A Conceptual Model-based Systems Engineering (MBSE) approach to develop Digital Twins

Viviana Lopez
Complex Engineering Systems Laboratory
Department of Manufacturing and Industrial Engineering,
University of Texas Rio Grande Valley,
Brownsville, Texas.

Aditya Akundi
Complex Engineering Systems Laboratory
Department of Informatics and Engineering Systems,
University of Texas Rio Grande Valley,
Brownsville, Texas.

2022 IEEE International Systems Conference (SysCon) | 978-1-6654-3992-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/SysCon53536.2022.9773869

Abstract— **A digital twin (DT) is an interactive, real-time digital representation of a system or a service utilizing onboard sensor data and Internet of Things (IoT) technology to gain a better insight into the physical world. With the increasing complexity of systems and products across many sectors, there is an increasing demand for complex systems optimization. Digital twins vary in complexity and are used for managing the performance, health, and status of a physical system by virtualizing it. The creation of digital twins enabled by Model-based Systems Engineering (MBSE) has aided in increasing system interconnectivity and simplifying the system optimization process. More specifically, the combination of MBSE languages, tools, and methods has served as a starting point in developing digital twins. This article discusses how MBSE has previously facilitated the development of digital twins across various domains, emphasizing both the benefits and disadvantages of adopting an MBSE enabled digital twin creation. Further, the article expands on how various levels of digital twins were generated via the use of MBSE. An MBSE enabled conceptual framework for developing digital twins is identified that can be used as a research testbed for developing digital twins and optimizing systems and system of systems.**

Keywords—MBSE, Digital Twin, Digital Shadow, Digital Model, SysML

I. Introduction

A. Model-based Systems Engineering

Model-based Systems Engineering (MBSE) is described as the structured utilization of modeling to assist system requirements, design, analysis, verification, and validation activities starting with the conceptual design phase and continuing through the development and subsequent life cycle stages [1]. The three major components of enabling an MBSE approach are modeling languages, modeling methods, and modeling tools [2]. These three major components are used in conjunction: a tool for modeling systems, a language for carrying out the model(s), and a technique for implementing the model(s). MBSE methods refer to a collection of linked engineering processes, scientific methodologies, and associated technologies that enable systems engineering in a model-based environment. Modeling languages are used to communicate the needs and architecture of a system. Additionally, modeling tools may be used to digitize a system and its components which have been identified using the modeling language of preference. The MBSE method enables system contributors and stakeholders to collaborate throughout a system life cycle for convenient data exchange and a centralized understanding of a physical system's status by sharing information via models instead of traditional one-use documents [3].

B. An Understanding of Digital Twins from Literature

A digital twin (DT) is an interactive, real-time digital representation of a system or service utilizing onboard sensor data and internet of things technology. Data from the physical system is used to develop and enhance the digital twin by providing an accurate and consistent, real-time model of a physical system. The notion of a digital twin, first proposed in 2002 by Michael Grieves [5], is increasingly being echoed upon in MBSE landscape. A digital twin is continuously updated with the corresponding physical system and performance data throughout its system life cycle [6]. However, a review of scientific articles proved that a precise definition of a DT has yet to be developed as definitions vary across different domains. According to Kritzinger et al., there are three virtual representation levels of a digital twin. Each level has a distinct purpose and scope throughout the system's lifecycle, helping with decision-making and addressing challenges. Depending on the level of data integration, some virtual models are created manually and have no physical data from the product/systems, while others are extensively interconnected with real-time data exchange [7]. It is observed that the terms digital model (DM), digital shadow (DS), and digital twin (DT) are used interchangeably across literature based on the level of interoperability among a virtual model created and its corresponding physical system. Figure 1 attempts to illustrate the core differences between a DM, DS, and DT.

A DM is a digital depiction of a physical system that does not utilize any computerized data exchange between the physical system and the virtual model [7]. Data from the physical system is manually input, negating the real-time exchange of data between the physical system and DM. The level of complexity can only pertain to the detail of physical system components and environment. Any information gained from a DM will not directly affect the physical system. As seen in Figure 1 information about the state of the physical system is manually input by a user to the digital model. This manual exchange of information is represented in the figure as a dotted line.

A DS is all that a DM is with an addition of an integrated one-way data flow between the state of an existing physical system and the state of a virtual model [7]. Any modification made to the physical system will result in an automated update to the DS, which is accomplished via an information exchange that is processed by a database. This automatic one-way exchange of information is represented as a solid line in Figure 1. However, a change in the virtual model will not directly affect a change in the physical system. Changes determined by the DS must be manually implemented in the physical system by the user.

A DT has real-time interconnectivity between an existing physical system and the virtual model [7]. Changes in the virtual model can directly affect the physical system. The DT can also make decisions that change the performance, functionality, or status of the physical system. Other physical elements, such as the environment of a physical system, may affect the status of the DT as information is automatically transmitted through a database. A change in the physical system causes a change in the DT state and vice versa as represented by Figure 1.

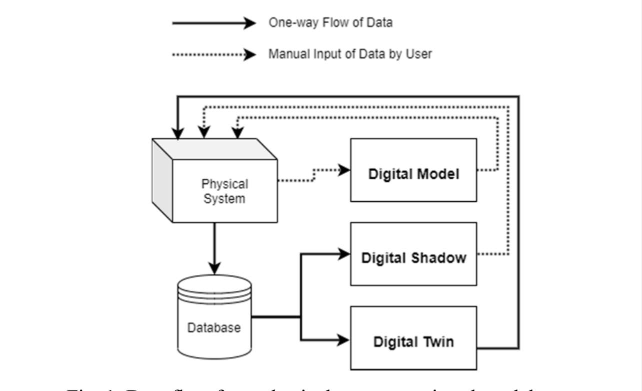

Fig. 1. Data flow from physical system to virtual model type

C. *Intersection of MBSE and Digital Twins*

The information and data obtained by a DT has seen an impact on the design and optimization of physical systems. The use of the three pillars of MBSE can serve as a DT's starting point. Processes may be streamlined by using MBSE techniques. MBSE languages, methods, and tools such as SysML, Cameo Systems Modeler, MagicGrid have been utilized to develop system models gradually [4]. SysML is a graphical language that enables users to create system specifications via the use of three kinds of model diagrams: behavior diagrams, requirement diagrams, and structure diagrams [2]. As a result, these diagrams can be translated into a programming language like Java and utilized to mimic the system model in a simulation engine [4]. Cameo Systems Modeler is a cross-platform MBSE environment that allows users to create, track, and digitize system characteristics using SysML model diagrams [4] [14] [19]. System stakeholders and contributors can then easily track system models and those models are then saved as XMI files, or distributed to documents, graphics, and web interfaces. MagicGrid enables separating the process of creating a system model into three domains: problem, solution, and implementation [4] [14] [19]. Comparing simulation outputs to actual results can reveal important information about the physical system's performance, health, and status. Engineers can create event-driven or agent-based simulations to investigate the behavior and interactions of the DT using an appropriate MBSE tool and language [6]. To encourage synchronous model creation and improve model data re-usability, MBSE offers a standard guideline for system management, system-to-system architecture, and operational scenarios. MBSE allows users to collect model data from engineering and manufacturing products and processes. Users of MBSE may utilize modeling and simulation data to generate a DT of a physical system at each stage of its correspondent lifecycle phase [3]. The implemented MBSE method depends on functional, operational, and other system requirements to generate a DT, which reflects system behavior and functionality. The analysis of a system's requirements, behavior, structure, and parameters, as well as their representation in a modeling language such as SysML, have previously been used in the instance of creating a DT using MBSE. Integrating MBSE aids in the establishment of synchronization across different engineering disciplines such as structural, technical, inspection, software, and other various elements of a physical system [8].

II. The Use of MBSE for Developing Digital Twins – A Brief Exploration

In this section, we explore how MBSE has been previously used for DT development across various industry domains. Within the aerospace domain, assembling and running a DT for system function, physical impacts, operational environment, and purpose were explored [9]. An ice protection system for a regional airplane was studied to see whether an MBSE enabled DT might handle lifecycle components more effectively. Physical processes and performance were specified using a DT. Using interoperability standards to connect a DT from the simulation environment has been proven to significantly reduce the amount of effort needed to join large IT data systems. However, MBSE tool integration is still hindered by various proprietary languages, syntax, and formats. Due to increasing digitization and system complexity, a DT has been used to optimize, assess alternatives, reduce faults, and replace physical mockups and prototypes [9]. A technique for applying digital twins and MBSE to civil aviation, emphasized the advantages and disadvantages of DT implementation within product family development [10]. For a DT, system variation must be based on both a physical prototype and customer requirements. When there are fewer variations and more continuous production, product families need more information processing and management. By developing modular product families, which includes various process elements for reducing internal system variety while leaving external variety unchanged, the combined Product Development and Mechanical Engineering Design (PKT)-Approach data model was used to analyze the product structure of an aircraft cabin monument and its processes [10]. Using an MBSE approach during system development assists in the handling of complicated and big datasets. MBSE has also been utilized to build a DT for multi-UAV swarms. Standalone MBSE tools cannot provide model integration, tool interoperability, experimentation, or data collection [11]. Therefore, MBSE has been observed to be used as a research testbed for aeronautical systems and systems-of-systems which allows for the development and testing of abstract models. An environment for modeling, simulation, and integration is essential in the MBSE testbed. When users start with a comprehensive set of operating limitations and requirements using MBSE, DT development is not only streamlined but will ensure system safety [11]. To create and analyze a DT virtual simulation with real-world system components gives access to real-time interconnected data. Using MBSE as a foundation for a physical system to develop a DT allows for future aeronautical and automotive optimization.

Using a DT in a model-based environment may also help optimize manufacturing shop floors and machine layout prior to building and implementation. To predict physical disturbances and create preventive actions, a DT was built for a material handling system [12]. A DT with a physics-based simulation that could anticipate certain disruptions was developed for this material handling system. With such a high degree of system complexity, an MBSE approach proved successful in ensuring anticipated system functionality and advantages, while reducing implementation disruptions. A system user may visualize abnormalities through the DT. The DT can have the ability to detect problems automatically by monitoring the system's performance and health. Using a DT can provide cost-effective traceability and MBSE procedures. There is a large need for increased DT usage in all types of production systems to identify, prevent, and mitigate potential problems. One such application is the use of MBSE to build a DT for streamlining a manufacturing shop floor using SysML and MagicGrid [4]. The development of both a DM and DT allows for controlled, reusable, and traceable system data. The DT integrates real-time data via state cycle scanning, making it more sustainable than prior document-based systems. A physical system is not needed if the equipment and configurations are predetermined. While MBSE can manage data interconnection, further MBSE and DT implementation is required to improve DT research capabilities [4].

An MBSE enabled DT was developed to improve the shipbuilding process by reducing errors and increasing efficiency [3]. An MBSE approach enabled to view digital and physical manufacturing processes by creating digital twins of physical systems, thus improving operational procedures and data traceability in both the real and virtual environments. MBSE was also used to create a DT by expanding a product model [14].

A. *Benefits and Challenges of developing an MBSE enabled DT.*

Controlling the MBSE process to create a system or service becomes difficult due to factors such as imprecise design information, interconnection of design functions, and changes in design progress where product development becomes too time-consuming. Because MBSE retains comprehensive information on not only how the system was created but also how it may be improved, using an MBSE enabled DT generates privacy and ownership concerns, as many people are concerned about widespread data sharing with suppliers and prospective customers [6]. However, because MBSE enables organized data storage, the information obtained from a DT may be used for a variety of other purposes, such as performing quality checks at the conclusion of the manufacturing process or throughout the remainder of the product's life cycle. For product families, the interconnectivity of information provided by MBSE about current products into the DT of the product family may be utilized in future lifecycle stages, resulting in a flood of new advantages. This information may be utilized to create new product variations or the next generation of current products [10]. Additionally, by combining DT with MBSE, it is possible to develop a strategy for more effectively addressing environmental and other problems, since potential modifications may be validated using an MBSE methodology rather than physically testing them, resulting in the usage of less material and energy over time [3].

III. An MBSE Conceptual Framework for Developing Digital Twins

In this section, guided by the observed use of MBSE approach to develop digital twins across various domains (Section II), we attempt to map a theoretical MBSE enabled conceptual framework for developing a DT. To design, develop, and implement a physical system, system requirements must first be defined to develop a system design. These system requirements are then visualized and modeled using a variety of MBSE modeling languages. Identifying these requirements is crucial for the development of a DT.

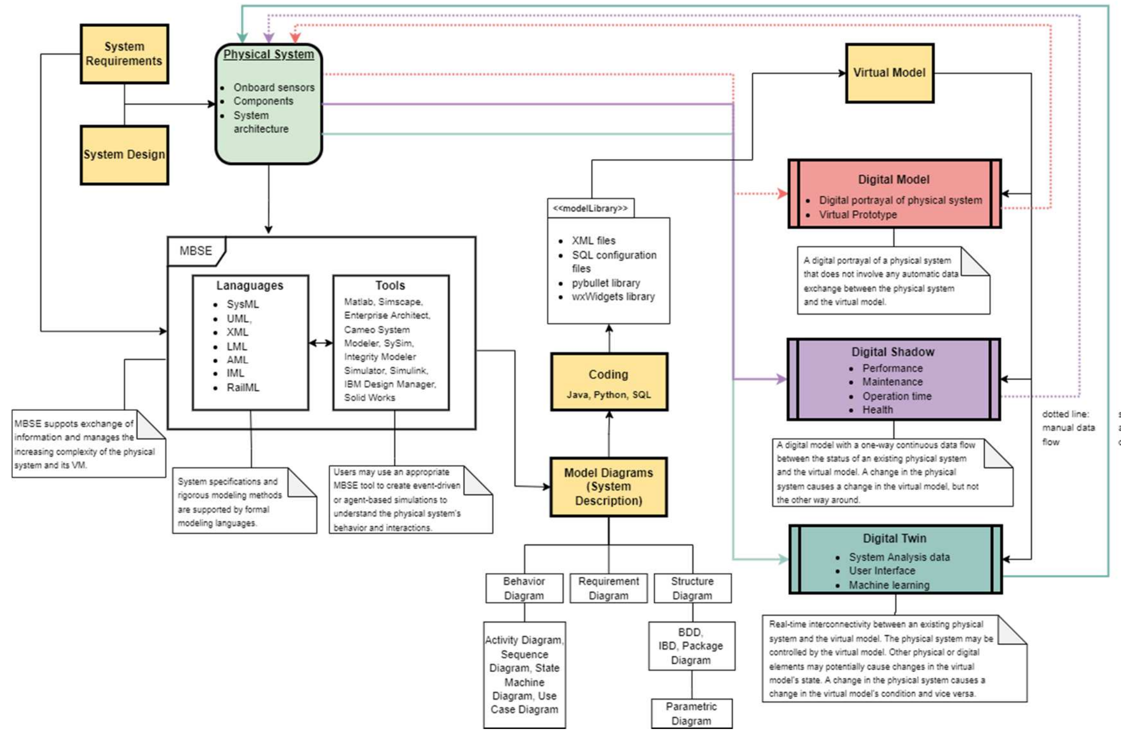
System Requirements
System Design
Physical System
Onboard sensors
Components
System architecture
MBSE
Lanaguages
SysML
UML,
XML
LML
AML
IML
RailML
Tools
Matlab, Simscape, Enterprise Architect, Cameo System Modeler, SySim, Integrity Modeler Simulator, Simulink, IBM Design Manager, Solid Works
MBSE suppots exchange of information and manages the increasing complexity of the physical system and its VM.
System specifications and rigorous modeling methods are supported by formal modeling languages.
Users may use an appropriate MBSE tool to create event-driven or agent-based simulations to understand the physical system's behavior and interactions.
<<modelLibrary>>
XML files
SQL configuration files
pybullet library
wxWidgets library
Coding
Java, Python, SQL
Model Diagrams (System Description)
Behavior Diagram
Requirement Diagram
Structure Diagram
Activity Diagram, Sequence Diagram, State Machine Diagram, Use Case Diagram
BDD, IBD, Package Diagram
Parametric Diagram
Virtual Model
Digital Model
Digital portrayal of physical system
Virtual Prototype
A digital portrayal of a physical system that does not involve any automatic data exchange between the physical system and the virtual model.
Digital Shadow
Performance
Maintenance
Operation time
Health
A digital model with a one-way continuous data flow between the status of an existing physical system and the virtual model. A change in the physical system causes a change in the virtual model, but not the other way around.
Digital Twin
System Analysis data
User Interface
Machine learning
Real-time interconnectivity between an existing physical system and the virtual model. The physical system may be controlled by the virtual model. Other physical or digital elements may potentially cause changes in the virtual model's state. A change in the physical system causes a change in the virtual model's condition and vice versa.
dotted line: manual data flow
solid line: automatic data flow

Figure 2.0 MBSE enabled process to develop varying virtual models from a physical system

Second, modeling tools can then be implemented to conform with the standards of a specified modeling language, allowing users to build complex model diagrams. Every time a user makes a change to a feature on a diagram created by a modeling tool, it is then changed to the specific diagram itself and other connected model diagrams. This interconnectivity allows for an organized and streamlined system design and implementation process.

The most utilized modeling language identified was SysML which represents system structure, behavior, requirements, and restrictions. SysML is an extension of UML, and some of its rules are specified in the UML standard [2]. There are nine types of SysML diagrams as illustrated in the lower section of Figure 2.0: block definition diagram (BDD), internal block diagram (IBD), use case diagram, activity diagram, sequence diagram, state machine diagram, parametric diagram, package diagram, and requirements diagram. Please refer to the sources listed in Table I for additional information on how the nine types of SysML diagrams were observed to be used to create digital twins and how they were implemented.

TABLE I. SysML Diagram Types

BDD Diagram	BDD can be used to depict system elements like blocks and value types and their connections. A BDD is often used to illustrate system hierarchy and categorization trees [4] [15] [16] [17] [18].
IBD Diagram	The IBD describes a block's internal structure. An IBD can be used to display the interfaces between the internal sections of a block [4] [13] [15] [19] [20].
Use Case Diagram	The use case diagram depicts a system's actions and the actors who initiate and participate in them. A use case diagram illustrates the services provided by a system in conjunction with its actors [4] [19] [21].
Activity Diagram	The activity diagram is used to describe a behavior, focusing on system logic and the translation of inputs into outputs. Activity diagrams were frequently used to analyze and describe intended system behavior [4] [15] [19] [20] [22] [23].
Sequence Diagram	The sequence diagram is used to describe a process. To accurately define a process for the development stage of the life cycle, sequence diagrams are frequently employed. Sequence diagrams are also a great resource for defining test scenarios [4] [12] [19] [20].
State Machine Diagram	A state machine diagram is used to describe a block's state and potential changes. In the development stage of a systems life cycle, state machine diagrams are used to describe a block's behavior [4] [9] [15] [21] [22].
Parametric Diagram	The parametric diagram can be used to show how equations and inequalities are linked to design parameters [13] [14] [20].
Package Diagram	The package diagram shows a model's package containment structure. A package diagram illustrates model components included in packages, their dependencies, and the connections between requirements [4].
Requirements Diagram	The requirements diagram is used to illustrate text-based requirements and their connections to other model components that fulfill, verify, and improve them. The requirements diagrams are used to link subsystem functionality to the systems' needs and create a baseline for future DT development [4] [13].

Only when an appropriate MBSE tool is employed, is the quantity and combination of model diagrams proportional to the complexity of the virtual model type. Since the level of MBSE integration varies across different domains, tools and data exchange components must be considered when developing the desired virtual model type. A DM, for example, may be robust in its representation of various system components and may also suggest improvements, but it will not be able to transmit data in real-time. Regardless of the complexity of the virtual models, a DT is defined by its ability to transfer data in real-time to and from the physical system and virtual model, as defined by the different virtual model types. It is important to emphasize that, although digital twins may be created using an MBSE framework, the benefit of MBSE is that complicated systems can be structured, streamlining system design and implementation. Table II maps the type of SysML diagrams that can be used to facilitate the development of virtual models i.e. either a DM, DS, or a DT.

TABLE II. Virtual Model Types and the Corresponding SysML Model Diagram Required

Digital Model	State Machine Diagram, BDD, IBD, Parametric Diagram Use Case Diagram, Activity Diagram, Requirement Diagram, and Sequence Diagram
Digital Shadow	BDD, IBD, Package Diagram, Parametric Diagram, and Requirements Diagram, Sequence Diagram, Activity diagram, Use Case Diagram, and State Machine Diagram
Digital Twin	BDD, IBD, Use Case Diagram, Activity Diagram, Sequence Diagram, State Machine Diagram, Package Diagram, and Requirements Diagram

The next stage in this procedure is to create data connectivity through executable program files that are kept in a database or model library and are written in a suitable programming language. File reading and writing enable the required information to be transformed into a simulated virtual model once the system model data has been processed [4] [19] [24]. One method for doing this is to generate a SQL configuration file, retrieve the database connection information from the file, and then establish the connection once the DT begins to operate. The SQL instructions are then sent to a database, and the returned results are stored on a respective MBSE modeling tool. Once the query results are obtained, real-time information exchange between the physical system and virtual model is established. The kind of program files generated will vary according to the tool(s) utilized, as well as the system communication devices and database(s) used. Depending on the desired virtual model type, information gathered from changes in either the physical system or virtual model are implemented manually or automatically. In addition to the three levels of a virtual model shown in Figure 1., those three levels of a virtual model are also depicted in Figure 2. Finally, the type of virtual model that is generated will be determined by not only the amount and combination of model diagrams utilized but as well as the method by which data is transferred between the physical system and virtual model will also be determined. In Figure 2., the type of data exchange is represented by either a dotted or a solid line, in the same way as it is in Figure 1. Manual data exchange is depicted as a colored dotted line. A user will have to manually make changes in either the physical system or the virtual model, and those changes will not be reflected in either until the user makes the adjustments manually [7]. The automatic (real-time) exchange of data is depicted by a colored solid line. In the case of a DS and DT, information is supplied into the virtual model type in real-time. The most important distinction between a DS and a DT is that only a DT has the capacity of making real-time modifications to the physical system, while a DS does not have this capability. A greater degree of a virtual model may be created if more sophisticated model diagrams are used and the capacity of a physical system to transmit information becomes more complex.

IV. Conclusion and Futurework

It is observed that digital twin (DT) technologies are still in their infancy and additional research is needed in implementation framework design, data processing, storage, and security for digital twins using enabled by MBSE. The current integration efforts by the research community on MBSE and DT development has so far shown that the benefits outweigh its challenges. In this paper, an attempt to map an MBSE enabled DT development framework is portrayed based on the literature observed in facilitating digital twins using MBSE languages, tools, and techniques. The authors are currently working on exploring the applicability of the identified framework to develop digital shadow and digital twins and verify the repeatability. Further research is needed to determine the benefits of each virtual model type and the correlation between system complexity and DT system optimization.

Acknowledgment

This work was partially supported by the National Science Foundation under grant 2112650 (CREST - Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS)), and University of Texas Rio Grande Valley (UTRGV) Presidential Research Fellowship (PRF) Award . The authors wish to express sincere gratitude for their financial support.

Disclaimer: "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or UTRGV PRF Award Office."

References

[1] Friedenthal, Sanford, Regina Griego, and Mark Sampson. "INCOSE model-based systems engineering (MBSE) initiative." In INCOSE 2007 symposium, vol. 11. 2007.

[2] Delligatti, Lenny. SysML distilled: A brief guide to the systems modeling language. Addison-Wesley, 2013.

[3] Pang, Toh Yen, Juan D. Pelaez Restrepo, Chi-Tsun Cheng, Alim Yasin, Hailey Lim, and Miro Miletic. "Developing a digital twin and digital thread framework for an 'Industry 4.0'Shipyard." Applied Sciences 11, no. 3 (2021): 1097.

[4] Liu, Juan, Jianhua Liu, Cunbo Zhuang, Ziwen Liu, and Tian Miao. "Construction method of shop-floor digital twin based on MBSE." Journal of Manufacturing Systems 60 (2021): 93-118.

[5] Grieves, Michael, and John Vickers. "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems." In Transdisciplinary perspectives on complex systems, pp. 85-113. Springer, Cham, 2017.

[6] Madni, Azad M., Carla C. Madni, and Scott D. Lucero. "Leveraging digital twin technology in model-based systems engineering." Systems 7, no. 1 (2019): 7.

[7] Kritzinger, Werner, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. "Digital Twin in manufacturing: A categorical literature review and classification." IFAC-PapersOnLine 51, no. 11 (2018): 1016-1022.

[8] Phanden, Rakesh Kumar, Priavrat Sharma, and Anubhav Dubey. "A review on simulation in digital twin for aerospace, manufacturing and robotics." Materials Today: Proceedings 38 (2021): 174-178.

[9] Bachelor, Gray, Eugenio Brusa, Davide Ferretto, and Andreas Mitschke. "Model-based design of complex aeronautical systems through digital twin and thread concepts." IEEE Systems Journal 14, no. 2 (2019): 1568-1579.

[10] Laukotka, Fabian, Michael Hanna, and Dieter Krause. "Digital twins of product families in aviation based on an MBSE-assisted approach." Procedia CIRP 100 (2021): 684-689.

[11] Madni, Azad M., Dan Erwin, and Carla C. Madni. "Digital Twin-enabled MBSE Testbed for Prototyping and Evaluating Aerospace Systems: Lessons Learned." In 2021 IEEE Aerospace Conference (50100), pp. 1-8. IEEE, 2021.

[12] Glatt, Moritz, Chantal Sinnwell, Li Yi, Sean Donohoe, Bahram Ravani, and Jan C. Aurich. "Modeling and implementation of a digital twin of material flows based on physics simulation." Journal of Manufacturing Systems 58 (2021): 231-245.

[13] Kutzke, Demetrious T., James B. Carter, and Benjamin T. Hartman. "Subsystem selection for digital twin development: A case study on an unmanned underwater vehicle." Ocean Engineering 223 (2021): 108629.

[14] Wang, Yübo, Tanja Steinbach, Jonathan Klein, and Reiner Anderl. "Integration of model based system engineering into the digital twin concept." Procedia CIRP 100 (2021): 19-24.

[15] Brusa, Eugenio. "Digital Twin: Towards the Integration Between System Design and RAMS Assessment Through the Model-Based Systems Engineering." IEEE Systems Journal (2020).

[16] Schluse, Michael, Marc Priggemeyer, Linus Atorf, and Juergen Rossmann. "Experimentable digital twins—Streamlining simulation-based systems engineering for industry 4.0." IEEE Transactions on industrial informatics 14, no. 4 (2018): 1722-1731.

[17] Schluse, Michael, Linus Atorf, and Juergen Rossmann. "Experimentable digital twins for model-based systems engineering and simulation-based development." In 2017 Annual IEEE International Systems Conference (SysCon), pp. 1-8. IEEE, 2017.

[18] Delbrügger, Tim, and Jürgen Rossmann. "Representing adaptation options in experimentable digital twins of production systems." International Journal of Computer Integrated Manufacturing 32, no. 4-5 (2019): 352-365.

[19] Tsui, Roy, Devin Davis, and John Sahlin. "Digital Engineering Models of Complex Systems using Model - Based Systems Engineering (MBSE) from Enterprise Architecture (EA) to Systems of Systems (SoS) Architectures & Systems Development Life Cycle (SDLC)." In INCOSE International Symposium, vol. 28, no. 1, pp. 760-776. 2018.

[20] Wang, Haoqi, Hao Li, Xiaoyu Wen, and Guofu Luo. "Unified modeling for digital twin of a knowledge-based system design." Robotics and Computer-Integrated Manufacturing 68 (2021): 102074.

[21] Hause, Matthew. "The Digital Twin Throughout the SE Lifecycle." In INCOSE International Symposium, vol. 29, no. 1, pp. 203-217. 2019.

[22] Walter, Benedikt, Dennis Kaiser, and Stephan Rudolph. "From Manual to Machine-executable Model-based Systems Engineering via Graph-based Design Languages." In MODELSWARD, pp. 201-208. 2019.

[23] Tarabini-Castellani, L., V. Gómez, J. Fombellida, S. Ramirez, N. Puente, R. Contreras, and R. Haya. "Lessons learned from the use of SysML in Space Systems at SENER Aeroespacial."

[24] Eisenträger, Marlene, Simon Adler, Matthias Kennel, and Sebastian Möser. "Changeability in Engineering." In 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 1-8. IEEE, 2018.