.0 AAS, Eclipse BaSyx, and Eclipse Ditto, including its capability to embed the physical controller behavior; and the visualization and physics/geometry capabilities of iTwin, which are highly relevant for DT engineering. It is also worth mentioning that although the INTO-CPS co-simulation

TABLE 4 Qualitative analysis of the capabilities of the selected frameworks based on the incubator DT services (● = *Well-covered*, ◐ = *Moderately-covered*, ○ = *Poorly- or non-covered*).

Framework implementation	S1	S2	S3	S4	S5	S6	S7	Advantages	Limitations
SAP I4.0 AAS	◐	○	○	○	●	◐	○	It provides a dockerized DT where it is possible to create some rules and store the state of and interact with the DT and PE	No integration of simulations or behavioral models
Eclipse BaSyx	◐	○	○	○	●	◐	○	It is possible to use the SDK to add some programmatic features for the DT. It is possible store the state of and interact with the DT and PE easily	The integration of simulations or behavioral models needs to be done externally or programmatically and may not be trivial
Eclipse Ditto	◐	○	○	○	●	◐	○	It is possible to use the SDK to add some programmatic features for the DT. It is possible store the state of and interact with the DT and PE easily	The integration of simulations or behavioral models needs to be done externally or programmatically and may not be trivial
iTwin	◐	●	○	◐	○	◐	○	Easy integration of visualization and physics/geometry behind the models	The PE-to-DT interfacing may not be trivial. Some services may require external integrations and deal with the constraints of this framework due to its category
INTO-CPS	◐	◐	●	●	◐	◐	◐	It is highly modular thanks to the FMI standard. Most of the services can be provided by orchestrating specific FMUs	The PE-to-DT interfacing may not be trivial and some FMUs need to be designed for the particular application

framework perfomed the better, it has some difficulties when interfacing the PE to the DT bi-directionally, and that some of the FMUs that are used, need to be tailored to the case study, which may reduce their reusability and generality.

4 | DISCUSSION

Throughout the development of this bottom-up survey a number of advantages, disadvantages, challenges, and opportunities were identified in relation to open-source technologies for DTs. First, it is important to mention that open-source tools can have problems regarding long-term support, number of releases, superfluous functionalities, and performance. Compared to other commercial solutions, they might not be as promising or stable, but they are open for modifications, customizations, and research purposes. It can also be the case that there is no commercial solution of a new technology before several minor releases of open-source contributions become available. Open-source contributions can be high quality when backed-up with lifetime storage and policies to ensure the long-term support; some renowned foundations, such as the Eclipse Foundation, Apache, or the Linux Foundation, offer the warranties to deal with this matter.

It it also worth mentioning that the explored frameworks offer different features and functionalities for different industrial domains, therefore, choosing the right one according to specific requirements is key to successful adoption of DTs. In our case, the incubator case study fitted in five out of the 14 frameworks, where the implementations took place. Certainly, the solutions can be complementary, and so, they can be combined for particular situations if needed. External tools that are available for DTs or IoT also have room for being combined with these frameworks.

4.1 | Main advantages and limitations

Across the exploration of the different frameworks, it is possible to highlight some of the following advantages:

1. Ditto offers a well-defined metastructure for IoT and Web of Things applications with a robust communication infrastructure and integrability.
2. Equinox offers a specialized set of tools for aeronautics and aerospace engineering.
3. AAS implementations offer a well-defined metastructure for industrial automation applications with a wide range of interoperable tools and is aligned with several standards at the industrial level.
4. CPS-Twinning offers the inclusion of AutomationML for DTs in simulated networks and is intended to cover physical simulations in its second version.
5. Twined and DTDL offer a common schema for defining DT structures and have potential subsequent use with commercial solutions.
6. iTwin.js offers an all-in graphical package for 3D visualizations of DTs or DSs that is mainly used for infrastructure applications but can be extended to other domains if required.
7. DTCC and Terria JS are both foundation geospatial data frameworks that can be used for DT application considering the location and the integration with several layers on maps.
8. INTO-CPS offers a different set of tools for co-simulation that can be extended to DT application with certain requirements, but it has an advantage in what refers to high-fidelity modeling and integration with FMU blocks.

As the frameworks have some advantages with respect to some features, they also have some drawbacks, which are mainly categorized as lack of documentation, lack of examples (or the provided examples are too simplistic), lack of public case studies, lack of long-term support, complexity, lack of extensibility to other domains, and lack of openness for integration with other tools. The limitations are worse for some of the tools than the others, something that can be represented by the *Reproducibility* field of Table 3.

4.2 | Capabilities regarding built-in simulations and data analytics

As DT frameworks, it is expected that these can provide additional insights when a DT is up and running. Unfortunately, the current advances of built-in simulations and data analytics are at a minimum level of development. APIs are so far, the common *analytics* method for most of the frameworks, where the user can access the data and perform specific hard analytic tasks. As it can be seen in Table 2, the support for analytics in open-source frameworks is basic. At maximum, a level 2, meaning integrated dashboards or basic analytic tasks are provided. There is a challenge and opportunity regarding the creation of *out-of-the-box* integrated analytics for DTs in DT frameworks.

DT frameworks that provide a metastructure for defining the twins lead the way on this matter, since it is possible to generate automated inferences and reasoning based on a common reusable structure that even a machine knows what that stands for.

The same applies for built-in simulations because it is expected that a DT runs along with models and simulations, but most of the explored frameworks do not have a simulation engine to run the models. In this case, a tool that is able to run simulations, such as INTO-CPS, TerriaJS, and DTCC, leads the way. For some other cases, it is possible to integrate the DT framework with external simulation engines, but this is not *out-of-the-box* and requires manual setup.

Alternatives for bridging the gap between *structured data DT frameworks* and *Co-simulation and Model-based DT frameworks* need to be approached, such as for example, the one presented in Reference 52 to integrate FMI-based simulations into DT platforms.

The frameworks based on the AAS specification, provide a meta data structure and semantics for managing assets. The recommendations given in the documentation[40] does however not provide a concise way to model different data sources. A submodel element represents an attribute, but one may wish to have an initial value, an operational value, but also an estimated value and maybe more, but AAS does not specify how to manage this, and it will thus be difficult to derive knowlegde across a system of assets from different providers.

4.3 | From theory to practice

An important aspect to be analyzed when discussing the theory-to-practice transition of DTs is the orchestration of services.[51] DTs usually need the orchestration of multiple components, including models, enablers, and usages, that is, the DT *constellation*. A DT constellation can be difficult to set up from scratch, especially when the DTs are designed via composition to enable reusability. In this sense, methods and tools from model-driven engineering may be adopted to drive the generation of system descriptions and DevOps tools can benefit the continuous evolvement, integration, and deployment of DTs.[53]

Another relevant matter is that some DT frameworks (and DT implementations) focus mainly on fulfilling the requirement for bi-directional communication. This is especially evident for the DT frameworks that are built on top of IoT frameworks, that is, the *Structured data DT frameworks*, with some additional extensions to include data models and the actual bi-directional communication. However, if a DT is intended to have a usage, that is, business goal,[54] the DT needs to be backed-up by simulation models, inference and reasoning mechanisms, and data analysis. Therefore, incorporating these kinds of modules is essential for a DT to fulfill its purpose.

And the other way around, the DT is not only about heavy computations either. The DT should contain the right models for a particular scope and goal.[54,55] Even if multiplicity of DTs is a controversial topic,[34] the coexistence can lead to a more seamlessly development with specialized views of different parts of a system. Multiple DTs featuring the same PE can be used for internal comparison and testbeds of multiple *what-if* scenarios.[56]

On the same agenda, DTs are supposed to fulfill a particular business goal, which is then derived into usages and services, like optimization, task allocation, monitoring, and so forth. These usages may imply either indirect or direct or both actions of the DT on the PE, which is another controversial topic.[34] Nevertheless, these usages are not easy to implement at all, and are usually designed and developed specifically for a particular DT-to-PE pair. However, the ideal scenario is that these usages are leveraged from modules, which need to be easily parameretized for multiple, if not any, DT-to-PE pairs. This idea goes in connection with using model-driven and DevOps mechanisms to configure and deploy DTs.

Finally, using composable DTs enhances the reusability of components.[34] Therefore, approaching the DT engineering process using composable DTs is recommended, which also provides more flexibility to set up large DT systems.[57]

4.4 | Open-source versus commercial initiatives

Throughout this survey we could identify some advantages and disadvantages of selecting open-source software for DT applications, and it is a relevant point to discuss. This is a decision that has to be taken according to project requirements, budget, and scope. Naturally, commercial software offers more stability and support than open-source software, but generally, they also create some dependencies between manufacturers that can affect negatively the projects at the mid- or long-term. However, they offer *out-of-the-box* functionalities and modules that, perhaps, open-source software does not, and so, could be more useful for quick starts and stability of the project during the different phases. Commercial software is also often backed-up with better support, which can be useful when dealing with the software. On the other hand, the open-source software offers more flexibility at the expense of less stability and support. It breaks the barriers of having low budget for SMEs and start-ups. It could also take more time and can be difficult to handle for the first phases of projects. In the mid- or long-term, they can be easily modified in case of changes or adapted to new requirements with less troubles than commercial software. Finally, if the open-source software is designed based on standards it offers more reliability for the long-term, since commercial software can take even more time to be settled in the market.

4.5 | Limitations of the study

This study had known limitations that are worth mentioning and can work as a reference for future works in this kind of surveys on DT frameworks.

1. First, this survey was limited to analyze only open-source software, so we did not consider commercial software or commercial extensions of the explored DT frameworks. This is a limitation with respect to potential available solutions in the market. This can also be considered an opportunity for SMEs to identify and use potential DT tools and get on board in the DT community with a low budget.
2. Second, this kind of survey is useful for a few years, because tools are a *moving target*, especially, because are associated to software systems. Therefore, this survey can become obsolete when new tools or even new features are released.
3. Third, few tools were analyzed in this survey, so it does not cover a wide spectrum of available DT frameworks and tools in the current market. It was mainly because of the limitation of surveying only open-source software that allows

creating DTs upon a common framework and not a collection of individual tools. However, there are other available tools that can be found on the web and can be used for several specific functionalities in DT applications.

4. Fourth, some of the analyses in this survey were made from the authors' perspective, which may lead to potential bias.

We acknowledge that there are other DT tools and frameworks, including commercial and open-source, which are worth exploring. Some of the other relevant complementary open-source tools retrieved along the search but not explored include Fiware (https://www.fiware.org/), IoTivity (https://iotivity.org/), EdgeX Foundry (https://www.edgexfoundry.org/), Node-RED (https://nodered.org/), InfluxDB (https://www.influxdata.com/get-influxdb/), Grafana (https://grafana.com/), Apache Kafka (https://kafka.apache.org/), Eclipse Vorto (https://eclipse.dev/vorto/), and Eclipse Hono (https://projects.eclipse.org/projects/iot.hono). Some of the complementary commercial tools include General Electric's Digital Twin Framework (https://www.ge.com/research/project/digital-twin-framework), Azure Digital Twins (https://learn.microsoft.com/en-us/azure/digital-twins/), Ansys Twin Builder (https://www.ansys.com/products/digital-twin/ansys-twin-builder), and Unity Digital Twins (https://unity.com/solutions/digital-twins). The reader is encouraged to further explore and use these tools for DT engineering.

5 | RELATED WORK

There are some previous works that have worked on the analysis of tools for DTs. Qi et al.[58] discuss and analyze several different DT technologies and tools from different perspectives, including connectivity, modeling, data management, and interfacing with the physical world. They cover a wide range of tools that can be used for DT applications with a certain degree of coding or hand work. This survey does not distinguish the tools between commercial or open-source solutions. Another systematic review is presented by Tao et al.,[49] where six modeling aspects and their corresponding enabling technologies and tools are analyzed.

Lehner et al.[46] analyze three different DT platforms, namely, Microsoft Azure Digital Twins, Amazon Web Services IoT Greengrass, and Eclipse Hono + Vorto + Ditto. This survey analyzes the platforms in 10 different dimensions each one with a requirement. Two thirds of the frameworks are commercial solutions.

Liu et al.[28] analyze DTs regarding concepts, frameworks, applications, and technologies. This review provides insights regarding key technologies for DT in the dimensions of data representation, modeling, and environments to run simulations for DT applications.

Pribiš et al.[59] analyze different implementations of AAS and provide a new one for embedded systems running on ARM-based microcontrollers.

Picone et al.[60] develop WLDT, a Java library that acts as a middleware for agent-based DT applications in the scope of IoT, and compare their tool with Eclipse Ditto regarding the overhead delay performance. They also provide some implementations with their tool in public case studies and show the integration with different communication protocols and external tools.

Autiosalo et al.[47] summarize some open-source implementations for DT applications at the metadata level and propose an open-source web server implementation to store, manage, and distribute DT documents based on a git-based architecture for DTs.

The surveyed tools have been used in different DT setups in the literature, including iTwin in Reference 61, Eclipse BaSyx in Reference 62, Eclipse Ditto in References 63,64, TerriaJS in Reference 65, DTCC in Reference 66, and DTDL in Reference 67. A DT for a similar case study, a thermal chamber, was created under a framework for DTs with open-source tools, where Eclipse Ditto was used as the interface between PE and PT.[68] The other open-source software tools used for this case study are FreeCAD for the CAD modeling, OpenFOAM and SimFlow for the simulation of fluid dynamics, and ParaView for visualization.

While the above cited surveys have a broader focus on technologies and tools than the surveys cited in the introduction, none of the above compares multiple DT frameworks under the same case study. This survey considers tools for different domains and industries that are available on the web and implements a basic case study with some of the explored ones. The tools are analyzed and categorized in 10 different dimensions that can be relevant and of interest for distinct audiences. It also shows potential benefits of each of the explored tools according to requirements and scope of the applications.

Table 5 provides a summary of the main similarities and differences of the complementary studies identified as related work.

TABLE 5 Summary of selected literature review on DT frameworks.

Approach	Emphasis	Similarities	Differences
Qi et al. [58]	Technologies and tools for DTs	Covers aspects in relation to interfacing with the physical world, behavioral modeling, simulation, data analytics, and integrated DT services	The review of tools and technologies is descriptive. Many of the listed tools are commercial. The analyzed tools are part of the DT constellation of enablers, rather than actual DT frameworks
Lehner et al. [46]	DT Platforms	Some of the analyzed dimensions are the same or similar. Eclipse Ditto is also surveyed. The requirements are leveraged from a case study approach	The review focuses on IoT-based DT frameworks. No analysis in terms of simulation and data analytics
Liu et al. [28]	Theoretical aspects of DTs	Reviews key technologies in relation to data, modeling, and simulation	Technologies, tools, and methods are mixed. No particular framework is further studied. The review and categorization of technologies is descriptive
Pribiš et al. [59]	Asset Administration Shell tools	Analyzes four of the AAS frameworks, AASX, Eclipse BaSyx, SAP I4.0 AAS, and NOVAAS. It also covers the aspects for interoperability and storage	Focuses only on AAS frameworks and is limited to APIs, infrastructure, and integration with OPC UA at the practical level
Autiosalo et al. [47]	DT implementations at the meta-data level	Discusses and analyzes the frameworks DTDL and AAS	Is limited to meta-data DT solutions
Shah et al. [68]	A DT framework based on open-source software tools	Uses a case study approach and analyzes complementary tools for DTs including simulation modeling. Uses Eclipse Ditto as the interface between PE and DT	Does not analyze other tools or frameworks for the orchestration of DTs

6 | CONCLUSIONS AND FUTURE WORK

This paper analyzed a special niche of DTs regarding open-source framework for the creation of generic and extensible solutions for DT applications in multiple domains. It assessed different aspects that are aligned with consensual criteria in the field of DTs and went deeper to explore functionalities, practicalities, and capabilities of some of the tools. It analyzed, categorized, and discussed the information and the theory-to-practice transition with the purpose of enabling practitioners to quickly get started with some of the open-source DT frameworks that are available. It also enables the reader to decide which tool fits a specific application the best based on domain, requirements, and scope.

During the execution of this bottom-up survey, some of the found gaps were regarding the expected features that a DT should contain and what a DT framework should provide, including: (1) limitations of integrated high-fidelity models, simulation, and data analytics at the generic scale; (2) lack of documentation, examples, tutorials, and community support that allow new users to get used to a new environment and to create advanced DTs in complex scenarios, and not just simplistic demos; and (3) limited coverage of common features for DTs, expressed as strengths in some areas, that can be, for example, communication and data modeling, but as weaknesses in other areas, such as high-fidelity physical modeling and simulations.

It is worth highlighting that the *Support for Analytics* criterion has room for improvement in terms of *out-of-the-box* features. These features can be interpreted as DT services, for example, a DT service for built-in analytics, which are expected to be provided in DT applications, and can be either generated at the application level or at the framework level as standard software packages or modules. In general, there is room for research in built-in DT services both at the application level and at the framework level and it should be studied in depth in future work.

The findings of this survey led the authors to propose yet another framework for DT engineering within the DT-as-a-Service (DTaaS) concept, presented by Talasila et al.[69] as a DTaaS platform. This framework is still under development and a future work in relation to DT frameworks with built-in DT services for simulation and reusable modules.

Some of the limitations of this research that were identified include the boundary to survey only open-source DT frameworks of an indeed fast-moving field. The number of tools was also low, so there is room for a newer similar survey when the technology market of DTs has expanded enough. There is also room for contributions on DT safety and security aspects from the software framework perspective. Finally, there is a need for research into how automated reasoning and analytics can be configured and managed at the generic level for DT frameworks, that at the moment are hardly implemented.

Some frameworks, such as the AAS-based, support meta data structures that in principle can enable this kind of reasoning, but semantics and methods are in no way standardized. For instance, the meta structure supports assets, submodels and submodel elements, which can be nested, but there is no guidance of how to structure and deal with this; for instance, when there is static, live, and simulated data. Security aspects, both at the cybersecurity and safety level, are also interesting to be researched in future work. This is relevant since the DTs are in direct contact with the machinery and security is a criterion covered by the ISO 23247:2021 standard, but it is not fully considered by open-source frameworks as a main component.

AUTHOR CONTRIBUTIONS

Santiago Gil contributed with collecting the candidate frameworks, proposing the taxonomy for evaluating the frameworks, assessing and documenting the frameworks' capabilities, implementing the case study, analyzing and discussing capabilities and categories, and writing the draft. **Peter H. Mikkelsen** contributed with documenting the frameworks' capabilities, implementing the case study, and reviewing and editing of the final document. **Cláudio Gomes** contributed with validating the candidate frameworks, validating the taxonomy for evaluating the frameworks, documenting the frameworks' capabilities, implementing the case study, and reviewing and editing of the final document. **Peter G. Larsen** contributed with validating the analyses, project supervision, and reviewing and editing of the final document. All authors have read and approved the manuscript.

ACKNOWLEDGMENTS

The authors would like to thank Matthias Eckhart, author of CPS-Twinning; Anders Logg and Vasilis Naserentin, authors of DTCC; Tom Clark, author of Twined; and iTwin's support team for their help in better understanding and setting up the environments for the trials of the case study. The authors would also like to thank Hao Feng, Prasad Talasila, and Till Böttjer for their contributions to this study. This work was partially funded by 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.

CONFLICT OF INTEREST STATEMENT

The INTO-CPS Co-simulation Framework is maintained by the INTO-CPS Association, currently hosted by Aarhus University. The authors acknowledge this creates a potential bias that may influence the readers' perception of the tool's performance, integrity, or relevance. We have conducted a comprehensive and unbiased evaluation of the INTO-CPS tool and have presented the results transparently in this paper, ensuring that the research is as objective as possible.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID

Santiago Gil https://orcid.org/0000-0002-1789-531X
Peter H. Mikkelsen https://orcid.org/0000-0003-2321-758X
Cláudio Gomes https://orcid.org/0000-0003-2692-9742
Peter G. Larsen https://orcid.org/0000-0002-4589-1500

REFERENCES

1. Grieves M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. 2015.

2. Terzi S, Bouras A, Dutta D, Garetti M, Kiritsis D. Product lifecycle management–from its history to its new role. *Int J Product Lifecycle Manag*. 2010;4(4):360-389. doi:10.1504/IJPLM.2010.036489
3. Negri E, Fumagalli L, Macchi M. A review of the roles of digital twin in CPS-based production systems. *Procedia Manuf*. 2017;11:939-948. doi:10.1016/j.promfg.2017.07.198
4. Kritzinger W, Karner M, Traar G, Henjes J, Sihn W. Digital twin in manufacturing: a categorical literature review and classification. *IFAC-PapersOnLine*. 2018;51(11):1016-1022. doi:10.1016/j.ifacol.2018.08.474
5. Cheng Y, Zhang Y, Ji P, Xu W, Zhou Z, Tao F. Cyber-physical integration for moving digital factories forward towards smart manufacturing: a survey. *Int J Adv Manuf Technol*. 2018;97(1-4):1209-1221. doi:10.1007/s00170-018-2001-2
6. Park H, Easwaran A, Andalam S. Challenges in digital twin development for cyber-physical production systems. In: Chamberlain R, Taha W, Törngren M, eds. *Cyber Physical Systems. Model-Based Design*. Vol 11615. Springer International Publishing; 2019:28-48. doi:10.1007/978-3-030-23703-5_2
7. Zhang H, Ma L, Sun J, Lin H, Thürer M. Digital twin in services and industrial product service systems. *Procedia CIRP*. 2019;83:57-60. doi:10.1016/j.procir.2019.02.131
8. Aivaliotis P, Georgoulias K, Alexopoulos K. Using digital twin for maintenance applications in manufacturing: state of the art and gap analysis. Paper presented at: 2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), IEEE: Valbonne Sophia-Antipolis, France. 2019:1–5. doi:10.1109/ICE.2019.8792613
9. Kutin AA, Bushuev VV, Molodtsov VV. Digital twins of mechatronic machine tools for modern manufacturing. *IOP Conf Ser: Mater Sci Eng*. 2019;568:012070. doi:10.1088/1757-899X/568/1/012070
10. Bradac Z, Marcon P, Zezulka F, Arm J, Benesl T. Digital twin and AAS in the industry 4.0 framework. *IOP Conf Ser: Mater Sci Eng*. 2019;618:012001. doi:10.1088/1757-899X/618/1/012001
11. Cimino C, Negri E, Fumagalli L. Review of digital twin applications in manufacturing. *Comput Ind*. 2019;113:103130. doi:10.1016/j.compind.2019.103130
12. Lu Y, Liu C, Wang KIK, Huang H, Xu X. Digital twin-driven smart manufacturing: connotation, reference model, applications and research issues. *Robot Comput-Integr Manuf*. 2020;61:101837. doi:10.1016/j.rcim.2019.101837
13. Tao F, Zhang H, Liu A, Nee AYC. Digital twin in industry: state-of-the-art. *IEEE Trans Industr Inform*. 2019;15(4):2405-2415. doi:10.1109/TII.2018.2873186
14. Arrichiello V, Gualeni P. Systems engineering and digital twin: a vision for the future of cruise ships design, production and operations. *Int J Interact Design Manuf*. 2020;14(1):115-122. doi:10.1007/s12008-019-00621-3
15. Marmolejo-Saucedo JA, Hurtado-Hernandez M, Suarez-Valdes R. Digital twins in supply chain management: a brief literature review. In: Vasant P, Zelinka I, Weber G-W, eds. *Intelligent Computing and Optimization*. Vol 1072; Springer; 2020:653-661. doi:10.1007/978-3-030-33585-4·63
16. Lim KYH, Zheng P, Chen CH. A state-of-the-art survey of digital twin: techniques, engineering product lifecycle management and business innovation perspectives. *J Intell Manuf*. 2020;31(6):1313-1337. doi:10.1007/s10845-019-01512-w
17. Barricelli BR, Casiraghi E, Fogli D. A survey on digital twin: definitions, characteristics, applications, and design implications. *IEEE Access*. 2019;7:167653-167671. doi:10.1109/ACCESS.2019.2953499
18. Raj P, Surianarayanan C. Digital twin: the industry use cases. *Advances in Computers*. Vol 117. Elsevier; 2020:285-320. doi:10.1016/bs.adcom.2019.09.006
19. Josifovska K, Yigitbas E, Engels G. Reference framework for digital twins within cyber-physical systems. Paper presented at: 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), IEEE: Montreal, QC, Canada. 2019:25–31. doi:10.1109/SEsCPS.2019.00012
20. Macchi M, Roda I, Negri E, Fumagalli L. Exploring the role of digital twin for asset lifecycle management. *IFAC-PapersOnLine*. 2018;51(11):790-795. doi:10.1016/j.ifacol.2018.08.415
21. International Organization for Standardization. *Automation Systems and Integration — Digital Twin Framework for Manufacturing*. ISO 23247:2021(E). International Organization for Standardization; 2021 https://www.iso.org/standard/78743.html
22. Barnstedt E, Lin SW, Boss B, et al. Open source drives digital twin adoption. *IIC J Innov*. 2021;March 2021:1-16. https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1818
23. Feng H, Gomes C, Thule C, Lausdahl K, Iosifidis A, Larsen PG. Introduction to digital twin engineering. Paper presented at: 2021 Annual Modeling and Simulation Conference (ANNSIM), IEEE: Fairfax, VA, USA. 2021:1–12. doi:10.23919/ANNSIM52504.2021.9552135
24. Feng H, Gomes C, Thule C, Lausdahl K, Sandberg M, Larsen PG. The incubator case study for digital twin engineering. arXiv:2102.10390 [cs, eess]. 2021.
25. Paredis R, Gomes C, Vangheluwe H. Towards a family of digital model/shadow/twin workflows and architectures. Proceedings of the 2nd International Conference on Innovative Intelligent Industrial Production and Logistics, SCITEPRESS–Science and Technology Publications: Online Streaming. 2021:174–182. doi:10.5220/0010717600003062
26. Moyne J, Qamsane Y, Balta EC, et al. A requirements driven digital twin framework: specification and opportunities. *IEEE Access*. 2020;8:107781-107801. doi:10.1109/ACCESS.2020.3000437
27. Shao G, Helu M. Framework for a digital twin in manufacturing: scope and requirements. *Manuf Lett*. 2020;24:105-107. doi:10.1016/j.mfglet.2020.04.004
28. Liu M, Fang S, Dong H, Xu C. Review of digital twin about concepts, technologies, and industrial applications. *J Manuf Syst*. 2021;58:346-361. doi:10.1016/j.jmsy.2020.06.017

29. Fuller A, Fan Z, Day C, Barlow C. Digital twin: enabling technologies, challenges and open research. *IEEE Access*. 2020;8:108952-108971. doi:10.1109/ACCESS.2020.2998358
30. Leng J, Wang D, Shen W, Li X, Liu Q, Chen X. Digital twins-based smart manufacturing system design in industry 4.0: a review. *J Manuf Syst*. 2021;60:119-137. doi:10.1016/j.jmsy.2021.05.011
31. He B, Bai KJ. Digital twin-based sustainable intelligent manufacturing: a review. *Adv Manuf*. 2021;9(1):1-21. doi:10.1007/s40436-020-00302-5
32. Oakes BJ, Parsai A, Van Mierlo S, et al. Improving digital twin experience reports. MODELSWARD 2021–Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development, Modelsward. 2021:179–190. doi:10.5220/0010236101790190
33. Autiosalo J, Vepsalainen J, Viitala R, Tammi K. A feature-based framework for structuring industrial digital twins. *IEEE Access*. 2020;8:1193-1208. doi:10.1109/ACCESS.2019.2950507
34. Dalibor M, Jansen N, Rumpe B, et al. A cross-domain systematic mapping study on software engineering for digital twins. *J Syst Softw*. 2022;193:111361. doi:10.1016/j.jss.2022.111361
35. Human C, Basson AH, Kruger K. A design framework for a system of digital twins and services. *Comput Ind*. 2023;144:103796. doi:10.1016/j.compind.2022.103796
36. Antonino PO, Capilla R, Kazman R, et al. Continuous engineering for industry 4.0 architectures and systems. *Softw Pract Exper*. 2022;52(10):2241-2262. doi:10.1002/spe.3124
37. Bracke V, Sebrechts M, Moons B, Hoebeke J, De Turck F, Volckaert B. Design and evaluation of a scalable internet of things backend for smart ports. *Softw Pract Exper*. 2021;51(7):1557-1579. doi:10.1002/spe.2973
38. Atalay M, Murat U, Oksuz B, Parlaktuna AM, Pisirir E, Testik MC. Digital twins in manufacturing: systematic literature review for physical–digital layer categorization and future research directions. *Int J Comput Integr Manuf*. 2022;35(7):679-705. doi:10.1080/0951192X.2021.2022762
39. Plattform Industrie 4.0. Reference Architectural Model Industrie 4.0 (RAMI 4.0)-an Introduction. Technical Report, ZVEI-German Electrical and Electronic Manufacturers Association. 2016 https://web.archive.org/web/20210615073607/. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf?__blob=publicationFile&v=7
40. ZVEI. Details of the asset administration shell: part 1-the exchange of information between partners in the value chain of industrie 4.0. Berlin, Germany. 2020 https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1-V3.html
41. Thule C, Lausdahl K, Gomes C, Meisl G, Larsen PG. Maestro: the INTO-CPS co-simulation framework. *Simul Model Pract Theory*. 2019;92:45-61. doi:10.1016/j.simpat.2018.12.005
42. Gomes C, Thule C, Broman D, Larsen PG, Vangheluwe H. Co-simulation: a survey. *ACM Comput Surv*. 2018;51(3):49:1-49:33. doi:10.1145/3179993
43. Blochwitz T, Otter M, Akesson J, et al. Functional Mockup Interface 2.0: the standard for tool independent exchange of simulation models. Paper presented at: 9th International Modelica Conference, Linköping University Electronic Press: Munich, Germany. 2012:173–184. doi:10.3384/ecp12076173
44. Frasheri M, Ejersbo H, Thule C, Esterle L. Rmqfmu: bridging the real world with co-simulation for practitioners. In: Macedo HD, Thule C, Pierce K, eds. *Proceedings of the 19th International Overture Workshop*. Overture; 2021.
45. Gomes C, Meyers B, Denil J, et al. Semantic adaptation for fmi co-simulation with hierarchical simulators. *Simulation*. 2019;95(3):241-269. doi:10.1177/0037549718759775
46. Lehner D, Pfeiffer J, Tinsel EF, et al. Digital twin platforms: requirements, capabilities, and future prospects. *IEEE Softw*. 2022;39(2):53-61. doi:10.1109/MS.2021.3133795
47. Autiosalo J, Siegel J, Tammi K. Twinbase: open-source server software for the digital twin web. *IEEE Access*. 2021;9:140779-140798. doi:10.1109/ACCESS.2021.3119487
48. Coetzee S, Ivánová I, Mitasova H, Brovelli MA. Open geospatial software and data: a review of the current state and a perspective into the future. *ISPRS Int J Geo Inf*. 2020;9(2):1-30. doi:10.3390/ijgi9020090
49. Tao F, Xiao B, Qi Q, Cheng J, Ji P. Digital twin modeling. *J Manuf Syst*. 2022;64:372-389. doi:10.1016/j.jmsy.2022.06.015
50. Kofman E, Junco S. Quantized-state systems: a DEVS approach for continuous system simulation. *Trans Soc Model Simul Int*. 2001;18(3):123-132.
51. Feng H, Gomes C, Gil S, et al. Integration of the Mape-K loop in digital twins. Paper presented at: Annual Modeling and Simulation Conference (ANNSIM 2022), IEEE. 2022:102–113. doi:10.23919/annsim55834.2022.9859489
52. Lehner D, Gil S, Mikkelsen PH, Larsen PG, Wimmer M. An architectural extension for digital twin platforms to leverage behavioral models. Paper presented at: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), IEEE. 2023:1–8. doi:10.1109/CASE56687.2023.10260417
53. Fritzsch J, Bogner J, Haug M, et al. Adopting microservices and DevOps in the cyber-physical systems domain: a rapid review and case study. *Softw Pract Exper*. 2023;53(3):790-810. doi:10.1002/spe.3169
54. VanDerHorn E, Mahadevan S. Digital twin: generalization, characterization and implementation. *Decis Support Syst*. 2021;145:113524. doi:10.1016/j.dss.2021.113524
55. Oakes B, Gomes C, Larsen P, et al. Examining model qualities and their impact on digital twins. Paper presented at: 2023 Annual Modeling and Simulation Conference (ANNSIM), IEEE Computer Society. 2023:220–232 https://doi.ieeecomputersociety.org/
56. Schluse M, Priggemeyer M, Atorf L, Rossmann J. Experimentable digital twins-streamlining simulation-based systems engineering for industry 4.0. *IEEE Trans Industr Inform*. 2018;14(4):1722-1731. doi:10.1109/TII.2018.2804917

57. Gil S, Mikkelsen PH, Tola D, Schou C, Larsen PG. A modeling approach for composed digital twins in cooperative systems. Paper presented at: 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE. 2023:1–8. doi:10.1109/ETFA54631.2023.10275601
58. Qi Q, Tao F, Hu T, et al. Enabling technologies and tools for digital twin. *J Manuf Syst*. 2021;58:3-21. doi:10.1016/j.jmsy.2019.10.001
59. Pribiš R, Beňo L, Drahoš P. Asset administration shell design methodology using embedded opc unified architecture server. *Electron (Switzerland)*. 2021;10(20):1-17. doi:10.3390/electronics10202520
60. Picone M, Mamei M, Zambonelli F. WLDT: a general purpose library to build IoT digital twins. *SoftwareX*. 2021;13:100661. doi:10.1016/j.softx.2021.100661
61. Bruno S, Del Serrone G, Di Mascio P, Loprencipe G, Ricci E, Moretti L. Technical proposal for monitoring thermal and mechanical stresses of a runway pavement. *Sensors*. 2021;21(20):1-15. doi:10.3390/s21206797
62. Kannoth S, Hermann J, Damm M, et al. Enabling SMEs to industry 4.0 using the BaSyx middleware: a case study. In: Biffl S, Navarro E, Löwe W, Sirjani M, Mirandola R, Weyns D, eds. *Software Architecture*. Springer International Publishing; 2021:277-294.
63. Shah K, Prabhakar TV, Sarweshkumar CR, Abhishek SV, Vasanth Kumar T. Construction of a digital twin framework using free and open-source software programs. *IEEE Internet Comput*. 2022;26(5):50-59. doi:10.1109/MIC.2021.3051798
64. Kamath V, Morgan J, Ali MI. Industrial IoT and digital twins for a smart factory: an open source toolkit for application design and benchmarking. Paper presented at: GIoTS 2020-Global Internet of Things Summit, Proceedings. 2020; 0–5. doi:10.1109/GIOTS49054.2020.9119497
65. Sta Ana RR, Escoto JE, Fargas D, Panlilio K, Jerez M, Sarmiento CJ. Development of a digital twin for the monitoring of smart cities using open-source software. *Int Arch Photogram Remote Sens Spatial Inf Sci ISPRS Arch*. 2021;46:281-288. doi:10.5194/isprs-Archives-XLVI-4-W6-2021-281-2021
66. Forssen J, Hostmad P, Wastberg BS, et al. An urban planning tool demonstrator with auralisation and visualisation of the sound environment. Paper presented at: E-Forum Acusticum 2020, Lyon, France, 7-11 December 2020. 2020:869–871. doi:10.48465/fa.2020.0374
67. Lehner D, Sint S, Vierhauser M, Narzt W, Wimmer M. AML4DT: a model-driven framework for developing and maintaining digital twins with AutomationML. Paper presented at: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). 2021. doi:10.1109/ETFA45728.2021.9613376
68. Shah K, Prabhakar TV, Sarweshkumar CR, Abhishek SV, Vasanth Kumar T. Construction of a digital twin framework using free and open-source software programs. *IEEE Internet Comput*. 2022;26(5):50-59. doi:10.1109/MIC.2021.3051798
69. Talasila P, Gomes C, Mikkelsen PH, Arboleda SG, Kamburjan E, Larsen PG. Digital twin as a service (DTaaS): a platform for digital twin developers and users. Paper presented at: Digital Twin 2023: the 2023 IEEE International Conference on Digital Twin, IEEE: Portsmouth, UK. 2023.

How to cite this article: Gil S, Mikkelsen PH, Gomes C, Larsen PG. Survey on open-source digital twin frameworks–A case study approach. *Softw Pract Exper*. 2024;1-32. doi: 10.1002/spe.3305