Modeling and Observing ProtoDigital Twin with ROS2 and Omniverse

Alican Tüzün
University of Applied Sciences Upper Austria
Wels, Austria
tuzunalican@gmail.com

***Abstract*—Digital twins are becoming more and more important for the efficient and effective development and operation of cyber-physical systems. However, digital twins are only useful if they reflect the real-world system accurately enough, i.e.their quality is high enough. This claim entails the question, of what the term quality in the context of digital twins means and how it can be measured. In this article, we present our experience with the quality assurance of a digital twin for an assembly line in the automotive industry. We explain our preliminary definition of digital twin quality, which we derive from classical quality models for general software systems. Furthermore, we describe quality issues, which we were able to detect in a digital twin of an assembly line in the automotive industry. Finally, we conclude how to leverage our experience in different contexts and how to generalize the underlying approaches.**

I. Introduction

As new and complex technologies advance to emerge, it becomes progressively difficult to understand and define their core concepts. One of the significant current examples in the field of digitalization is the notion of digital twins, which has resulted in numerous definitions [22], that vary from one derivation to another. Because, even though the terms "digital" and "twin" are easy to grasp, their consolidation develops a new challenging concept, which might be resulted in misunderstanding and misapplication. Therefore, a practical example of a proto, easy-to-model example of the digital twin prototype and instance can significantly improve understanding of the notion of the digital twin.

The digital twin (DT) was proposed in 2002 by Grieves [20], as an ideal form of product life cycle management. The proposal was to create a co-existed information twin of a real system to gather new information to minimize the system needs and waste, such as material, time, and energy. It should be noted that initially this concept was called the mirrored spaced model [20], and later the information mirroring model (IMM) [18], [19]. IMM had all the components of today's DTs: a real system (RS), a virtual system, a connection between them, and additionally the virtual simulation [18]. However, after co-authoring with Vickers in 2010, the term 'digital twin' was adopted and Grieves simplified the model by excluding the virtual simulation component [20].

Since then, there has been a considerable amount of literature published on DTs [22], [21], however, to date, even though Grieves has already given the definition [20], there has been little agreement on the precise definition and application. Numerous authors have considered different definitions, including digital twins as a multi-physics environment [17], an equivalent to a product [26], a digital copy [27], a cyber component of a Cyber-Physical System [16], and many more [28], [23], [25]. However, if inspected carefully, most of these not only don't coincide with Grieves's definitions but also give some extra aspects to it.

This absence of agreement exists not only in the literature; it also reaches the industry. This resulted in the development of many different general platforms to construct digital twins [3], [12]. Furthermore, game engines such as Unity [11] and Unreal Engine [7] have become popular tools for being an environment for digital twins, and some of them already offer dedicated digital twin platforms [9], [2]. However, while there are some accessible open-source projects [4], [6], most of the digital twin platforms today are not free to use, which makes it harder to acquire practical knowledge about digital twins.

To address the challenges associated with the accessibility and comprehensibility of the notion of digital twins, the author presented the process of assembling a real system along with a digital twin prototype (DTP) [20]and digital twin instance (DTI) [20], both being forms of DT, by following adopted lifecycle models. Furthermore, the author presented the observations, including the challenges and solutions encountered during the process.

During the implementation, the author utilized Omniverse [8] as a digital twin environment (DTE) [20] and used ROS2 (Robot Operating System 2) [10] as a sensor and communication middleware and gave behavior to the real system. An HC-SR04 ultrasonic sensor [5], Raspberry Pi 4(RPI4) [1], basic electronic components and several third-party libraries have been used to construct the real system.

II. Method

To start the process of developing the digital twins and real system (RS), first, a digital prototype (DTP) representing the possible form of the RS, also called a selected prototype, was developed. Second, the physical components of the RS relative to the DTP were assembled. Subsequently, a connection between the RS and the digital twin instance (DTI) was established, resulting in the capturing of information about the product throughout the remaining lifecycle stages.

Nonetheless, the Digital Twin (DT) is a compendium of product information and evolves through the entire lifecycle of the product. Therefore, it was important to recognize the DTP and DTI's lifecycles, which most of the time run in parallel with the product lifecycle.

Therefore, the author explored different lifecycle frameworks and standards [13], [15], [14] and adapted the lifecycle models from ISO/IEC 24748-1 to the notion of digital twins. These frameworks include the stages of Concept, Development, Support&Maintenance, and finally Retirement [14]. However, to effectively demonstrate the construction of the digital twins, the author presented the creation of the DTP and DTI from the perspective of the real system's lifecycle stages. This approach involved an additional stage called production, which includes the process of production of the RS [14].

It should be noted that, in the ideal world, there would be no need for a development or a concept phase for the RS. However, currently, it is not yet feasible to completely digitalize every aspect of the real system, hence DTI and DTP were used to assist the RS through its lifecycle by transforming atoms to bits [24].

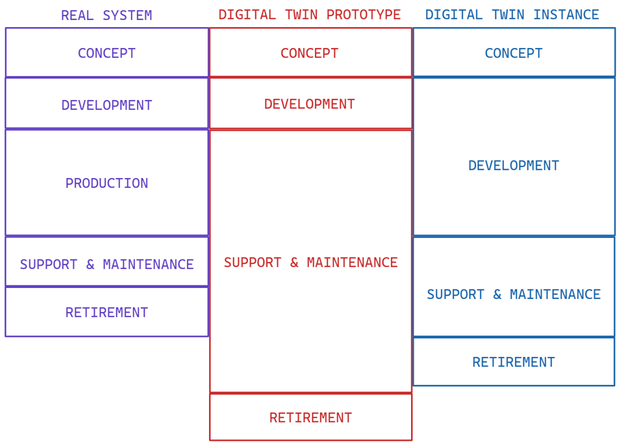

Fig. 1. All three lifecycles of RS, DTP, and DTI have been illustrated in order with three different colours.

It should be noted that, in the ideal world, there would be no need for a development or a concept phase for the RS. However, currently, it is not yet feasible to completely digitalize every aspect of the real system, hence DTI and DTP were used to assist the RS through its lifecycle by transforming atoms to bits /ref.

A. Concept Stage

This stage was responsible for exploration, fact-finding, and initial planning while considering the economic, technical, and strategic aspects of the system, in alignment with the desires of the stakeholders.

To ensure efficiency and productivity, a fast-paced requirement engineering approach was applied, which resulted in the identification of functional and quality requirements, as well as constraints for the DTP, DTI, and RS. Once the requirement analysis was done, and the necessary approvals were obtained, the RS proceeded to the Development Stage.

B. Development Stage

During the development stage, the requirements and design solutions for each system were transformed into feasible products which were used in the following stages of the lifecycle. The primary objective of this stage was to develop hardware and software models as well as their corresponding interfaces. These developed models were first analyzed and later verified, and validated.

Even though the inclusion of information about the assembly and production planning, maintenance and support procedures as well as retirement considerations, and various other aspects should be ideally present in this stage, they were not included in this study due to time and budget constraints.

As a result of this stage, the RS was digitally developed and assembled within the DTE as a DTP. This DTP was prepared to assemble the physical components of the RS in the physical, observable environment. After the verification and validation process of the developed RS within the DTE was finished, RS proceeded to the Production Stage.

C. Production Stage

The production stage of the RS was constrained by the development stage of the DTP, which progressed with the assistance of DTP in the observable environment. As an outcome, the RS was initialized in the observable world and was ready for operation during the support and maintenance stage. However, another constraint, the development stage of the DTI, constrained this initiation. Therefore, the development of the DTI was completed before the initiation of the support and maintenance stage of the RS. After the development, the connection between the DTI and RS was established, tested, and evaluated to ensure data capture during the operation.

D. Service and Maintenance Stage

After the production stage, the RS and DTI started to function simultaneously. This functionality or rather a behavior was demonstrated to both technical and non-technical individuals, and this way, the usage of the system was simulated. However, the feedback and assessment from these individuals, such as the quality of the system relative to the user perspective, was out of scope for this paper and hence was not included. Furthermore, since there was no need for modifications while using/operating the system, as a result, maintenance considerations were excluded. As an outcome of this stage, the real system was ready to be put out of service, in other words, was ready to retire.

E. Retirement Stage

At this stage, the RS was decommissioned and the reusable parts were collected. Simultaneously, DTI and DTP were commencing to service state as the product retired. Since there was no significant waste generated, there was no need for a disposal process.

III. Implementation

By following the lifecycle framework in Methods, the author ended up using several hardware and software to conceptualize, develop, produce, operate, and retire the RS along with the DTI and DTP.

First, in the concept stage the requirements were gathered, listed finally categorized into three different groups. Second these requirements were revised and elicitated into more requirements before the development stage. In the development stage, these requirements were fulfilled with selected software and hardware, and the real system was developed digitally, and later integrated within the Omniverse. Next digital twin prototype behavior is written with C++ along with the Robot Operating System Nodes as a ROS2 Package. These nodes were connected to the DTE with the ROS2 Bridge, hence the behavior of the digital twin prototype against the requirements was tested, validated, and verified. In the production stage, the real system was successfully assembled in the observable environment according to the digital twin prototype. Next, the digital twin prototype was copied and developed further, to make a seamless connection between the digital twin instance and a real system. After the development of the digital twin instance was finished, the connection between the real system and the digital twin instance was tested. In the support & maintenance stage, the behavior of the real system and digital twin instance is observed by the individuals, which resulted in great feedback, which proved subjectively, the seamless connection between the real system and digital twin instance with required fidelity. After the observations, the system got into the retirement stage, and the digital twin instance and later digital twin prototype were retired.

A. Implementation of the Concept Stage

For the concept stage, first, the author focused on the requirement analysis and gathered all the requirements for the possible system. Software such as Obsidian along with the Excalidraw community plugin was used to conceptualize the systems and most importantly for the documentation of the requirements. Each system requirement was considered separately and later put together in the table. Requirements for the DTE were not considered.

Later, several prototyping not only digital but also partially physical prototyping was done. Requirements were revised, elicitated, and reworked, which resulted in a feasible solution with the selected components to develop a digital twin prototype within the Omniverse.

B. Implementation of the Development Stage

After these requirements, the development of the DTP was started. To create a DTP of the planned RS, two different software programs were used for two different purposes including Fusion 360, a Computer-Aided Design software, and Blender, a 3D-Graphics software. Later the modeled DTP is integrated into the Omniverse, and the behavior is given with the ROS2 node.

RP 1.1	Digital Twin Prototype should have high fidelity.
RP 1.2	Digital Twin Prototype must be free to build.
RP 1.3	Digital Twin Prototype should twin the behavior of the notifications.
RP 1.4	Digital Twin Prototype should have Red, Green, and Blue LEDs.
RI 1.1	Digital Twin Instance should have high fidelity.
RI 1.2	Digital Twin Instance must be free to build.
RI 1.3	Digital Twin Instance should twin the behavior of the notifications with the real system in real-time.
RI 1.4	Digital Twin Instance should have Red, Green, and Blue LEDs.
RS 1.1	The real system should cost no more than 200 euros.
RS 1.2	The real system should notify the user if the digital twin instance is active after 3.5 secs.
RS 1.2.1	The real system should have one Blue LED lamp for the RS1.2
RS 1.3	The real system should notify the user if something is within the 10 cm at the front of the HC-SR04 ultrasonic sensor.
RS 1.3.1	The real system should have one Red LED lamp for the RS 1.3
RS 1.4	The real system should notify the user if there is nothing within the 10 cm at the front of the HC-SR04 ultrasonic sensor.
RS 1.4.1	The real system should have one Green LED lamp for the RS 1.4
RS 1.5	The real system notifications should be visible in industrial real-time, meaning less than 10 Ms.
RS 1.6	The real system notifications should be visible at night.
RS 1.7	The real system should be able to connect to a Wi-Fi network for remote monitoring and control.
RS 1.8	The real system software should be free to use.
RS 1.9	The real system should be compatible with the selected digital twin environment.
RS 1.10	The real system should have a microcontroller to give it controllability.

Fig. 2. Incorrect (a) and incomplete (b) status has been demonstrated with a simplified finite state machine. Incorrectness can be observed in the behavior of the machine after 2 state transitions. Incompleteness, on the other hand, can be observed in the missing state. It should be noted that, even though the second state machine is missing a component, it shows the right behavior, hence it is correct.

First, pre-existing digitalized possible RS components were sourced from the internet, and modified for use. The ones that were not found, such as jumper cables, were developed by the author in the Fusion 360. Once all the required components were available and the model reached the required fidelity level, detailing was stopped and parts were digitally assembled. Lastly, the assembled digital model was converted to a .fbx file.

After the conversion, the Blender was used, to convert a .fbx file to a .usd file for the intention of importing into the DTE. However, before the conversion, adjustments were made to the .fbx file to fix issues, such as corrupted visuals and inherited hierarchy problems, which occurred due to the insufficient performance of the conversion. Later, the corrected file was converted into a .usd file.

Finally, Blender was reopened, and the previously converted .usd file was imported. Further adjustments were made to the file and were saved again as a .usd file. Upon completion of this process, the file was prepared for importation into the DTE.

Furthermore, headless Ubuntu 20.04 was installed on a micro-SD card, and inserted into the RPI4. Second, the cryptographic network protocol SSH was used to connect the RPI4 to a local computer that already had Ubuntu 20.04. Subsequently, the Robot Operating System 2 (ROS2) version of Foxy was installed on both the RPI 4 and the local computer. Lastly, to access and control the GPIO pins of the RPI4, the Wiring Pi library was installed on RPI4.

ROS2 was used as a sensor and communication middleware to give behavior to the real system components as well as to digital twin instances. Therefore, a custom package was developed with C++ to read data from the ultrasonic sensor and control the state of three LEDs. This behavior was integrated into a single node, which was executed with a single-thread executor. Lastly, the combined ROS2 node was run, and the required behavior of the real system was observed in the digital twin prototype to ensure that the three LEDs were giving the possible state of the RS.

The chosen DTE for this demonstration was Isaac Sim, which is fully integrated into the NVIDIA Omniverse platform. Isaac Sim was designed specifically for the training and testing of complex robotic systems in realistic virtual environments. Despite the simplicity of the demonstration, which was not a complex robotic system, there were two crucial reasons for the platform selection. First, Isaac Sim offers a ROS2 bridge, which enabled a seamless connection between ROS2 and Omniverse, facilitating relatively easy and smooth data exchange. Second, it leverages RTX Graphics, which delivered high-performance graphics rendering, resulting in an improvement in visual fidelity.

First, to initialize the DTE, Isaac Sim was installed on a local computer using the Omniverse launcher. Once the installation was completed, the correct ROS2 bridge was selected. At this step, it was important not to source the ROS2 Workspace within the same terminal that Isaac Sim was operating, for example with the automatic sourcing via bashrc, to prevent potential compilation errors.

After the initialization, the modified .usd file was imported into the Isaac Sim scene. After the import, a visual inspection of the individual parts was then conducted, and needed adjustments were made.

Next, to integrate the behavior, visual scripting has been used with the integrated ROS2 bridge, which resulted in the integrated behavior. Later behavior of the LEDs was tested, and needed corrections have been made. As a consequence, the required level of fidelity of the DTP, within the DTE was reached successfully, and the next step, which was the assembly of the real system was ready.

C. Implementation of the Production Stage

First, the components were assembled on the prototype breadboard in accordance with the DTP. Next, circuit components were connected with the jumper cables to the appropriate general-purpose input/output (GPIO), ground, and power pins of Raspberry Pi 4 (RPI4).

After the assembly, the DTI interfaces were developed and the ROS2 node from the DTP was copied and later modified, to connect the RS to the DTI. It should be noted that only the behavior of the LEDs was twinned and not the other attributes of the RS such as spatial data of the components were not. This connection was tested visually.

After the development of the DTI was finalized, the RS was ready to proceed to the operation in the support and maintenance stage.

D. Implementation of the Support and Maintenance Stage

RS, DTI, and DTP were shown to the stakeholders and their assessment of the quality attributes was written. After all, stakeholders saw the RS, DTI, and DTP, the RS was ready to be decommissioned with the corresponding DTI.

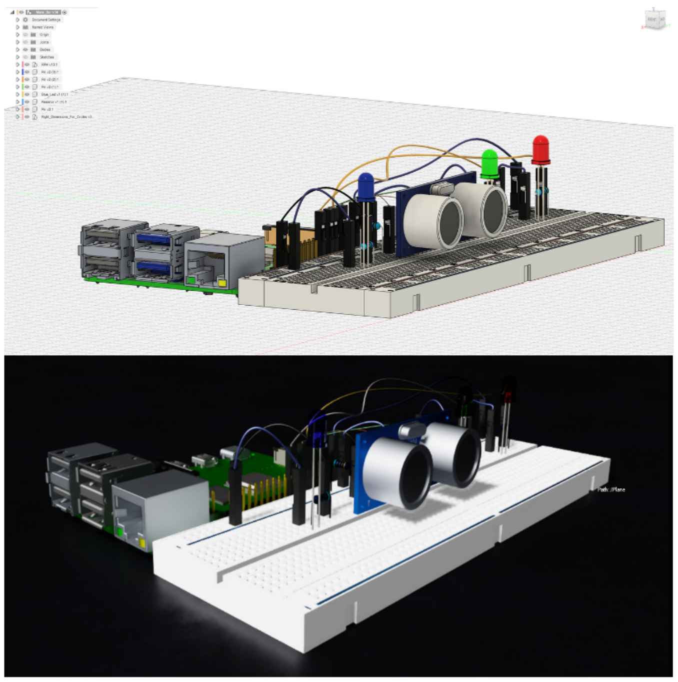

Fig. 3. Incorrect (a) and incomplete (b) status has been demonstrated with a simplified finite state machine. Incorrectness can be observed in the behavior of the machine after 2 state transitions. Incompleteness, on the other hand, can be observed in the missing state. It should be noted that, even though the second state machine is missing a component, it shows the right behavior, hence it is correct.

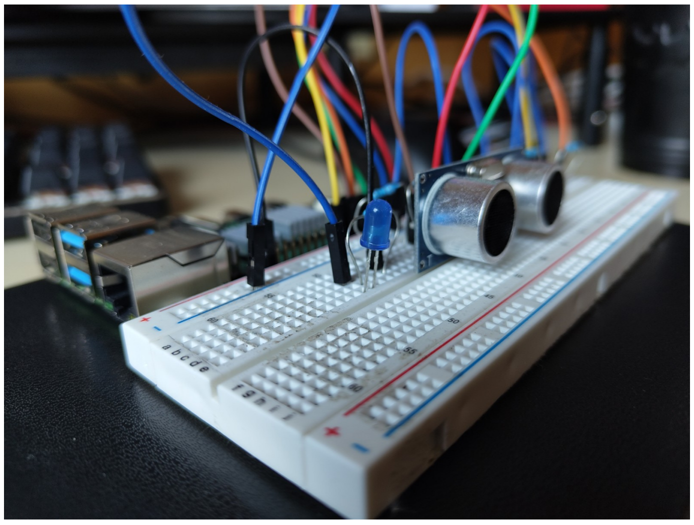

Fig. 4. Incorrect (a) and incomplete (b) status has been demonstrated with a simplified finite state machine. Incorrectness can be observed in the behavior of the machine after 2 state transitions. Incompleteness, on the other hand, can be observed in the missing state. It should be noted that, even though the second state machine is missing a component, it shows the right behavior, hence it is correct.

E. Implementation of the Retirement Stage

The RS was decommissioned and disassembled, and the DTI was achieved with it. DTP is also archived, however, with the many products, that would not be the case.

IV. Observations

To address the challenges associated with the accessibility and comprehensibility of the notion of digital twins, authors followed a lifecycle model to present the lifecycle stages of an RS, with the DTI and DTP.During these stages, there were some observations, and challenges during the process of creating the digital twins.

A. Observations while Creating Digital Twin Prototype

First, the authors did not consider the CAD environment as a digital environment, instead considered Omniverse as a digital environment. Therefore, in this application, conversion of CAD files to .obj or .fbx and later to .usd was inevitable. The results of the conversions show that a significant amount of model data gets lost, corrupted, or duplicated during the file conversions because of the complexity of the models. Therefore to minimize the data loss, reducing the number of parts by grouping parts together as one solid part and removing the unnecessary details, modifying the names of the parts to simple ones, and deleting material properties were helpful and needed. However, some of these processes are irreversible, hence they might reduce the fidelity of the model drastically.

Further observations show that, during the modeling process, non-conventional ways, such as modeling electric cables as 3D in CAD, increased the overall fidelity of the DTP and consequently.

Lastly, there was a significant difference in data loss, due to the quality of file converters, Blender and Embedded Omniverse Converter, when the conversion of the .fbx file to a .usd file was wanted. Hence, before putting the .fbx file directly into the omniverse, putting it into the blender and modifying there, gives a better conversion of .fbx to .usd as well as gives more flexibility to manipulate the corrupted data which occurs during the conversion to .fbx file from .step file.

B. Observations while creating a real system

First, the assembly time of the real system according to the digital model, dramatically decreased, due to the provided spatial and visual data from the digital model, especially the cabling between the RPI and the breadboard.

Also the safety of components, during the assembly, also increased, which prevented common mistakes such as giving higher voltage to GPIO pins, which could result in damaging the RPI, was prevented.

C. Observation while creating a DTI

The evolvability of the DTP is important, which reduces the development time of the DTI drasticaly. For example, predefined interfaces while developing the DTP, would be really helpful to develop the DTI. In ideal case, the whole DTP could be used as a template, if the DTP is adjustable, hence evolvable.

D. Observation while operating the DTI

Even though, the behavior of the LEDs were correctly twinned, the other information about the LEDs such as spatial data was missing which resulted in incorrect ideal twinning process. For example, removing the LED from the circuit was not presented in the digital twin instance. For this representation, it was not an issue, because in the requirements of the DTI and DTP that was not a required statement, however for the industrial applications it should be considered, and can be done with the machine vision.

V. Conclusion

Digital twins as a notion, a powerful tool or rather approach to support(or even replace) the lifecycle stages of the product, which helps to reduce the needed material, energy, and time by using the information, which is acquired by the digital twin of the product.

With the help of new technologies and approaches, such as machine learning, IoT, VR/AR, Robotics, and Automation, the information about the product can be exploited even in much more depth, which results in better DTP's and DTI's. However, until today, there isn't a standardized framework, which shows how this exploitation should be handled. Such challenges, Cybersecurity, data integration, model accuracy, and model calibration are adding up to the standardization problem. Consequently, more research and development should be made for the digital twins, to implement and trade the information with the needed material, energy, and time.

Questions such as Is the DTI just the evolution of the DTP or is it another entity?, How can the quality of the DTI's as well as DTP's be qualified? Is the lifecycle of the product can be fully replaceable by the DTI and DTP? How to standardize the DTP and DTI? What is the lifecycle of the DTI and DTP? can be researched further.

References

[1] Buy a raspberry pi 4 model b – raspberry pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. (Accessed on 06/12/2023).

[2] Digital twin - unreal engine. https://www.unrealengine.com/en-US/digital-twins. (Accessed on 06/12/2023).

[3] Digital twin simulation-based software—ansys. https://www.ansys.com/products/digital-twin. (Accessed on 06/12/2023).

[4] Eclipse ditto™ • open source framework for digital twins in the iot. https://www.eclipse.org/ditto/. (Accessed on 06/12/2023).

[5] How hc-sr04 ultrasonic sensor works & how to interface it with arduino. https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/. (Accessed on 06/12/2023).

[6] itwin.js. https://www.itwinjs.org/. (Accessed on 06/12/2023).

[7] The most powerful real-time 3d creation tool - unreal engine. https://www.unrealengine.com/en-US/. (Accessed on 06/12/2023).

[8] Omniverse platform for creating and operating metaverse applications — nvidia. https://www.nvidia.com/en-us/omniverse/. (Accessed on 06/12/2023).

[9] Powering the world's digital twins — unity. https://unity.com/solutions/digital-twins. (Accessed on 06/12/2023).

[10] Ros 2 documentation — ros 2 documentation: Foxy documentation. https://docs.ros.org/en/foxy/index.html. (Accessed on 06/12/2023).

[11] Unity real-time development platform — 3d, 2d, vr & ar engine. https://unity.com/. (Accessed on 06/12/2023).

[12] Virtual twin experiences - the next generation of digital twins — dassault systèmes. https://www.3ds.com/virtual-twin. (Accessed on 06/12/2023).

[13] ISO/IEC JTC 1/SC 7. Systems and software engineering — software life cycle processes. Technical report, International Organization for Standards, 2017.

[14] ISO/IEC JTC 1/SC 7. Systems and software engineering — life cycle management — part 1: Guidelines for life cycle management. Technical report, International Organization for Standards, 11 2018.

[15] ISO/IEC JTC 1/SC 7. Systems and software engineering — system life cycle processes. Technical report, International Organization for Standards, 2023.

[16] Kazi Masudul Alam and Abdulmotaleb El Saddik. C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE Access*, 5:2050–2062, 2017.

[17] Edward Glaessgen and David Stargel. *The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles*.

[18] Michael Grieves. *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. McGraw-Hill Education Ltd, 01 2005.

[19] Michael Grieves. Product lifecycle management: the new paradigm for enterprises. *International Journal of Product Development - Int J Prod Dev*, 2, 01 2005.

[20] Michael Grieves. Origins of the digital twin concept. *Florida Institute of Technology / NASA*, 08 2016.

[21] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.

[22] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361, 2021. Digital Twin towards Smart Manufacturing and Industry 4.0.

[23] Weichao Luo, Tianliang Hu, Wendan Zhu, and Fei Tao. Digital twin modeling method for cnc machine tool. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–4, 2018.

[24] Negroponte Nicholas. *Being Digital*. Vintage, 1996.

[25] Nikolaos Nikolakis, Kosmas Alexopoulos, Evangelos Xanthakis, and George Chryssolouris. The digital twin implementation for linking the virtual representation of human-based production tasks to their physical counterpart in the factory-floor. *International Journal of Computer Integrated Manufacturing*, 32(1):1–12, 2019.

[26] Greyce N. Schroeder, Charles Steinmetz, Carlos E. Pereira, and Danubia B. Espindola. Digital twin data modeling with automationml and a communication methodology for data exchange. *IFAC-PapersOnLine*, 49(30):12–17, 2016. 4th IFAC Symposium on Telematics Applications TA 2016.

[27] Rikard Söderberg, Kristina Wärmefjord, Johan S. Carlson, and Lars Lindkvist. Toward a digital twin for real-time geometry assurance in individualized production. *CIRP Annals*, 66(1):137–140, 2017.

[28] Fei Tao, Fangyuan Sui, Ang Liu, Qinglin Qi, Meng Zhang, Boyang Song, Zirong Guo, Stephen C.-Y. Lu, and A. Y. C. Nee. Digital twin-driven product design framework. *International Journal of Production Research*, 57(12):3935–3953, 2019.