Towards a Classification Framework for the Digital Twin Tools: A Taxonomy

Mert Ozkaya[1] and Alper Turunc[2]
[1]*Department of Computer Engineering, Yeditepe University, Istanbul, Turkey*
[2]*DFDS, Istanbul, Turkey*

Keywords: Digital Twin, Classification Framework, Taxonomy, Tools, Technologies.

Abstract: Digital twins (DTs) have gained an ever-increasing popularity in many industries for the real-time monitoring of physical systems and performing useful operations such as predictive maintenance and early fault detection. Many commercial and open-source DT tools are available on the market, which offer services or development environment for developing and using DTs. However, our research shows that practitioners still use conventional programming technologies for developing DTs in-house rather than using the DT-specific tools. We believe that one reason here is to do with the lack of any taxonomy for the determining the DT tools that are of use for the practitioners. In this paper, we propose our attempt for a classification framework that can be used for analysing and comparing different DT tools. Having reviewed the literature and practitioners' needs, we addressed twelve main features for DT tools, which are divided into sub-features. These are *(i)* domain, *(ii)* deployment, *(iii)* type, *(iv)* maturity, *(v)* knowledge management, *(vi)* system integration, *(vii)* quality assurance, *(viii)* reusability, *(ix)* extensibility, *(x)* abstraction, *(xi)* enabling technology, and *(xii)* development platform. We believe that our classification framework will be useful for different stakeholders: *(i)* practitioners who wish to develop and use DTs, *(ii)* tool vendors who can determine strengths and weaknesses of their tools, *(iii)* researchers who address the tool weaknesses.

1 INTRODUCTION

Digital twin (DT) is nowadays considered as one of the hottest topics of computer science and used in several industries for the real-time monitoring of any physical systems remotely and optimising the product development lifecycle (Rasheed et al., 2020; Jones et al., 2020; Hu et al., 2021). These include such industries as aerospace, automotive, manufacturing, mining, energy, construction, healthcare, agriculture, smart cities, and oil and gas (Singh et al., 2022). DTs can offer several advantages for those industries which include the real-time simulation for the optimised solutions, personalisation of products, enhanced communication among stakeholders, reduced cost of prototyping, early detection of errors and predictive maintenance, and remote accessibility (Singh et al., 2021; Rasheed et al., 2020).

As DTs have gained an ever-increasing popularity in industries, several different software development companies offer DT tools whose goal is essentially to help users solving their business problems effectively and productively. DT development companies can provide varying solutions for the DT development. Some companies release DT tools that actually offer a development platform with APIs, frameworks, and libraries which can be used by practitioners in developing domain-specific DTs for the specific needs. Also, some companies offer DT tools as a service and such DT tools provide several DT operations that can be used over cloud licenses (or on-premise) for particular domains and industries. Besides, some companies offer DT tools with an open-source support for the development of specific DTs using re-usable and customisable generic code under certain constraints. Table 1 shows some of the well-known DT tools[1].

To understand the use of DT tools in practice, we focussed on the results of our recent industry survey that has been conducted online between June-August 2024 and addresses practitioners' perspectives on the DT techniques and technologies. Our survey received 131 responses from diverse industries and profiles[2].

[1]The full list of DT tools is accessible here: https://zenodo.org/records/13956879

[2]The survey questions and the raw response data are accessible here: https://zenodo.org/records/13628837

Ozkaya, M. and Turunc, A.
Towards a Classification Framework for the Digital Twin Tools: A Taxonomy.
DOI: 10.5220/0013239900003896

In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2025)*, pages 263-272
ISBN: 978-989-758-729-0; ISSN: 2184-4348

Table 1: The list of DT tools.

Digital Twin Tool	Website
AnyLogic	www.anylogic.com
Tecnomatix	www.plm.sw.siemens.com/en-US/tecnomatix
aPriori	www.apriori.com
XMPro App Designer	www.xmpro.com
Autodesk Tandem	www.intandem.autodesk.com
Azure Digital Twins	www.azure.microsoft.com/en-us/products/digital-twins
Unlearn.AI	www.unlearn.ai
Ansys Twin Builder	www.ansys.com
Bentley Systems	www.bentley.com/software/infrastructure-digital-twins
Rockwell Automation	www.rockwellautomation.com
General Electric	www.ge.com
Mimic Simulation	www.mimicsimulation.com/simulation-software
Veerum	www.veerum.com
CADMATIC eShare	www.cadmatic.com/en/products/cadmatic-eshare
Archilogic	www.archilogic.com
Golem	www.golem.at
Value Chain Resilience GmbH	www.vcresilience.com
Agrilogic	www.agrilogicconsulting.com
Twinzo	www.twinzo.eu
Eclipse Ditto	www.eclipse.dev/ditto
Eclipse BaSyx	www.eclipse.dev/basyx
iTwin.js	www.itwinjs.org
SewerAI	www.sewerai.com
Bosch IoT Suite	www.bosch-iot-suite.com
Oracle IoT Production Monitoring Cloud	www.docs.oracle.com/en/cloud/saas/iot-production-cloud
OpenSpace Group Ltd	www.open-space.io
Process Genius Oy	www.processgenius.fi
Blackshark	www.blackshark.ai

So, we learned that despite many DT tools being available on the market, practitioners still use the conventional programming languages (e.g., Python and Java) to develop their DTs in-house from scratch. Practitioners also face with challenges on such concerns as the system integration, security, performance, data quality, organisation, environment, development platform, and abstraction. Despite many commercial and open-source DT tools, no any classification framework is available for analysing and comparing the DT tools. As discussed in Section 3, while the literature includes several analytical works about DTs, none of them provide any lists of the existing DT tools, any list of criteria for comparing the DT tools, or any work that provide the comparison of the DT tools for some specific requirements.

In this paper, we aim to propose a classification framework which describes the important features that need to be considered for developing and using DT systems. We believe that our classification framework will be useful for many stakeholders. Practitioners can use the framework for analysing and comparing the DT tools with regard to their needs. Researchers can identify the key features of any DT tools and conduct relevant researches. Tool vendors can identify the weakness and strengths of their DT tools.

Our classification framework is depicted in Figure 1, which consists of twelve different features. These are *(i)* domain, *(ii)* knowledge management, *(iii)* system integration, *(iv)* quality assurance, *(v)* reusability, *(vi)* development platform, *(vii)* deployment, *(viii)* type, *(ix)* extensibility, *(x)* abstraction, *(xi)* enabling technology and *(xii)* maturity, and each feature is further divided into a set of sub-features. In the rest of the paper, we initially introduce our methodology for identifying the DT tool features and discussed similar works in the literature. Then, we discuss each tool feature separately in terms of their sub-features.

In the rest of the paper, we initially discuss our methodology for the DT domain analysis. Then, we discuss the similar studies in the literature. Next, we introduce our classification framework in terms of its features and sub-features. Lastly, we discuss a case study based on the analysis of a well-known DT tool in terms of our classification framework.

2 DOMAIN ANALYSIS

To come up with a set of features for classifying DT tools, we performed a comprehensive domain analysis. To this end, we firstly used *google scholar* and searched for the papers that discuss *(i)* the definitions of DTs and their characteristics (Barricelli et al., 2019; Singh et al., 2021; Mihai et al., 2022; Rasheed et al., 2020; Jones et al., 2020; Tao et al., 2019; Hu et al., 2021; Fuller et al., 2020), *(ii)* the enabling DT technologies and any challenges in applying those technologies (Michael et al., 2022; da Rocha et al., 2022; Qi et al., 2021; Lv and Xie, 2022; TAO et al., 2024; Newrzella et al., 2022; Shao and Helu, 2020), *(iii)* the applications of DTs in different industries (Ghaboura et al., 2023; Lin et al., 2022; Perno et al., 2022; Biesinger et al., 2019; Purcell and Neubauer, 2023; Ali et al., 2023), and *(iv)* the analysis of DT applications and technologies from different perspectives (Gil et al., 2024; Liu et al., 2022; Alcaraz and Lopez, 2022; Minerva et al., 2020; Wu et al., 2021; Lo et al., 2021). Having identified the relevant papers, we went through each paper and identified any attributes that are discussed in the paper and can be considered as the DT tool features. A DT tool feature here can either be a commonly considered functionality by the DT users, a crucial quality concern for DT tools, a technique or technology needed for developing DT tools, or some other requirement for the effective use of the DT tools. All identified attributes that represent the candidates for being the DT tool features have been stored on an Excel file. We created a separate sheet for each feature in the Excel file. We analysed each feature for determining the sub-features that represent the main concern(s) about that feature. To this end, we searched the literature

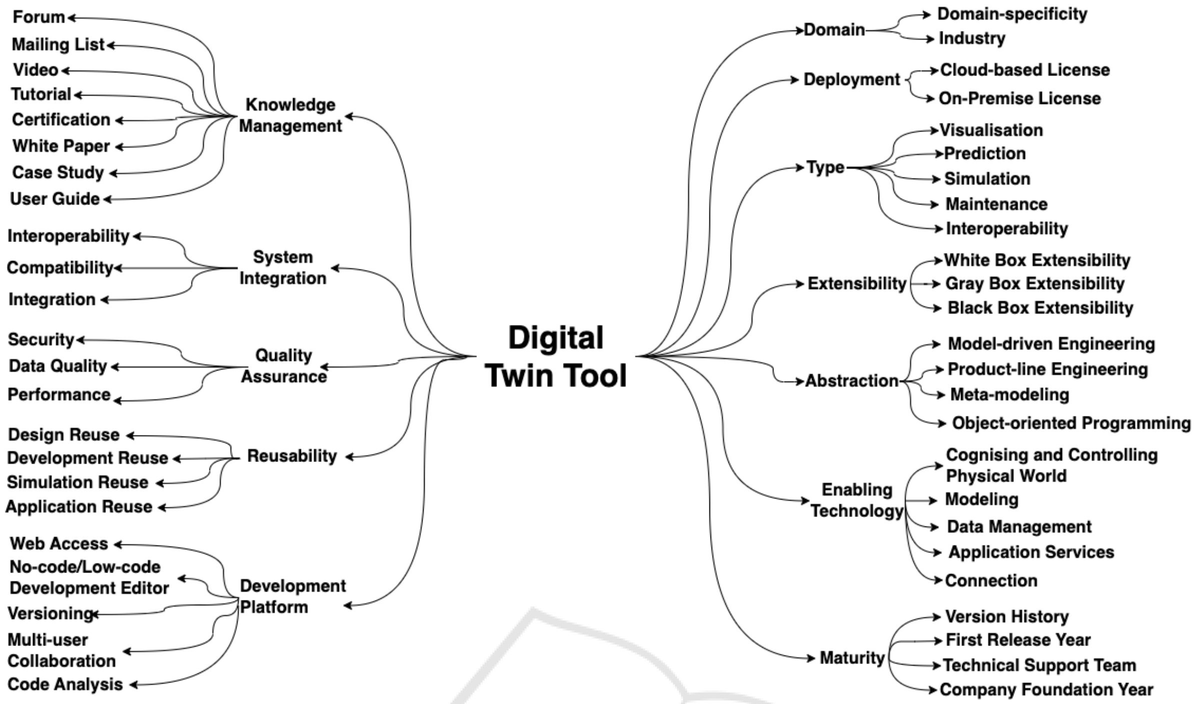

Figure 1: The taxonomy for DT tools.

and identified any well-cited papers that discuss the feature-related techniques, approaches, and concerns. By doing so, we were able to determine a set of sub-features for each feature and stored those sub-features in the relevant sheets of the Excel file.

After identifying the DT tool features by means of the analysis of the relevant papers, we further used the results of our recent industry survey[2] for understanding practitioners' perspectives on the DT technologies. The results of our survey lead to many interesting findings about the practitioners' *(i)* adoption of DT technologies, *(ii)* motivations for using and developing DTs, *(iii)* development experiences on DTs, and *(iv)* their challenges from diverse perspectives. The survey results revealed many interesting features that practitioners are concerned with but was not so explicit in our literature analysis (e.g., the reusability and abstraction). So, we further extended the Excel file with the newly identified features for the DT tools.

After documenting the features, we performed a pilot study with one academic and two practitioners who have considerable experiences on DTs. The academic and two practitioners have been provided with the Excel file that includes the extracted features from the papers and requested to provide a review based on their knowledge and experience on DT systems. By doing so, we were able to remove any redundancies and inconsistencies, and add any missing features. So, we ended up with a taxonomy of the DT tool features that is depicted in Figure 1. Our taxonomy consists of twelve different features each of which is decomposed into a cohesive set of sub-features.

3 RELATED WORK

The literature includes several papers that discuss DTs from different perspectives, which includes the *(i)* characterisations of DTs (e.g., (Barricelli et al., 2019; VanDerHorn and Mahadevan, 2021; Jones et al., 2020; Liu et al., 2021)), *(ii)* applications of DTs in different domains (e.g., (Cimino et al., 2019) for manufacturing, (Sun et al., 2023) for healthcare, (Verdouw et al., 2021) for farming, and (Opoku et al., 2021) for construction), *(iii)* challenges that can be faced with when using and developing DTs (Verdouw et al., 2021), *(iv)* the analysis of DTs from particular aspects (e.g., simulation (Boschert and Rosen, 2016), fault monitoring (Bofill et al., 2023), and maintenance (Errandonea et al., 2020)), and *(v)* case studies for diverse problems (e.g., (Mendi, 2022; Lu et al., 2020; Peng et al., 2020)).

The classification frameworks in the literature focus on, e.g., software evolution (Mens et al., 2003), architecture description languages (Medvidovic and Taylor, 2000), problem solving techniques in particular domains (e.g., the taxonomy for scheduling algorithms for wireless mesh networks (Gabale et al., 2013)), component-based software engineering (Crnkovic et al., 2011), software connectors (Mehta et al., 2000) and software change impact (Lehnert, 2011). The literature also includes classification framework studies about the software tools, such as the classifications of software testing tools (Graham, 1991), static code analysis tools (Novak et al., 2010), software fault monitoring tools (Delgado et al., 2004), model transformation tools (Kahani et al., 2019), and manufacturing scheduling tools (Dios and Framinan, 2016). However, no any effort has been made for classifying the DT tools. While some generic classification frameworks have been proposed for classifying software tools in general such as (Firth et al., 2018), those studies may not be so helpful due to lacking support for the DT context.

4 DT TOOL FEATURES

In this section, we discuss each DT tool feature that is depicted in Figure 1 in terms of its sub-features.

4.1 Domain

As indicated in our survey results[2], DTs are used in many industries for solving diverse problems. Therefore, we consider the domain as an important factor for classifying the DT tools. The domain feature here is concerned with the problem domain for which any DT tool is developed and used, and consists of two sub-features - i.e., industry and domain-specificity. While the industry sub-feature represents the industry(ies) that a DT tool can support (e.g., manufacturing, defense, and automotive), the domain-specificity sub-feature represents the specific problem(s) that is(are) addressed in a particular industry. For instance, (Fahim et al., 2022) shows the use of predictive modeling in wind turbines, (Peker and Akdur, 2019) shows the real-time simulation for signal processing in the defense industry, and (Hodavand et al., 2023) shows the fault detection and diagnosis for the smart buildings in the construction industry.

4.2 Knowledge Management

Knowledge management for any system is to do with managing the information and resources, which includes collecting, analysing information and transforming them into knowledge that can be stored, shared, and evolved (Despres and Chauvel, 1999). Knowledge management for a DT tool consists of several sub-features, which are *(i)* forum, *(ii)* mailing list, *(iii)* video, *(iv)* tutorial, *(v)* certification, *(vi)* white paper, *(vii)* case study, and *(viii)* user guide. Forum represents an online discussion platform for the DT tool users to ask questions and get useful responses from tool experts. Mailing list represents a collection of e-mails that belong to the subscribed users for a DT tool, and the DT tool vendor can send broadcast e-mails for notifying the users about tool-related news (e.g., version updates). Video represents any visual presentations that introduce either the features of any DT tool, usage patterns, user guide, or case-studies which can be useful for the DT tool users. Tutorial represents any visual presentation or textual document that can teach users how to use the DT tool in a series of stages. Certification represents the confirmation that can be granted by the tool vendors or any contracted organisation for any DT tool user so as to prove their abilities on the DT tool. White paper represents any report that discusses an innovation, issue or technique about a DT tool from a scientific point of view. Case study represents any document that discusses the evaluation of a DT tool on a particular case or real-problem. User guide represents a document that describes how to use a DT tool and its features.

4.3 Deployment

Deployment is concerned with the making a DT tool available for the users so that users can use the DT tool and perform their goals. Here, we consider two deployment options (i.e., the sub-features), which are cloud-based license and on-premise license. Cloud-based license is based on the subscription method where the users of a DT tool choose any package and pay its price on a monthly or yearly basis so as to rent the DT tool which is operated on cloud. On-premise license prompts a DT tool software to be installed on the local servers of customers through which users can access the DT tool.

4.4 Type

Type for a DT tool is concerned with the functionalities that can be performed with the DT tool. Therefore, types can be distinguished with their operations on the DT model that represent the abstraction(s) over the physical system. We consider five types of DTs, which are visualisation DTs, prediction DTs, simulation DTs, maintenance DTs, and DTs for interoperability (Liu et al., 2021; Piroumian, 2021). Note that any DT tool may be of multiple types at the same time. A visualisation DT focuses on visualising any model of the physical system in various notations (e.g., physical, mathematical, logical) which can be updated dynamically with the data collected from the physical system. A prediction DT uses AI techniques (e.g., machine learning, deep learning, decision trees) and makes useful predictions on the models that represent abstractions over the physical system. A simulation DT focuses on executing the models of the physical system and checking if certain scenarios are satisfied or not. A maintenance DT focuses on analysing the models of the physical systems for some quality properties so as to predict any future failures and thus planing and scheduling the system maintenance. A DT for interoperability focuses on managing the communication and coordination among a set of physical devices (e.g., sensors and actuators) and the physical system in a way that certain standards (e.g., security protocols, simulation standards, communication protocols) are followed.

4.5 System Integration

System integration is concerned with the support for developing a DT by means of integrating several heterogenous systems in a way that those systems can interact and exchange data without any problems (Perno et al., 2022). We consider here the system integration in terms of the support for *(i)* integration, *(ii)* interoperability, and *(iii)* compatibility. Integration is concerned with the support for *(i)* integrating with external systems that can provide useful services and *(ii)* developing DTs in terms of the composition of existing systems. Interoperability is concerned with the support for enabling the heterogenous systems (e.g., sensors and actuators) to exchange data with a DT. Compatibility is concerned with the support for integrating heterogenous systems that run on different platforms and adopt different protocols of interactions.

4.6 Quality Assurance

Quality assurance is concerned with the support for checking the quality of a DT system that is developed or used via a DT tool. Here, we consider 3 main quality properties that are important for the development of high-quality DTs (Perno et al., 2022). These are *(i)* security, *(ii)* data quality, and *(iii)* performance. Security is concerned with the support for minimising any security threats for the physical and software compositions of DTs that can cause issues in terms of the availability, integrity and confidentiality of system resources (i.e., data and services) (Alcaraz and Lopez, 2022). Data quality is concerned with the support for accessing the right data from the right data sources in a way that meets the system quality requirements. Performance is concerned with the support for the efficient use of software and hardware resources so that system data can be exchanged between the physical and software systems with minimum latency.

4.7 Reusability

In our survey[2], reusability (Prieto-Diaz, 1993) has been observed as one of the top crucial challenges for the practitioners who develop DTs. Reusability is concerned with reducing the cost of designing, developing and operating DT systems and maximising the quality by means of reusing the tested and proven solutions and any existing applications. Here, we consider 4 sub-features, which are design reuse, development reuse, simulation reuse, and application reuse. The design reuse sub-feature is concerned with the support for the reusable design of DT systems and thus making quality design decisions with the least cost possible by means of, e.g., design patterns, architecture styles, design languages, etc. The development reuse sub-feature is concerned with the support for the re-usable implementation of DT systems by means of, e.g., programming frameworks, APIs, etc. The simulation reuse sub-feature is concerned with the support for the re-usable specifications of simulation models that can be executed for the real-time simulation by means of, e.g., model libraries, simulation modeling languages, etc. The application reuse sub-feature is concerned with the support for re-using external applications (e.g., COTS) by means of software connectors (or adaptors) that handle the adaptation of external systems to the DT systems.

4.8 Development Platform

Some DT tools may provide a development platform which provides interesting facilities for users and improves the DT development processes by maximising users' productiviness and effectiveness. As indicated in our survey[2], practitioners use different DT development platforms for diverse problem domains and indeed face with challenges. Given our analysis of our survey results, we consider here six different sub-features for the development platforms, which are *(i)* web access, *(ii)* no-code (or low-code) development editor, *(iii)* versioning, *(iv)* multi-user collaboration, *(v)* exporting, and *(vi)* code analysis. Web access enables users to use the DT development platform over internet without having to download and install any tools. No-code (or low-code) development editors provide graphical editors and enable users to specify high-level visual models for the DT systems and generate fully (or partially) executable code. Versioning enables managing the versions of the DT development artifacts (e.g., code) and performing such operations as comparing, merging, and accessing previous versions. Multi-user collaboration enables multiple users to work together on the development of the same DT system and perform the design and coding activities at the same time in a synchronised way. Automated code analysis enables the users to check their DT development code automatically and detect any bugs or quality issues.

4.9 Extensibility

Extensibility is concerned with the ability of extending the software tools (e.g., adding new functionalities or quality attributes) without breaking their core architectures and violating the principal design decisions (Kazman et al., 2022). DT tools are expected to be extensible since the desired requirements can always change and the DT tools need to be adapted. Otherwise, any changes lead to unmanageable costs and cause the DT tools to be discontinued being used. As revealed in our survey results[2], many practitioners prefer to use conventional programming technologies (e.g., Python and Java) in developing DTs in house despite many DT development platforms that are available on the market.

We consider the DT tool extensibility in terms of *(i)* white-box extensibility, *(ii)* glass-box extensibility, and *(iii)* black-box extensibility (Zenger, 2004). White-box extensibility enables the DT source-code to be changed by the users directly such as the open-source software projects. Glass-box extensibility enables the source-code to be accessed as read-only and any extensions are performed on a separate copy of the source-code from the original version. The inheritance and dynamic binding mechanisms of object-oriented programming are the common examples of glass-box extensibility. Black-box extensibility enables users to extend a DT by means of interfaces that abstract users from reaching the DT source-code and prompt any extensions to be performed under certain rules and constraints using such techniques as APIs, plug-in development tools, model-driven engineering tools.

4.10 Abstraction

Abstraction promotes the management of complexity and solving complex problems by *(i)* suppressing irrelevant details and focusing on important aspects and *(ii)* generalising the problems by separating the common parts from the varying parts (Kramer, 2007). As DTs are considered as complex systems, abstraction is a key skill and needs to be adopted throughout he DT development processes. We consider here the following abstraction techniques that can be supported by the DT tools, which are *(i)* model-driven engineering, *(ii)* software product-line engineering, *(iii)* modeling language, *(iv)* meta-modeling, *(v)* object-oriented programming. Model-driven engineering (MDE) (Schmidt, 2006) promotes the specifications of high-level, abstract models of any DTs from different viewpoints, the automated transformation of abstract DT models into useful artifacts such as simulator code, programming code, or any useful documentation . Software product-line engineering (SPLE) (Metzger and Pohl, 2014) promotes the development of similar products (e.g., similar DTs for a particular domain) using the minimum time and budget by separating commonalities of similar products from their variations. SPLE can be applied in different stages of the DT development including the analysis, design and implementation, and maximises the reusable development of DTs that derive from the same product family which possess common and varied features. Meta-modeling (Ozkaya and Akdur, 2021). promotes the definition of DT modeling languages in terms of abstract and concrete notation sets through which DT models can be specified (e.g., (Ozkaya, 2024)). Lastly, object-oriented programming (OOP) promotes the notion of classes that is the unit of abstraction which consists of a common set of methods and attributes and can be specialised using the inheritance mechanism (Stroustrup, 1987).

4.11 Enabling Technology

As aforementioned, DTs are complex systems that require the interactions of physical and software systems and perform complex operations including the real-time data analysis, modeling and simulation, visualisation, intelligent decision making, and AI-supported diagnosis and prediction. As revealed with our survey[2], practitioners prefer several different technologies for developing DTs. We consider here the enabling technologies for DTs in terms of five main features that represent the distinct categories of the DT technologies which have been proposed by Qi et al. (Qi et al., 2021). These are *(i)* cognising and controlling the physical world, *(ii)* modeling, *(iii)* data management, *(iv)* application services, and *(v)* connection. The feature for cognising and controlling the physical world is concerned with any technologies that enable gathering data from different physical devices (e.g., sensors and camera), transforming data into knowledge, and controlling physical systems using the transformed knowledge. Modeling is concerned with any technologies that enable the modeling of a product that is monitored by a DT in terms of different viewpoints including the 3D geometric modeling of the product shapes, structural modeling (e.g., the physical and logical structures), behaviour and interaction modeling, and rule modeling. Data management is concerned with any technologies that enable the data management activities which are the data collection, data transmission, data storage, data processing, data fusion, and data visualisation. Application services are concerned with the technologies that enable monitoring, simulation, diagnosis and prediction. Connection is concerned with any technologies that enable the communication, coordination and complex interactions for the physical and virtual systems that compose DTs such as the technologies for the secure data communications, human-computer interaction, and standard communication interfaces.

4.12 Maturity

DT tools can vary depending on their level of maturity which is concerned with the tool vendors' support for the tool evolution so as to promote the continuous improvement of their tools with regard to changing technologies, user demands, rules and regulations (Becker et al., 2009). We consider the DT tool maturity in terms of five key sub-features, which are the *(i)* version history *(ii)* first release year, *(iii)* live support, *(iv)* technical support, and *(v)* company foundation year. Version history is concerned with the number of tool versions that have been released so far and any details about the versions such as the version release dates, the changes that are included in each version, and any errors corrected. The first release year is concerned with the date on which the first, initial version of the DT tool has been released. The technical support team is concerned with such facilities as live chat, e-mail service, and help desk through which the tool users can be supported in resolving any technical issues that they face with while using the DT tools. Lastly, company foundation year represents how long the company has been working on the relevant ares and their level of experience.

5 CASE STUDY

In this section, we demonstrate the use of our classification framework that is discussed in Section 4 for AutoDesk Tandem[3], which is one of the most well-known development platforms for developing DTs. To collect data about each feature and their sub-features for our classification framework, we used AutoDesk Tandem's free version and searched over their comprehensive web-site.

Concerning the domain feature, AutoDesk Tandem is specific for the buildings domain and promotes for smarter buildings. AutoDesk Tandem is used by the architecture, engineering, construction, and operations industry.

Concerning the knowledge management, AutoDesk Tandem supports community forums for the users to share their ideas with each other. AutoDesk Tandem provides several e-learning resources such as online courses and tutorials, which can be categorised depending on the DT software, industry and users' job positions. Moreover, AutoDesk Tandem provides certifications for the students and practitioners to prove their skills on the use of AutoDesk Tandem for the DT development. AutoDesk Tandem provides a comprehensive blog page through which several white papers and case studies have been published. Lastly, AutoDesk provides several user guide documents for different aspects of the DT development platform.

Concerning the deployment, AutoDesk Tandem can only be used over a cloud platform and it is not possible to install the DT development software on a local machine.

Concerning the type, AutoDesk Tandem provides a visualisation support for the 3D graphics modeling of the buildings and simulation support for analysing the building performance.

Concerning the system integration, AutoDesk Tandem provides several facilities for managing the integration, interoperability, and compatibility issues. These facilities include streams, services, connectors, converters, data mappers, filters, channels and gateways, which can be used over a no-code development platform without having to write any code.

Concerning the quality assurance, AutoDesk Tandem provides an authentication API for the DT security. AutoDesk Tandem provides a data dashboard for managing the data quality such as checking the data correctness and completeness for different elements of the buildings. However, we could not reach any resources for improving the performance of DT systems.

Concerning the reusability, AutoDesk Tandem provides development reusability via the API libraries and application reusability via the integration facilities (e.g., plugins, connectors, etc.).

Concerning the development platform, AutoDesk Tandem provides a web application for developing and using DTs. No-code/low-code development platforms are provided for the system integration facilities. The past data used by DTs can be stored and

[3]AutoDesk Tandem: https://intandem.autodesk.com

accessed at any time, which can further be used for reverting any changes on the DT systems. AutoDesk Tandem enables multi-users to collaborate while developing and using DT systems. No any code analysis support is provided.

Concerning the extensibility, AutoDesk Tandem supports the black-box extensibility via their APIs and plugin tools.

Concerning the abstraction, AutoDesk Tandem supports model-driven engineering only. That is, 3D building models can be specified, executed for simulation purposes and managed in terms of different viewpoints.

Concerning the enabling technologies, AutoDesk Tandem supports cognising and controlling the physical world thanks to its APIs and system integration facilities. AutoDesk Tandem also supports the 3D modeling of buildings, management of data via its data dashboard features, application services (i.e., monitoring, simulation and diagnosis), and the connection technologies.

Concerning the maturity, AutoDesk Tandem's first release has been announced on July 12, 2021 and several releases have been published so far. AutoDesk Tandem does not provide any live support. Technical support is provided via the customer representatives upon request. Lastly, the the AutoDesk company has been founded in 1982.

6 CONCLUSION

Many DT tools are available on the market, which can be used by the practitioners to obtain DT solutions for their business problems. DT tools can either offer a DT development platform (e.g., APIs, frameworks, and libraries), any DT products that can be rented for solving specific problems, or an open-source environment for extending an existing DT tool. Despite many DT tools existing, practitioners develop their DTs inhouse using the conventional programming technologies. We believe that an important reason here is that no any classification framework for comparing the DT tools have been proposed so far. Therefore, it is not easy for the practitioners to determine the DT tools available on the market, compare the tools among each other, and choose the DT tool(s) that best fit their requirements.

In this paper, we aimed to propose a classification framework for the DT tools. To this end, we firstly analysed the DT literature and further conducted a practitioner survey among 131 practitioners from diverse industries. Performing a pilot study on our analysis results on the DT domain, we proposed a taxonomy which consists of twelve different features. These are knowledge management, domain, deployment, system integration, type, extensibility, quality assurance, abstraction, reusability, enabling technology, development platform, and maturity. Each feature is further decomposed into a set of sub-features. Moreover, we demonstrated our classification framework using one of the most well-known DT development platforms called AutoDesk Tandem.

We are currently working on validating our classification framework with some well-known DT tools besides AutoDesk Tandem. We will further conduct a survey and interviews among the practitioners who work for the defense industry and frequently use and develop DTs so as to validate our classification framework. Besides, we will analyse 80 different DT tools with regard to the features of our classification framework. By doing so, we aim to provide a really useful guide that can be used by the practitioners for identifying the existing DT tools, analysing the features of interest, and choosing the DT tool(s) that best meet their needs.

REFERENCES

Alcaraz, C. and Lopez, J. (2022). Digital twin: A comprehensive survey of security threats. *IEEE Communications Surveys & Tutorials*, 24(3):1475–1503.

Ali, W. A., Fanti, M. P., Roccotelli, M., and Ranieri, L. (2023). A review of digital twin technology for electric and autonomous vehicles. *Applied Sciences*, 13(10).

Barricelli, B. R., Casiraghi, E., and Fogli, D. (2019). A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671.

Becker, J., Knackstedt, R., and Pöppelbuß, J. (2009). Developing maturity models for it management. *Business & Information Systems Engineering*, 1(3):213–222.

Biesinger, F., Kraß, B., and Weyrich, M. (2019). A survey on the necessity for a digital twin of production in the automotive industry. In *2019 23rd International Conference on Mechatronics Technology (ICMT)*, pages 1–8.

Bofill, J., Abisado, M., Villaverde, J., and Sampedro, G. A. (2023). Exploring digital twin-based fault monitoring: Challenges and opportunities. *Sensors*, 23(16).

Boschert, S. and Rosen, R. (2016). *Digital Twin—The Simulation Aspect*, pages 59–74. Springer International Publishing, Cham.

Cimino, C., Negri, E., and Fumagalli, L. (2019). Review of digital twin applications in manufacturing. *Computers in Industry*, 113:103130.

Crnkovic, I., Sentilles, S., Vulgarakis, A., and Chaudron, M. R. (2011). A classification framework for software component models. *IEEE Transactions on Software Engineering*, 37(5):593–615.

da Rocha, H., Pereira, J., Abrishambaf, R., and Espirito Santo, A. (2022). An interoperable digital twin with the ieee 1451 standards. *Sensors*, 22(19).

Delgado, N., Gates, A., and Roach, S. (2004). A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE Transactions on Software Engineering*, 30(12):859–872.

Despres, C. and Chauvel, D. (1999). Knowledge management(s). *Journal of Knowledge Management*, 3(2):110–123.

Dios, M. and Framinan, J. M. (2016). A review and classification of computer-based manufacturing scheduling tools. *Computers & Industrial Engineering*, 99:229–249.

Errandonea, I., Beltrán, S., and Arrizabalaga, S. (2020). Digital twin for maintenance: A literature review. *Computers in Industry*, 123:103316.

Fahim, M., Sharma, V., Cao, T. V., Canberk, B., and Duong, T. Q. (2022). Machine learning-based digital twin for predictive modeling in wind turbines. *IEEE Access*, 10:14184–14194.

Firth, R., Mosley, V., Pethia, R. D., Gold, L. R., and Wood, W. (2018). A Guide to the Classification and Assessment of Software Engineering Tools.

Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971.

Gabale, V., Raman, B., Dutta, P., and Kalyanraman, S. (2013). A classification framework for scheduling algorithms in wireless mesh networks. *IEEE Communications Surveys & Tutorials*, 15(1):199–222.

Ghaboura, S., Ferdousi, R., Laamarti, F., Yang, C., and Saddik, A. E. (2023). Digital twin for railway: A comprehensive survey. *IEEE Access*, 11:120237–120257.

Gil, S., Mikkelsen, P. H., Gomes, C., and Larsen, P. G. (2024). Survey on open-source digital twin frameworks–a case study approach. *Software: Practice and Experience*, 54(6):929–960.

Graham, D. R. (1991). Software testing tools: A new classification scheme. *Software Testing, Verification and Reliability*, 1(3):17–34.

Hodavand, F., Ramaji, I. J., and Sadeghi, N. (2023). Digital twin for fault detection and diagnosis of building operations: A systematic review. *Buildings*, 13(6).

Hu, W., Zhang, T., Deng, X., Liu, Z., and Tan, J. (2021). Digital twin: a state-of-the-art review of its enabling technologies, applications and challenges. *Journal of Intelligent Manufacturing and Special Equipment*, 2(1):1–34.

Jones, D., Snider, C., Nassehi, A., Yon, J., and Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52.

Kahani, N., Bagherzadeh, M., Cordy, J. R., Dingel, J., and Varró, D. (2019). Survey and classification of model transformation tools. *Software & Systems Modeling*, 18(4):2361–2397.

Kazman, R., Echeverría, S., and Ivers, J. (2022). Extensibility. Technical Report CMU/SEI-2022-TR-002. Accessed: 2024-Oct-30.

Kramer, J. (2007). Is abstraction the key to computing? *Commun. ACM*, 50(4):36–42.

Lehnert, S. (2011). A taxonomy for software change impact analysis. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, page 41–50, New York, NY, USA. Association for Computing Machinery.

Lin, Y., Chen, L., Ali, A., Nugent, C., Ian, C., Li, R., Gao, D., Wang, H., Wang, Y., and Ning, H. (2022). Human digital twin: A survey.

Liu, M., Fang, S., Dong, H., and Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361. Digital Twin towards Smart Manufacturing and Industry 4.0.

Liu, Y. K., Ong, S. K., and Nee, A. Y. C. (2022). State-of-the-art survey on digital twin implementations. *Advances in Manufacturing*, 10(1):1–23.

Lo, C., Chen, C., and Zhong, R. Y. (2021). A review of digital twin in product design and development. *Advanced Engineering Informatics*, 48:101297.

Lu, Q., Parlikad, A. K., Woodall, P., Ranasinghe, G. D., Xie, X., Liang, Z., Konstantinou, E., Heaton, J., and Schooling, J. (2020). Developing a digital twin at building and city levels: Case study of west cambridge campus. *Journal of Management in Engineering*, 36(3):05020004.

Lv, Z. and Xie, S. (2022). Artificial intelligence in the digital twins: State of the art, challenges, and future research topics [version 2; peer review: 2 approved]. *Digital Twin*, 1(12).

Medvidovic, N. and Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Trans. Software Eng.*, 26(1):70–93.

Mehta, N. R., Medvidovic, N., and Phadke, S. (2000). Towards a taxonomy of software connectors. In *Proceedings of the 22nd International Conference on Software Engineering*, ICSE '00, page 178–187, New York, NY, USA. Association for Computing Machinery.

Mendi, A. F. (2022). A digital twin case study on automotive production line. *Sensors*, 22(18).

Mens, T., Buckley, J., Zenger, M., and Rashid, A. (2003). Towards a taxonomy of software evolution.

Metzger, A. and Pohl, K. (2014). Software product line engineering and variability management: achievements and challenges. In Herbsleb, J. D. and Dwyer, M. B., editors, *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014*, pages 70–84. ACM.

Michael, J., Pfeiffer, J., Rumpe, B., and Wortmann, A. (2022). Integration challenges for digital twin systems-of-systems. In *Proceedings of the 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosys-*

tems, SESoS '22, page 9–12, New York, NY, USA. Association for Computing Machinery.

Mihai, S., Yaqoob, M., Hung, D. V., Davis, W., Towakel, P., Raza, M., Karamanoglu, M., Barn, B., Shetve, D., Prasad, R. V., Venkataraman, H., Trestian, R., and Nguyen, H. X. (2022). Digital twins: A survey on enabling technologies, challenges, trends and future prospects. *IEEE Communications Surveys & Tutorials*, 24(4):2255–2291.

Minerva, R., Lee, G. M., and Crespi, N. (2020). Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, 108(10):1785–1824.

Newrzella, S. R., Franklin, D. W., and Haider, S. (2022). Three-dimension digital twin reference architecture model for functionality, dependability, and life cycle development across industries. *IEEE Access*, 10:95390–95410.

Novak, J., Krajnc, A., and Žontar, R. (2010). Taxonomy of static code analysis tools. In *The 33rd International Convention MIPRO*, pages 418–422.

Opoku, D.-G. J., Perera, S., Osei-Kyei, R., and Rashidi, M. (2021). Digital twin application in the construction industry: A literature review. *Journal of Building Engineering*, 40:102726.

Ozkaya, M. (2024). Towards an architecture modeling language for specifying digital twin architectures using C4. In Fujita, H., Meana, H. M. P., and Hernandez-Matamoros, A., editors, *New Trends in Intelligent Software Methodologies, Tools and Techniques - Proceedings of the 23rd International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT_24), Cancun, Mexico, September 24-26, 2024*, volume 389 of *Frontiers in Artificial Intelligence and Applications*, pages 151–164. IOS Press.

Ozkaya, M. and Akdur, D. (2021). What do practitioners expect from the meta-modeling tools? A survey. *J. Comput. Lang.*, 63:101030.

Peker, O. and Akdur, D. (2019). A method for elimination of false iff target reports by using isls and rsls techniques. In *2019 Signal Processing Symposium (SPSympo)*, pages 315–318.

Peng, Y., Zhang, M., Yu, F., Xu, J., and Gao, S. (2020). Digital twin hospital buildings: An exemplary case study through continuous lifecycle integration. *Advances in Civil Engineering*, 2020(1):8846667.

Perno, M., Hvam, L., and Haug, A. (2022). Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers. *Computers in Industry*, 134:103558.

Piroumian, V. (2021). Digital twins: Universal interoperability for the digital age. *Computer*, 54(1):61–69.

Prieto-Diaz, R. (1993). Status report: software reusability. *IEEE Software*, 10(3):61–66.

Purcell, W. and Neubauer, T. (2023). Digital twins in agriculture: A state-of-the-art review. *Smart Agricultural Technology*, 3:100094.

Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., and Nee, A. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 58:3–21. Digital Twin towards Smart Manufacturing and Industry 4.0.

Rasheed, A., San, O., and Kvamsdal, T. (2020). Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012.

Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31.

Shao, G. and Helu, M. (2020). Framework for a digital twin in manufacturing: Scope and requirements. *Manufacturing Letters*, 24:105–107.

Singh, M., Fuenmayor, E., Hinchy, E. P., Qiao, Y., Murray, N., and Devine, D. (2021). Digital twin: Origin to future. *Applied System Innovation*, 4(2).

Singh, M., Srivastava, R., Fuenmayor, E., Kuts, V., Qiao, Y., Murray, N., and Devine, D. (2022). Applications of digital twin across industries: A review. *Applied Sciences*, 12(11).

Stroustrup, B. (1987). What is "object-oriented programming"? In Bézivin, J., Hullot, J.-M., Cointe, P., and Lieberman, H., editors, *ECOOP' 87 European Conference on Object-Oriented Programming*, pages 51–70, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sun, T., He, X., and Li, Z. (2023). Digital twin in healthcare: Recent updates and challenges. *DIGITAL HEALTH*, 9:20552076221149651.

TAO, F., SUN, X., CHENG, J., ZHU, Y., LIU, W., WANG, Y., XU, H., HU, T., LIU, X., LIU, T., SUN, Z., XU, J., BAO, J., XIANG, F., and JIN, X. (2024). maketwin: A reference architecture for digital twin software platform. *Chinese Journal of Aeronautics*, 37(1):1–18.

Tao, F., Zhang, H., Liu, A., and Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415.

VanDerHorn, E. and Mahadevan, S. (2021). Digital twin: Generalization, characterization and implementation. *Decision Support Systems*, 145:113524.

Verdouw, C., Tekinerdogan, B., Beulens, A., and Wolfert, S. (2021). Digital twins in smart farming. *Agricultural Systems*, 189:103046.

Wu, Y., Zhang, K., and Zhang, Y. (2021). Digital twin networks: A survey. *IEEE Internet of Things Journal*, 8(18):13789–13804.

Zenger, M. (2004). *Programming language abstractions for extensible software components*. PhD thesis, EPFL, Lausanne.