Communication and Synchronization in Digital Twins
Benefits, Requirements and Constraints

Waalboer, Daan

Award date:
2021

Department of Mathematics and Computer Science
Software Engineering Research Group
In collaboration with ALTEN Technology

Communication and Synchronization in Digital Twins

Benefits, Requirements and Constraints

Daan Waalboer
1027257

Supervisors:

prof. dr. M.G.J. van den Brand
ing. R. Thelosen

Version 1.2

Eindhoven, August 25th, 2021

Abstract

The field of Digital Twin research is young, experiences rapid development and lacks standardization. Concepts and definitions are often misrepresented or misused. Considering the importance Digital Twin systems have begun to hold across industry sectors, this work aims to provide structure in such obfuscated yet valuable concepts. Key terminology and model variants are discussed, compared and categorized. Real-world examples are provided to emphasize Digital Twin benefits. Complementary fields of research and development crucial for the industry of the future are analyzed for their cohesion with Digital Twins. As proof of concept, a Digital Twin system is implemented in parallel to test and validate the claims of provided literature.

Preface

It has been a peculiar graduation phase, given the various COVID-19 regulations. Initially, I opted for a graduation internship at a company to experience the day-to-day workflow and meet lots of new people from different backgrounds. As we all know, this worked out differently than expected.

However, after getting used to the life of a hermit and arranging several home-office upgrades, the efforts started to bear some fruit. In the end, I am content with the work that I put together, and I hope it proves of value to future researchers and other interested parties.

During the past months, a couple of people assisted me through various means. Either through discussions, brainstorm sessions, or jokes in virtual meetings, or by cooking me dinner when the day suddenly contained fewer hours than my schedule required.

Thank you Mark, Rik, and Ellen, for your various means of assistance. Despite the heavily skewed division of the workload, I think we can consider this thesis a joint effort!

Contents

List of Figures

Chapter 1

Introduction

In recent years, development in IoT industry aspects such as digital transformation, high tech systems, intelligent buildings, cities, and mobility has progressed exponentially. Production robots and processes, along with all other kinds of motorized machinery, are equipped with arrays of sensors for continuous, live data collection [51]. This collected data can, among other things, be utilized for operational analysis and optimization [46]. Optimization is required to continuously reduce operational costs and materials as well as energy consumption, potentially in operational environments subject to frequent changes [33]. In order to enable continuous analysis and improvement of complex physical systems, Digital Models can be generated representing these systems. Feeding runtime data from the physical system into these digital representations allows organizations to monitor and analyze the system's behavior in great detail, without the need to temper with physical components.

This type of runtime analysis has been discussed several times, for example by Blair et al [10]. Blair et al. explain the notion of models@run.time to be "A promising approach to managing complexity in runtime environments" over a decade ago. They define a model@run.time to be "A causally connected self-representation of the associated system that emphasized the structure, behavior, or goals of the system from a problem space perspective".

Digital Twins (DTs) are exceedingly utilized throughout modern industries to achieve this kind of runtime analysis. Digital Twins exist in various formats and offer varying levels of connectivity and synchronization between physical and digital counterparts. To prevent ambiguity, three definitions can be highlighted. First, one can generate a Digital Model of a system. A Digital Model is merely a digital representation of the state a physical system had at a specific timestamp. In some cases, merely copying parameters from a physical entity into a digital snapshot representation is sufficient to form a Digital Model. After its creation, changes in the physical system will no longer affect the Digital Model. For other cases, this digital representation needs to be updated continuously based on the input and output parameters of the physical system. Such a continuously updated collection of data related to a system can be considered a Digital Shadow. Digital Shadows extend Digital Models in the sense that they continuously follow updates as they take place in the physical system. However, in the case of Digital Shadows, changes in the digital entity do not affect the physical system [20].

Essential for this thesis is the third definition, true Digital Twins (DT's) that can also synchronize both ways. In addition to continuously updating the digital representation based on physical parameters, the physical system can be adjusted or altered to mimic changes made by the digital entity. Differences between Digital Models, Digital Shadows, and Digital Twins are, among others, discussed by Fuller et al. [20]. Figure 1.1 is based on their example, highlighting the differences between the three.

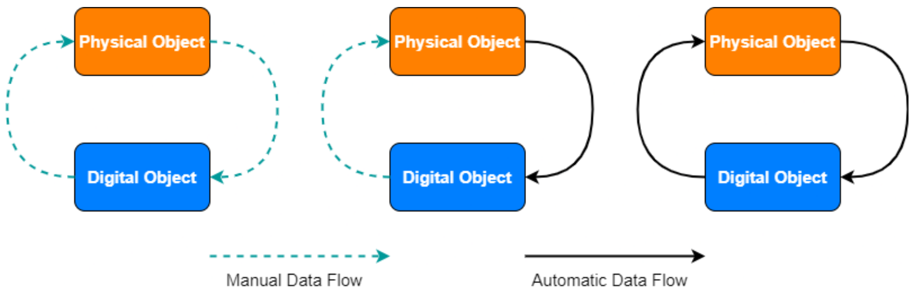


Figure 1.1: Difference in data flow for Digital Models, Digital Shadows and Digital Twins

Following this distinction by Fuller et al. and the definition by Qi et al., a Digital Twin, as referred to in this text, entails the combination of a physical asset or entity, as well as its digitized representation, which mutually communicate, promote, and co-evolve with each other through bidirectional interactions using a shared data source [46]. By this definition, in this paper we speak of Digital Twins when a digital entity continuously represents the most up-to-date version of its physical counterpart and its parameters, and enables propagating changes back into the physical entity autonomously as well. A Digital Twin may also represent an abstraction of a system, or specific parts in particular. Note that in some other works up until recently, people have referred to systems fitting this definition by calling them "Digital Shadows" [18]. Because Digital Twin research is still in active development, it is vital to highlight the differences between different types of models and connectivity. This thesis aims to deliver a straightforward, structured approach to compare and relate each type with and to others.

The value of implementing a Digital Model can range from locating bugs to analyzing and optimizing existing supply chains and factory systems, of which the latter is described by Kuhn & Wolfgang in the setting of a factory production system [34]. A Digital Shadow offers additional values, as analyzed among others by Zhang et al. concerning a hollow glass production line [62]. To take an extra step, Schluse, Priggemeyer, Atorf and Rossman discuss the implementation of Experimentable Digital Twins (EDTs) [50]. These EDTs are essentially Digital Shadows of a physical system, while also capable of accepting and processing extra input parameters in their digital state. These EDTs are used to test hypothetical scenarios in a Digital Shadow environment during runtime, but do not necessarily propagate changes back into their physical system.

In order to expand such applications to also propagate changes made in the Digital Model back into their physical counterparts, sufficient hardware support must exist before initiating optimization attempts. Using the five-dimension Digital Twin model as proposed by Tao et al., Qi et al. suggest a framework containing a range of enabling technologies for Digital Twins [54] [46]. Following the five-dimension Digital Twin model, they discuss specific technologies related to building physical systems, generating Digital Models, establishing connections and services, and data storage and processing solutions. Digital Twins require a reliable combination of all these technologies.

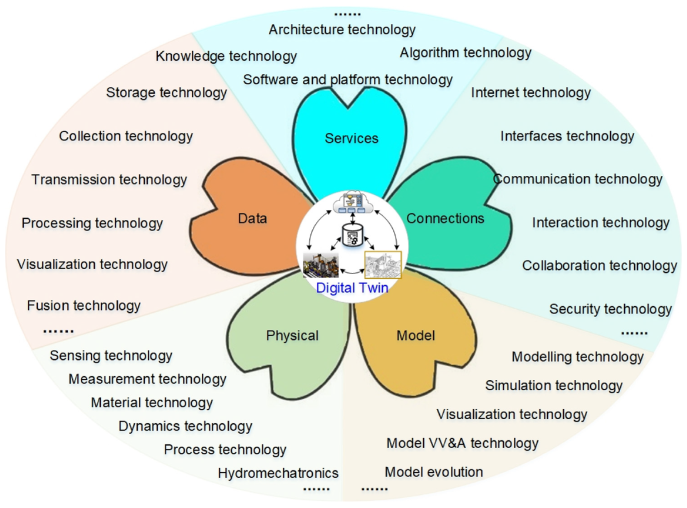


Figure 1.2: Framework of enabling technologies for Digital Twins, as provided by Qi et al. [46]

Given the available literature, The field of Digital Twins is an essential aspect of digitization in industry. Digital Twin systems enable efficient optimization of existing physical production processes and machinery. One of the main issues in the current field of Digital Twins involves the two-way synchronization between digital and physical entities. Updating variables in Digital Shadows and experimenting with additional variables in systems such as Experimentable Digital Twins have become common phenomena. However, propagating changes from virtual models back into their physical system offers its own branch of problems to overcome.

In this thesis, we embark on an extensive analysis of both the software and hardware technologies currently existing to utilize Digital Twins. A historical view on the evolution of Digital Twins and their relevant concepts and technologies is provided as supporting knowledge. Building on the status quo, the existing technologies are discussed and their corresponding benefits when building live-synchronizing Digital Twins are highlighted. Among others, a case-study is provided for which specific hardware and software requirements and constraints are analyzed and implemented. Through this thesis, an attempt is made to answer the following (fundamental) Research Questions (RQ).

- **RQ1** – Based on the available literature knowledge, why do we create Digital Twins, and what benefits do they offer?
- **RQ2** – Can scenarios be derived in which a physical entity can be influenced or optimized based on the data found in its virtual representation counterpart?
- **RQ3** – If so, what benefits does such influence offer?
- **RQ4** – What kind of data or components are required in virtual and physical systems or environments to facilitate such interactions between entities?

To establish a solid foundation of knowledge, model design engineering is discussed first in section 2.1, followed by the historical notion of models@run.time in section 2.2. Then, in sections 2.4, 2.6, 2.7 the evolution from Digital Models to Digital Shadows and Digital Twins is discussed based on the aspects of the three-dimensional and five-dimensional Digital Twin models (sections 2.3 and 2.8 respectively). Virtual prototypes and Experimentable Digital Twins are elaborated upon as well during this evolutionary discussion, in sections 2.5 and 2.9. The importance of Digital Twins for various industry sectors is discussed in section 2.10 and the capability maturity model is introduced along with a specific adaptation for Digital Twins in section 2.11. Data markets and machine economies, both important industries for Digital Twin applications of the future, are explored in section 2.12. Next, a case study is provided in 3 which aims to build and explore a basic Digital Twin system that adheres to the rules of the true Digital Twin defined in this paper.

This case study will be used to expand our understanding of the available research material and emphasize the existing capabilities and pitfalls in Digital Twins, as well as strengthen the theories established in this work.

Chapter 2

Relevant Research

In this section, the relevant material preceding and surrounding Digital Twins will be provided and discussed in detail. Model-Driven Architecture is defined first as a supporting introduction towards the concept of models@run.time. Further expanding the evolution of Digital Twins is the three-dimensional Digital Twin model, which in turn serves as a basis for the five-dimensional Digital Twin model. Woven into these concepts are a detailed introduction and comparison for Digital Models, Shadows, Twins, and several other key concepts.

2.1 MDA and MDE

The Object Management Group (OMG) [43] is a consortium founded in 1989 by several high-profile technology companies.

One of the standards OMG has documented is Model Driven Architecture (MDA). MDA refers to the OMG's standardized vision of the more general concept of Model-Driven Engineering (MDE). With the MDA, OMG defined an approach to avoid the nightmare that is legacy software maintenance. Model-Driven Architecture focuses on the separation of system functionality specification and the actual implementation for that functionality in a specific technology platform. This separation is especially relevant to IT systems, given their ever-changing environments and tooling. Using the MDA approach in system functionality specification, the same functionality can be implemented on multiple different platforms. Furthermore, integration and interoperability between applications can be supported as well. The MDA supports system upgrades and evolution regardless of changing environments and tooling throughout the system's lifetime [32].

By design, the MDA emphasizes PIMs (Platform Independent Models) over PSMs (Platform Specific Models). The general functionality specified in a PIM can be modeled on a specific (programming language) platform through a PSM, after which it can be implemented on that platform. In this sense, MDA focuses on exploiting models as abstraction tools to summarize system environment information and envision different perspectives for the system. Crucial for MDA is the bridge between platform-independent and platform-specific models. According to OMG, a PIM is a formal specification of a system's functionality and structure, which does not include technical details. A PSM is not a PIM, but it is not an implementation either. A PSM is a model specifying a system for a specific target platform, using the PIM as its template. This PSM can then be used as guidance during the implementation phase.

For further elaboration, see Figure 2.1. First, a PIM is constructed by applying high-level logic and knowledge to solve a general problem. After creating a PIM, PSMs can be created by modeling the PIM's logic towards a specific platform environment. Using these PSMs as guidelines for the implementation, the desired piece of software can be written for a specific platform. Considering the implementation follows the PSM, which should itself adhere to the PIM, the resulting software should provide the same functional behavior regardless of the platform in question.

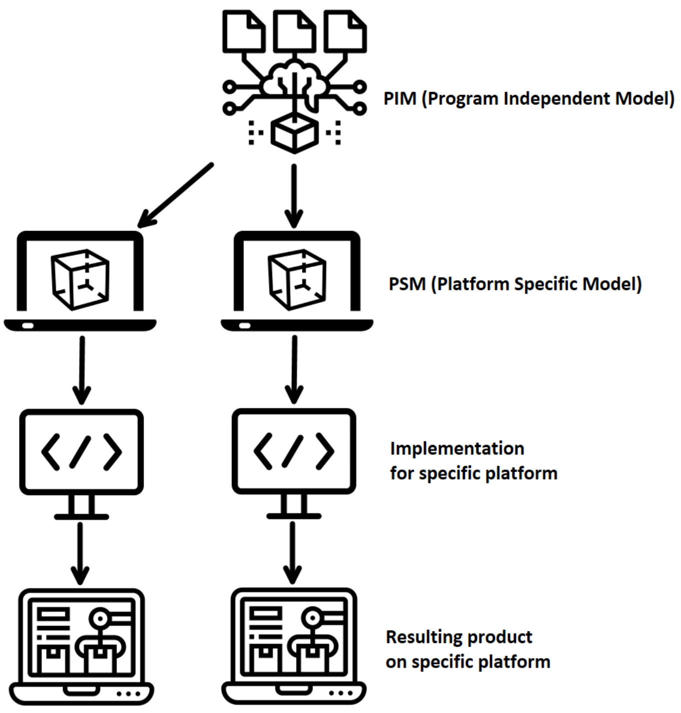


Figure 2.1: Visualizing the difference between PIMs, PSMs and model implementations for specific platforms

Such an MDE implementation is a valuable concept in Digital Twin development. Upon developing a hardware product, engineers can opt to introduce a generic set of rules (i.e., Platform Independent Model). These rules should enable flawless communication and optimization between the hardware product and its digital representation, which is modeled (PSM) and implemented for an arbitrary platform in a way that honors the ruleset of the PIM.

As Kent highlights, “a clear goal is that transformations between models should at least be partially automated, thereby reducing the burden of keeping models in step to the point that their intrinsic benefits outweigh the costs of their maintenance” [32]. The notion of this automated transformation between models in Digital Twin development is valuable, but not always feasible. An example of this infeasibility is provided in section 4.3.1. For now, consider the act of adding extra sensors or parts to a high-level hardware system design, of which implementation instances are utilized in various environments. To ensure any platform-specific models keep showing correct behavior, changes in the ruleset of the PIM should generate corresponding ruleset changes in all relevant PSMs as well. As long as these model rulesets are accurate and unambiguous, developers can alter their corresponding software implementations while maintaining adherence to the original model and hardware environment.

To briefly elaborate on the goal of partially automating transformations between models, consider the following MDE approach to developing hardware simulation prototypes as written by Nagy et al. Their work aims to both benefit the lithography company ASML as well as providing a general approach to developing Digital Models for hardware prototypes [42]. As hardware is often expensive to acquire and sometimes even dangerous to replace (i.e., in fragile or operational systems), Digital Models can prove beneficial. By creating a model representation of hardware prototypes, tests with unpredictable or even potentially harmful results can be run across a digital environment instead of damaging a physical prototype system in its test environment. An in-depth view of so-called Virtual Prototypes follows in section 2.5. An area of expertise seeking to extend the applicability of models produced by MDE approaches to the runtime environment is that of “models@run.time”.

2.2 Models@run.time

As explained above in section 2.1, a model in the context of MDE is a more abstract representation of a (software) system designed for specific purposes. The concept of models@run.time shares this view of what constitutes a model and aims to explore the benefits such models can provide at runtime.

As mentioned in Chapter 1, models@run.time lie at the heart of Digital Twin research and development. They seem to be the first clearly defined idea that concerns the self-governing of a system through a causally connected self-representation of that system from the perspective of a specific environment. According to Maes, one of the core principles of models@run.time is the notion of computational reflection. Maes defined: "computational reflection to be the behaviour exhibited by a reflective system, where a reflective system is a computational system which is about itself in a causally connected way" [38]. In other words, a core principle of models@run.time is the ability to reflect upon past behavior within the system, by the system itself. Such reflective capabilities enable changes in the runtime model to lead to corresponding changes in the reflective part of the system and the other way around. Through this reasoning, Blair et al. defined a runtime model as a causally connected self-representation of the associated system that emphasizes the structure, behaviour, or goals of the system and which can be manipulated at runtime for specific purposes [10]. Such a causally connected representation is crucial for adaptive systems for two complementary reasons. First, the model needs to provide continuous live information from its connected hardware system to facilitate accurate decisions based upon that information flow. Second, a causal connection allows developers to implement changes for the system in the Digital Model instead of in the hardware.

Future software creation in many systems will need to be long-lived in constantly changing environments and conditions. Furthermore, these changing conditions might not, or only partially, be known during development. These constant changes emphasize the critical role software models produced during development have to play at system runtime. Andersson et al. explain how models@run.time can assist in the creation of such change-driven software [6].

Blair et al. highlighted important other works and key challenges relevant for models@run.time [10]. They emphasize that similar to the way MDE aims to develop more general software engineering solutions, models@run.time are built from a problem space perspective rather than a specific solution environment. They also mention that for runtime models to be used to fix errors and provide live updates in a running system, significantly more complex mechanisms are required. A more straightforward application for runtime models is to enable a system to switch between sets of existing behavior or functionality during runtime, without processing modifications based on that behavior simultaneously.

Blair et al. go on to explain how several key dimensions are relevant for models@run.time [10]. First, there is the distinction between *structural* and *behavioral* models. Structural models represent how a system or software is built, focusing on relationships, components and connections, and general hierarchical structure. Behavioral models follow the execution flow of events and traces to provide the various possible pathways weaving through the system.

Then, models can be *procedural* or *declarative*. Where a procedural model reflects the actual structure in a specific system, a declarative model focuses on the goals set for the system. From the viewpoint of modeling the broader problem space instead of specific solutions, declarative models are preferred.

Additionally, models can be *functional* or *nonfunctional*. Most models to date are created as functional models, in the sense that they follow the functional behavior of the system they represent. Nonfunctional models can be developed to keep track of specific concepts such as a system's security or various performance metrics.

Finally, models can be *formal* or *informal*. Formal models are created by following strict computational mathematics, enabling accurate monitoring of a system's state and various parameters. A downside of formal models is their weak expression of domain-specific concepts. Informal models are created with domains abstractions kept in mind to circumvent that issue.

Various models can be required to adequately model an extensive system (of systems), combining the up- and downsides of these various model dimensional choices. Luckily, exponential advances in efficiency of modern computer technologies have provided the ability to combine, model, and analyze many different systems in the same live runtime environment. The area of Digital Twin development greatly benefits from a combination of different types of models depending on the problem at hand. Several examples will be provided later on in this thesis, using which the importance of the models@run.time concept is emphasized.

2.3 Three Dimension Digital Twin Model

Despite not yet being defined as Digital Twins, Dr. Michael Grieves first introduced the concept of an equivalence relation between digital and physical entities in 2003 at a Product Lifecycle Management course at the University of Michigan. Back then, Digital Modeling of physical systems was in its early phases because of development and data collection constraints [23]. Grieves went on to introduce the term "Digital Twin" later in 2011 [22]. In his process Grieves developed the "3-dimension Digital Twin model" as shown in Figure 2.2 [23]. The model displays a physical system or entity and a virtual representation or model, connected by two flows. The virtual model retrieves its current state through data parameters found in the physical entity, and the physical entity can receive information processed by the virtual model in return. Although simplistic, this model captures the important notions still found in Digital Twins.

Grieves explains that in the years after he introduced the three-dimensional Digital Twin model, both virtual modeling tools and physical sensor capabilities have evolved drastically. Advanced simulation capabilities were introduced in virtual modeling tools, allowing performance testing for complex systems (of systems) in real-time paired with feasible computing power requirements. Alongside these improvements in the virtual space, a wide range of sensors has been installed in nearly every piece of hardware. The large amounts of data collected by these physical sensors directly improve the accuracy and capability of virtual models.

This "3-dimension Digital Twin model" was to become the backbone of the five-dimensional Digital Twin model later proposed by Tao, Zhang & Nee as shown in Figure 2.10 later in this document [55].

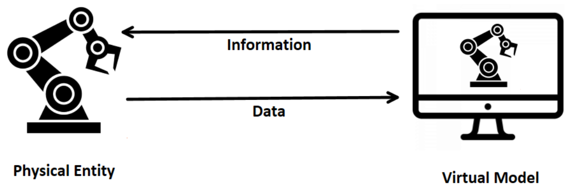


Figure 2.2: Three-dimension Digital Twin Model adapted from Grieves' proposal

2.4 Digital Models

Every Digital Twin system contains one or more Digital Models, but not every Digital Model is part of a Digital Twin system. As will be discussed in this chapter, Digital Models can consist solely of data parameters but can also be represented through extensive 3D modeling. A Digital Model is essentially no more than a digital representation of a specific physical system at a specific point in time. Such a “snapshot” can be utilized as a truthful source of information for a system or structure at that timestamp. For example, imaging tools have been used for decades to generate height maps of planet earth. These height maps can, among other things, be used to run hydrology simulations to estimate the movement of water bodies across landmass as visualized in Figure 2.3 [24].

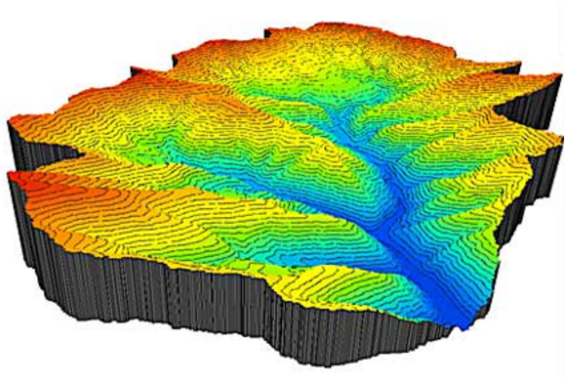


Figure 2.3: Example of a hydrology model mapping geographical altitudes [24]

Digital Models can be represented by a slight alteration of Grieves’ Three-dimensional model. Figure 2.4 displays this alteration. As the simplest subset of Digital Twins, a model is generated using existing data parameters from a physical entity. As such, they represent a specific snapshot state of a system. Logging such states in detailed models can offer benefits in various use-cases, not limited to hydrology, as shown above. For example, Boeykens, Quintero, and Neuckermans showcase 3D digital architecture models improving the design analysis of varying structures as shown in figure 2.5 [13]. This kind of digital architecture modeling goes beyond visualizing the looks of a structure remotely. By including material choices, density, pressure forces, and other structural decisions, the structural integrity of buildings can be tested for extensive wear and tear and weather resilience in a safe environment.

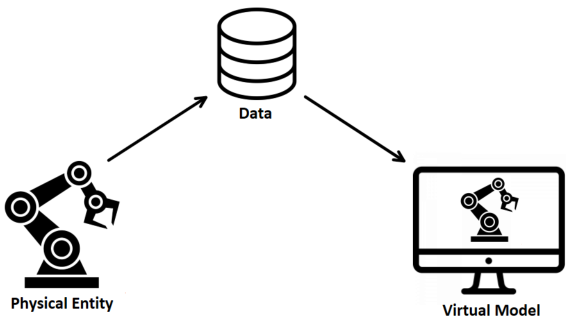


Figure 2.4: Digital Model represented using parts of the five-dimensional Digital Twin model. Each arrow represents (parameter) data flowing from one entity to another

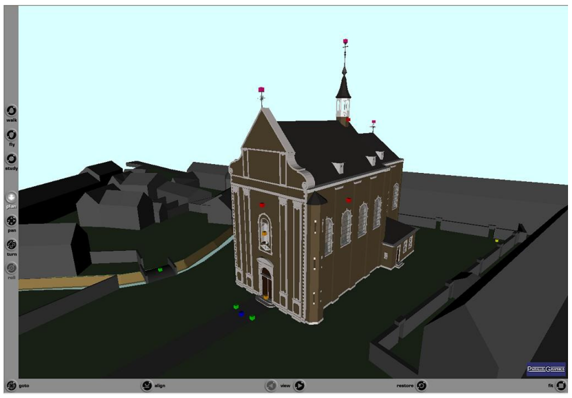


Figure 2.5: Example of Digital Models in architecture [13]

Digital Models such as the examples provided make use of both geometric modeling techniques as physical modeling techniques. The geometric model provides a virtual representation of the physical entity's shape, size, placement, and more. This representation can take form through wireframe modeling, surface modeling, or solid modeling to achieve different levels of detail. Whereas wireframe modeling highlights structural forms, surface and solid modeling provide realistic surfaces and three-dimensional integrity of components and materials [13].

A similar application of Digital Models in a more dynamic environment is found in the manufacturing and testing of passenger cars. Rauh explains how simulation and modeling of various car components are used to analyze the ride and handling properties of modern passenger cars [47]. Each separate component already has a corresponding virtual model. These models are then all joined together in coherence with the physical system environment. Combining these models into a larger structure allows car manufacturers to analyze the material responses exercised by the road on components such as tires, chassis, and suspensions [47]. Figure 2.6 displays an example of road models used in such analyses. On top of this, the effects different components have on each other can be explored as well.

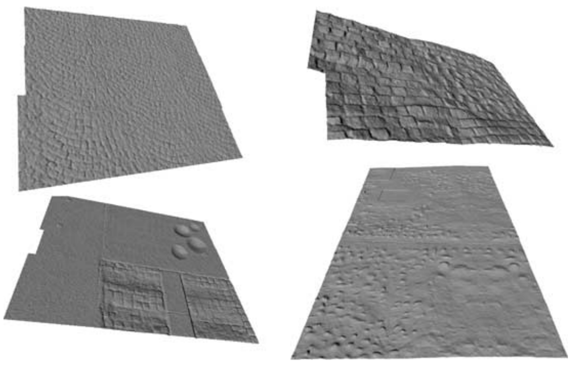


Figure 2.6: Example of road modelling explained by Rauh [47]

Technologies Enabling Digital Models

When looking at the technologies enabling Digital Models, a top priority is developing a proper understanding of the physical environment to be represented by the model. The attributes and techniques in Digital Model creation require knowledge from various fields, such as structural mechanics, materials science, and thermal physics. Before creating a virtual model, the physical world and its data should be cognized and perceived. We need to understand the power technologies, connections, transmissions, and process technologies involved when control systems drive their actuators to carry out specific actions in the physical system.

Only after developing a proper understanding of the environment and the entities to be represented, high accuracy can be achieved in model details and execution flows. Additionally, a proper understanding of an environment enables analyzing the correctness of parameters in existing models built for that environment. In order to develop an accurate virtual model of an existing physical system, measurements should be performed to gather spatial data and attribute information. For example, to construct the architecture models showcased by Boeykens, Quintero, and Neuckermans as well as the road models provided by Rauh, tools like laser measuring and image recognition can be used to gather spatial data [13] [47]. Further attribute information can be acquired by analyzing the materials and techniques used to build the physical entity. This material knowledge is required to implement realism in the physics aspect of the Digital Model.

Having acquired the parameters of the physical entity, the process of modeling can start. Modeling, in the scope of this thesis, can be defined as the process of generating a digital representation of a physical object or set of objects, which can be analyzed and managed through computer software. Using available spatial data, for example, a geometric model can be created to represent the physical appearance of the system in the virtual entity. Such a model can be developed using 3D modeling software solutions such as Unity, Maya, Blender, and many others [59][7][11]. These models need to support any possible movements from their physical counterparts, which can be incorporated by connecting programmable scripts to specific model components. These scripts require programming knowledge as well as an understanding of the hardware domain, to match digital sensor and actuator parameters to their respective physical components.

2.5 Virtual Prototypes

Besides modeling digital variants for existing physical machines and systems as discussed above, an additional use-case for Digital Models is the creation and exploitation of Virtual Prototypes. The idea of a Virtual Prototype is to generate a specific model and environment in order to explore possibilities of real-world implementations. Doing so might unearth potential issues and mishaps without posing significant financial or physical risks. An example of such a Virtual Prototype is provided by Frimpong and Li [19]. They discuss the development of a virtual model of a hydraulic shovel system. This virtual representation is modeled in a specific environment to see whether the machine would perform as expected before any physical assets are fabricated and mobilized to the project site. Specific goals are to explore the collaboration of various interconnected parts and components within the machine and the way it interacts with materials in the environment [19]. Figure 2.7 illustrates the Virtual Prototype as discussed by Frimpong and Li.

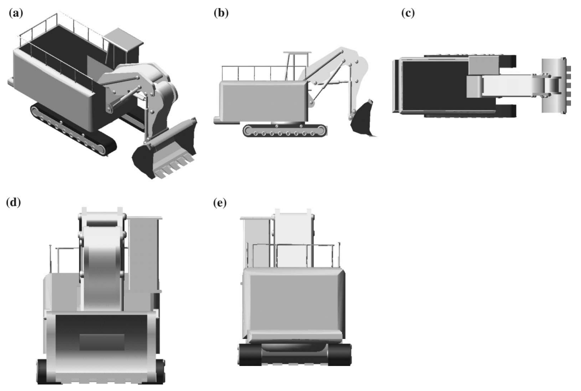


Figure 2.7: Virtual Prototype for hydraulic shovel machine [19]

The digital architecture models showcased by Boeykens et al. can also be utilized as Virtual Prototypes [13]. Instead of creating models for existing structures to perform integrity tests, entire models can be drawn for development plans. Doing so enables problem detection before physical construction begins, reducing the physical and financial risks that come with it. For example, material choices can be tested for strength and resilience in specific weather conditions such as thick layers of snow.

Similarly, the modeling of various car parts and combinations thereof described by Rauh is not only used to test existing products but also utilized in a prototyping environment to perform material testing before producing actual prototypes [47]. Depending on the types of materials and tests, this approach reduces financial and material cost compared to immediate physical prototype testing.

Note the distinction between the different applications of a Digital Model. Firstly in the form of a snapshot for an existing environment or system, Digital Models offer a resilient test environment that can be used for efficiency improvements and other optimizations or to locate issues without stressing the actual system during runtime. Secondly, Digital Models applied in the form of Virtual Prototypes allow for thorough reviews of materials and interactions in an often relatively cheap environment prior to time- and resource-heavy construction of machinery or buildings.

Technologies Enabling Virtual Prototypes

As a Virtual Prototype is essentially a Digital Model created prior to the construction of a physical system, the technologies required for creating Virtual Prototypes partially overlap with those of Digital Models. Where Digital Models can be physically measured and perceived and then be reconstructed in the virtual world, Virtual Prototypes, by definition, pre-exist before the creation of their physical counterparts. Because of this, a prototype is generated using theoretical parameter data by the rule of its creator. To this purpose, a model developer can look at the environment for which the prototype is created to define realistic parameters. Alternatively, they can use historical data of comparable systems as a starting point for their development. The creation of the actual model in a Virtual Prototype and its runtime behavior occur through similar modeling software environments as discussed for Digital Models built after an existing physical counterpart in section 2.4.

2.6 Digital Shadows

Extending the properties of Digital Models, the next step in the evolution towards fully interconnected Digital Twins can be defined as "Digital Shadows". Digital Shadows are essentially similar to a Digital Model sharing the same basic capabilities, with the addition of continuously mirroring changes in their physical entity as they occur during runtime. As such, they always represent a real-time representation of the physical system, which is continuously communicating its input and output parameters towards the digital entity. This upgrade improves relevance for Digital Shadows in many industries over that of regular, static models. Numerous industry supply chains and machine systems are subject to frequent change and a constant flow of information parameters. Being able to mirror these changes into the virtual model allows external parties to monitor processes in detail without requiring physical access to the system. Continuously monitoring parameters and actions in structures or processes involving (often heavy) machinery enables spotting or predicting potential failures before they can occur. An example hereof is provided by Ladj et al., using a Digital Shadow for tool failure prediction in an industry setting [35]. Maintenance and repair jobs can be scheduled accordingly, and downtime can be minimized. Also, the availability of a Digital Shadow allows third parties to receive access to process data without the paperwork and potential security risks involved with providing physical access [9]. Bauernhansl et al. reason that, on top of this risk-free access,

Digital Shadows can enable automated cost analysis and fee processing in data purchases as well [9]. For example, visualization tools might offer data streams for a price that correlates with the amount of available data. Using Digital Shadows, interested third parties could purchase such data for accurate prices without human intervention.

With the advances towards Industry 4.0 across industries, system data availability plays an ever-increasing role. The sheer amounts of available data are vastly increasing, and hardware systems are becoming more complex with every innovation.

Digital Shadows represent the next step in the process towards a true Digital Twin. As such, they offer more inter-connectivity than a Digital Model and can be conceptualized using the same schematic representation as used before. As visualized in figure 2.8, the information parameters from a physical entity are not only used to generate as well as update its virtual representation. They are also made available for external services such as optimization and prediction algorithms. Note that data in this overview flows one-way only. The virtual model and visualization services both have continuous access to live environment data transferred from the physical system, but neither receiving entity can alter the processes of the physical entity.

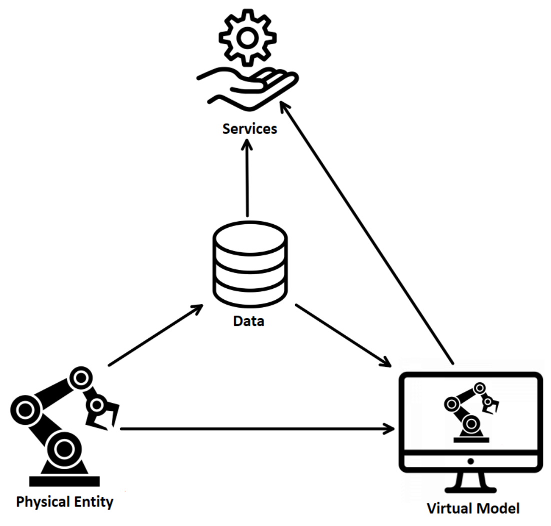


Figure 2.8: Digital Shadow represented using parts of the five-dimensional Digital Twin model. Each arrow represents (parameter) data flowing from one entity to another

Technologies Enabling Digital Shadows

In its essence, a Digital Shadow can be seen as a Digital Model which is continuously fed with live parameters from the physical system it represents. As such, part of it can be developed using the same programmable logic and 3D modeling tools as utilized for the creation of Digital Models explained in section 2.4. The core functionality for Digital Shadows is to continuously gather, combine, summarize, and provide information to various stakeholders towards various end goals. As such, to enable a shadow's functionalities, communication networks and protocols need to be available to securely transfer said information from the system to the Digital Shadow. Because of the continuous information expansion, a shadow requires a lot of extra storage capacity compared to a regular model. From a hardware aspect, the sensors and actuators installed in a modeled system need to be capable of sharing their parameter data on a much more frequent basis than they would in a Digital Model. As such, applying the capabilities of a Digital Shadow for a production environment is best decided upon in an early construction phase to reduce potential compatibility issues.

According to Qi et al., The technologies required for a Digital Shadow to operate need to fulfill the following iterative sequence. The first item is *data collection*. Sensor and actuator data needs to be generated and output by various physical hardware components. Second is *data transmission*. All sensory data needs to be relayed from the physical system to a digital environment. In this digital environment, *data storage* is required. After securing the raw data, an efficient Digital Shadow should be able to perform one or more rounds of *data processing*, in order to sanitize and prune the (often vast) available datasets from each hardware component depending on the desired analysis. Finally, in the *data fusion* step, information gathered from various different hardware components should be aligned and merged in order to generate a system-wide information bank [46]. Qi et al. provide figure 2.9 to highlight these steps for a Digital Twin. However, as the data flow from hardware system to Digital Model is similar in a Digital Shadow as it is in a Digital Twin, the same visual can be applied here for Digital Shadows. The image suggests a variety of software tools to be utilized for each step in the information flow sequence.

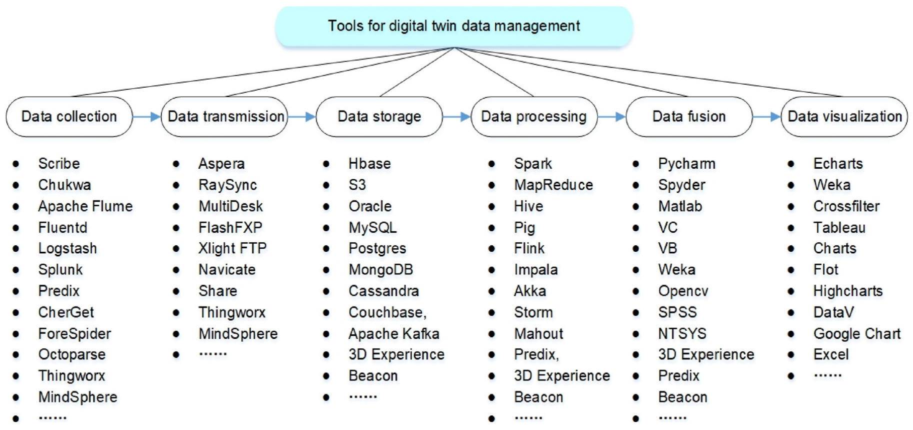


Figure 2.9: The handling of information from source to application, similar between models and twins [46]

Note that, for multiple (competitive) businesses and other stakeholders requiring access to the same data on a global scale, trust can be a legitimate concern [15]. Too often, multiple corporate entities install and monitor comparable or even equivalent hardware combinations simply because they cannot guarantee the validity of data provided by foreign entities. One of the solutions for this issue that was invented only recently is applying a sophisticated DLT (Distributed Ledger Technology) to implement a distributed (decentral) trustless data marketplace [28]. When none of the actors involved can modify, add or remove data without others noticing, the need for trust is eliminated altogether. Considering the importance of trustless data in a world evolving towards Industry 4.0 and mass application of IoT devices, this subject is further evaluated later on in section 2.12.

2.7 Digital Twins

The benefits of the live state-synchronization of a Digital Shadow over the static generation of a Digital Model are significant in most industries. However, the notion of a Digital Shadow can be improved even further. In essence, Digital Shadows provide valuable information for analysis and optimization in business processes and production lines. Considering the sheer amount of data that is captured nowadays, optimization analysis can no longer be reasonably performed by human operators. This growth in data size can be partially solved by exploiting pre-processing algorithms and other automated services to summarize the available data and highlight potential points of interest to the operator. Riesener et al. discuss such a pre-processing approach to work with immense data growth [48]. Despite saving time and other resources, this approach still depends on human intervention. After spotting potential issues or optimization possibilities, the operator needs to suggest or implement changes to physically implement in the system at hand. After doing so, the monitoring process re-iterates from the start.

As data keeps increasing in amount and size, there is a need for more permanent solutions. One such solution might be found in true Digital Twins. The ultimate envisioning of a true Digital Twin is to take an accurate Digital Shadow for a system environment and expand it by including the ability for the virtual entity to back-propagate changes to its physical counterpart. Such back-propagation enables fully autonomous operation in the Digital Twin system without human intervention, even for optimization processes. To elaborate, the twin ultimately generates, stores, processes, and analyzes its own data points. Using these data points, it suggests and executes hardware parameter optimizations by itself, for itself. With this expansion, we arrive at the Five-Dimensional Digital Twin model detailed by Tao, Zhang & Nee as shown in Figure 2.10 [55].

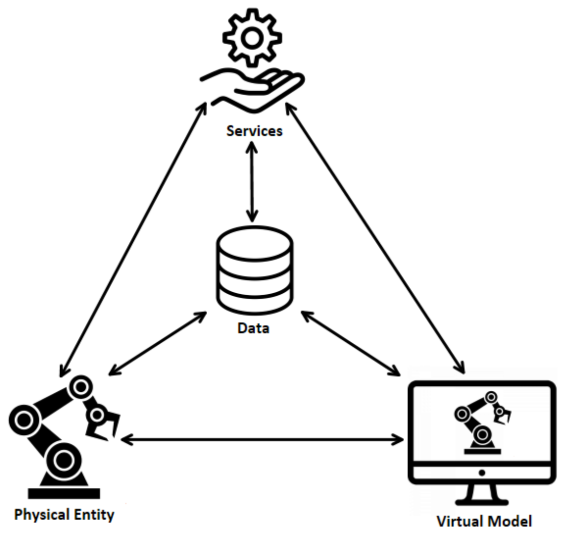


Figure 2.10: Five-Dimensional Digital Twin Model adaptation. Inspired by the adaptation of Steindl, Stagl, Kasper, Kastner & Hofmann [53] from Tao, Zhang & Nee [55]. Each arrow represents (parameter) data flowing from one entity to another

The ability for a Digital Twin system to channel information both ways between the physical and virtual entity and their related services is invaluable. The possibility to completely exclude the necessity of human intervention in an autonomous production facility unlocks unprecedented changes in the construction and planning of production facilities and factories. Removing human presence from any corporate environment drastically reduces the cost and effort required to provide a safe and motivating work environment. Facilities can be shrunk down by removing human pathways towards terminals and actuators that can now be monitored and controlled remotely and autonomously. Some real-world examples of Digital Twin usage are provided in section 2.10.

Technologies Enabling Digital Twins

For the sake of consistency, note that the technologies enabling Digital Shadows can be directly applied towards true Digital Twins as well. Collected hardware sensor data is still transmitted to some form of storage, and data processing and fusion retain their necessity as industrial datasets experience exponential expansion.

Of course, in the case of true Digital Twins, the system needs to make elaborate decisions based on the data gathered through the Digital Shadow process. Among others, a Digital Twin should detect potential optimizations in individual components and the system as a whole. For example, altering the operational speed of specific motors might improve production capacity or lower energy consumption without compromising performance. These kinds of decisions require a collaboration with the complementary research fields of Machine Learning (ML) and Artificial Intelligence (AI), which have been expanding exponentially in recent years. For Digital Twins to truly offer continuous autonomous functionality, vast datasets need to be processed, ordered, and analyzed using state-of-the-art Machine Learning models and prediction algorithms to handle crucial decisions [31]. In agreement with this statement, Min et al. proposed a framework combining Machine Learning with Digital Twin development [40]. Additionally, An et al. discuss the need for Machine Learning in Digital Twin creation and maintenance in the 3D bioprinting industry.

They emphasize how Machine Learning models outperform human computations in applications with many variable input and output parameters in rapidly changing environments [5]. Both these examples and others are detailed in section 2.10.

2.8 Five Dimension Digital Twin Model

Having analyzed the construction of the five-dimensional Digital Twin model by combining preceding and complementary technologies and concepts, we arrived at the model visualized in figure 2.10 introduced by Qi et al [46]. To concretize this model, the following explanations detail the unique aspects of and collaborations between the individual model parts as defined, gathered and used in this thesis.

2.8.1 Physical Entities

In this five-dimensional Digital Twin model, physical entities represent real-life objects such as machines, production lines, or even Systems of Systems (SoS) [12]. These entities are fitted with a variety of sensors capturing live data streams and also contain one or more actuators enabling relevant functionality. These entities, as well as the processes they take part in, can often benefit from continuous alteration and optimization of input parameters and services.

2.8.2 Virtual Entities or Model

Virtual entities in the five-dimensional Digital Twin model are detailed virtual representations of one or more physical entities. They are not merely a snapshot of a specific state but offer continuous updates based on the input and output parameters found in the physical entity. Virtual entities can be created using a combination of 3D modeling tools and behavior modeling software. An extensive list of examples for such tools is provided by Qi et al. in chapter 5 of their contribution [46].

2.8.3 Data

To facilitate the mirroring behavior required between the physical and virtual entity in a Digital Twin system, they need to tap from and store data in the same pool of information. The data component stores both input and output parameters for each entity and makes everything available for external services as well. As such, the data component should be compatible with multi-dimensional and multi-source data, both static (i.e., fixed attributes) and dynamic (i.e., variable parameters) in nature.

2.8.4 Services

To maximize benefit from a Digital Twin system, all data should be processed and analyzed to suggest optimizations in input parameters and component attributes such as hardware choices. The external services used for analysis and optimizations exist in parallel to the physical and virtual entity in the Digital Twin system and require access to the same shared data source for optimal benefit. An example of such services are administrator dashboards providing visualization and simulation capabilities to project management.

2.8.5 Connections

Just like in its simpler three-dimensional predecessor, all components in the five-dimensional Digital Twin model are interconnected. These connections represent all the data flows taking place continuously in a live Digital Twin system. Not just between the physical and virtual entities but also to and from their shared data component and external services. Everything is interconnected and shares the same live environment to optimize all operations.

2.9 Experimentable Digital Twins

A slight variation resembling aspects of both Digital Shadows and Digital Twins is the concept of *Experimentable Digital Twins*. Fundamentally, Experimentable Digital Twins match the functionality of a Digital Shadow. Recall that the Digital Shadow's core purpose is continuous live system representation and data analysis to suggest optimizations based on real data flows.

Consider the potential benefits of testing varying input parameters without pausing the system or posing a risk for operational processes. To this purpose, Schluse, Atorf & Rossman first introduced the notion of EDT's [49]. They discussed using simulators to make Digital Twins experimentable to further improve the level of optimization testing without impacting the physical environment.

For example, consider a complex physical system (i.e., many different sensors and actuators) paired to an accurate Digital Shadow. Depending on the expected or perceived system behavior, an analyst or developer might want to explore the system's responses beyond the real-time data flow. For example, would a variation in environmental temperature cause different results if all other parameters stay true to their continuously monitored values?

In essence, an EDT offers the live synchronization capabilities of a Digital Shadow, combined with the experimental freedom of a Digital Model. As such, it offers simulation environments that continuously maintain their accuracy and similarity to the real world. Despite the name, Experimentable Digital Twins are not true Digital Twins. They might be *derived* from an existing Digital Twin (or shadow) system, of course. As soon as custom parameters enter the simulation, we no longer have a fully synchronized Digital Twin but an accurate test or simulation environment instead. Parameter optimization suggestions can, after simulation tests, either be implemented manually (as is the case with Digital Shadows) or automatically (as in a true Digital Twin). These possible variations confirm the middle-ground status of the EDT. Figure 2.11 relates EDT's to other Digital Twin variants. Schluse, Priggemeyer, Atorf & Rossman discuss further examples and benefits enabled by Experimentable Digital Twins [50].

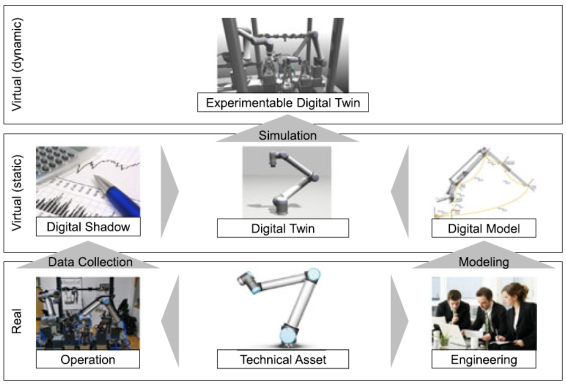


Figure 2.11: Hierarchical overview between physical and virtual entity, Digital Shadows, twins and *Experimentable Digital Twins* [50]

2.10 Why Digital Twins?

By now, we have introduced a sizeable amount of information regarding the differences between various types of Digital Models, twins, and variants thereof. To improve understanding of the general concept and the real-world benefits the research field has to offer, take a step back to view the bigger picture. During the recent past, massive advances in Information and Communication technologies have resulted in the creation and development of intelligent manufacturing paradigms. For intelligent manufacturing to benefit any field, Digital Modeling, simulation, and experimental verification can be utilized to improve efficiency in monitoring, optimizing, and controlling industry systems. At this point, virtual reality technologies based on CPS (Cyber-Physical Systems) and the Internet of Things are being combined in all sorts of systems building towards Industry 4.0, as explained in-depth by Brettel et al. and WollSchlaeger, Sauter & Jasperneite [14][60].

According to Ladj et al. and Uhlemann, Lehmann & Steinhilper, Digital Twin technology is a vital aspect of CPS to facilitate real-time analysis of production system behavior [35][57]. Analyzing, predicting, and optimizing such behavior is a critical approach to accelerate the evolution of industrial systems in general [36].

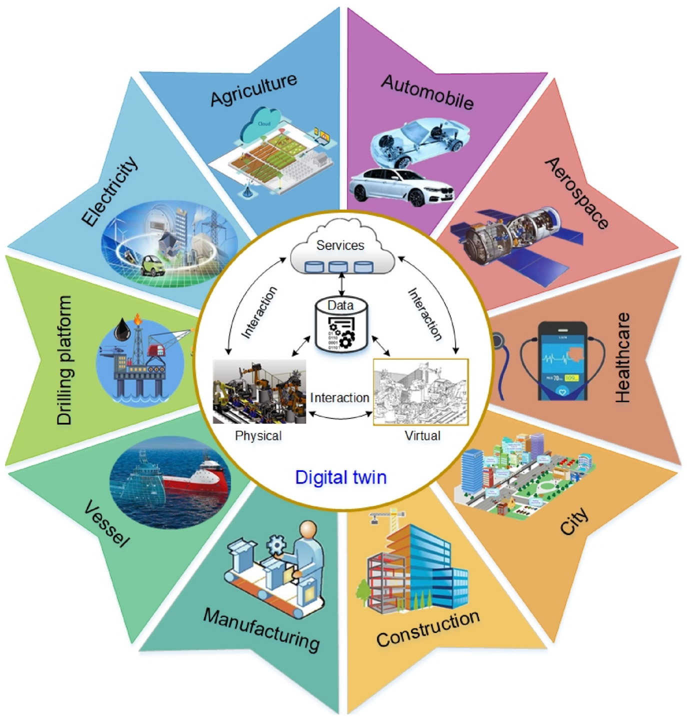


Figure 2.12: Various application fields for Digital Twins [46]

Figure 2.12 as provided by Qi et al. provides an insight into the various fields and disciplines that can benefit from Digital Twin utilization or have been doing so already. In addition to the examples already provided, some more will be discussed in this section. They serve to emphasize the relevance of Digital Twins in modern industry and its future.

To start with, Min et al. propose a framework for constructing a Digital Twin in a petrochemical industry environment [40]. They discuss the optimization scenarios in the petrochemical space that can be realized using Digital Twins. However, they also note that the academic research field of Digital Twins and their development and application is still in an exploratory stage. Often, the data in petrochemical product life cycles are isolated and fragmented and, as such, offer few possibilities for production control optimization. Given the volatile nature of final-product demand in the petrochemical industry, design, manufacturing, and service all require efficient and adaptive processes. Further emphasized by explaining how enterprises become more successful when able to provide quality products for low costs, the application of Digital Twins for production control optimization offers numerous benefits in this field.

Given the exploratory state of Digital Twins in industry, Min et al. propose a specific Digital Twin framework to facilitate petrochemical production control optimization. Their proposal combines the industrial Internet of Things with Machine Learning models, offering an environment-specific solution. They also discuss the importance of (time) synchronization between physical and digital entities in a Digital Twin system, given the demand for immediacy in this environment.

In an entirely different industry field, An et al. discuss combining big data and Digital Twins with Machine Learning to further evolve towards the future of 3D bioprinting [5]. This evolution will ultimately provide an optimized balance between virtual and physical (prototype) testing, reducing current resource consumption in bioprinting processes as well as enabling and accelerating other improvements and innovations. The framework discussed by An et al. aims to not only accurately generate replicas of biological tissues but also make them "biologically functional". As such, organs are to be replicated into a digital environment to be rendered in test environments. Such environments require fewer (financial) resources than physical prototypes. Following our definition in section 2.9, this approach revolves around creating *Experimentable* Digital Twins. Physical organs are to be replicated and then exposed to environmental modifications to study the relevant consequences. By optimizing the models through this virtual testing process, the amount of physically bioprinted organs can be significantly reduced.

Additionally, Ibrahim et al. mention the importance of Digital Twins in the future of lifetime assessment in LED (Light Emitting Diode) products and systems. Data gathered within such a Digital Twin system can enable calibration services to optimize LED behavior remotely, as well as facilitate problem prediction and detection [25].

Alexopoulos et al. propose a more generalized framework to utilize Digital Twin systems for training Machine Learning models in factory settings [2]. They explain how automated machinery such as robotic arms is often tasked with image recognition of items moving past it on conveyor belts. The ML models used for these image recognition tasks require vast amounts of accurate training data to ensure optimal efficiency and prevent erroneous image classifications (i.e., inaccurate readings). With the global movement towards Industry 4.0, materials transported on such conveyor belts can often vary in shape and structure. As such, there is little time to accommodate the large training data sets required for proper image recognition. Alexopoulos et al. apply Digital Models representing the conveyed materials to generate large amounts of training images from all angles, such that the ML model can be trained using those varying Digital Model image renderings. The application of their framework resulted in a robotic arm that could successfully recognize the orientation of an object passed underneath it without ever having "seen" such an object in the real-world environment. Such an example object model and its image renderings are provided in figure 2.13. This general framework can be exploited in any environment containing automated decision-based machinery with often-changing input parameters.

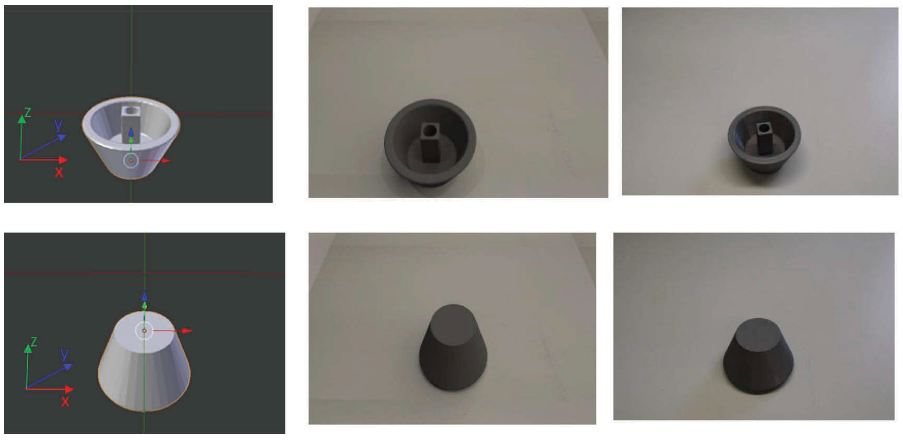


Figure 2.13: Digital Model for a simple object to be recognized by automated machinery in both form and orientation [2]

Note that, given the structured definitions and capability analysis in this thesis, Alexopoulos et al. actually used a digital *model* for their factory Machine Learning framework instead of the digital *twin* in their terminology [2]. As "Digital Twin" is often used as an umbrella term for all kinds of Digital Twin variations and subsets, the need for standardized definitions is emphasized.

Many other research and production projects exploring the possibilities of the young field of Digital Twins are currently being developed. For example, the European Union awarded 1.4 million Euros to Digital Twin software provider Akselos [1] in 2018 [21]. Using this grant, Akselos conducted the research required for piloting the Godess project (Global Optimal Design of Support Structures). By exposing materials to extensive constructional and environmental factors in their predictive Digital Twin technology, Akselos was able to reduce the material and financial costs for building steel foundations for offshore wind jackets by up to 30%.

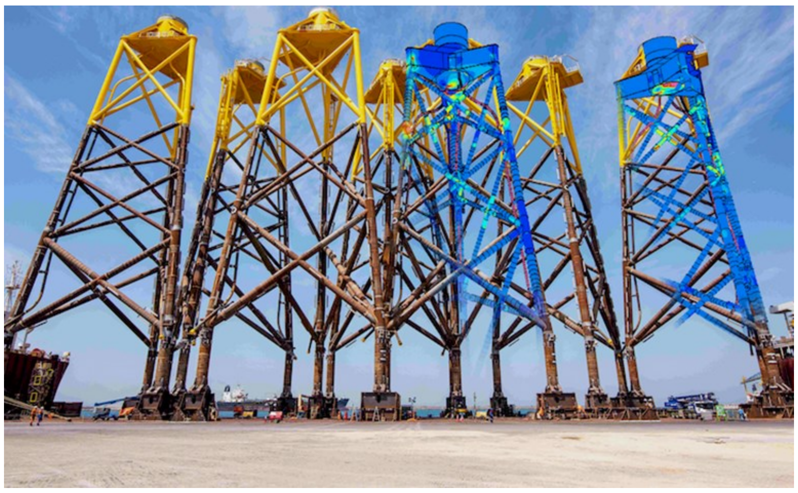


Figure 2.14: An impression of the steel structures for which the constructional physics and materials were tested using Akselos' Digital Twin software [21]

Another exciting example of Digital Twin application is called "Fields of the World". Fields of the World is an open-access dataset containing "Digital Twins" of over 3500 agricultural fields spanning an area of over 50.000 hectares. This effort has been provided by the company Agrimetrics [17] and supported by Microsoft's AI for Earth initiative [39], both of which strive towards environmental innovation and preservation. Despite growing amounts of available sensor data in various systems and entities, this data is often separated, and interoperability between systems is limited at best. "Fields of the World" aims to lift this limitation by offering a Digital Twin system constructed from a combination of all data points in the Agrimetrics Data Marketplace. Using this grand system, researchers and innovators can analyze whatever data required from all different sources in one combined, synchronized entity.

In recent years, most industries are starting to use Digital Twin-like systems for prototyping, testing, and monitoring their assets throughout entire product life cycles. There are ample examples where the availability of continuously synchronized digital representations of systems provides benefits to industry operators. Several examples have been discussed, but the list of real-world Digital Twin applications is expanding rapidly.

2.11 Capability Maturity Model

The field of Digital Twins and numerous variations thereof is young and still in its early development stages. As such, this thesis attempts to define, analyze and compare all crucial aspects. In 1991 the first version of a framework known as the Capability Maturity Model (CMM) was created to assess the maturity of software development within a company or organization [44]. Later this CMM also proved helpful to define the evolutionary status of various other frameworks and newly developed technologies. This CMM can also be utilized to evaluate the existential phase for any specific Digital Twin implementation. First, the details of the CMM will be introduced, after which an augmented version for Digital Twins will be provided.

As the CMM aims to assess the state of maturity for software (organizations), a general definition is required to separate *maturity* from *immaturity*. Software development is considered *immature* by the CMM when there is no clear process specified, and the organization solves issues on a reactionary basis (i.e., when they arise). Such an immature process often results in compromised deadlines, functionality, or quality caused by a lack of proper contingency planning. A *mature* software process organization, on the other hand, is distinguished by a proper ability to plan both regular development and resolving issues in parallel without compromising either one. This ability results in plans being realistic and goal-oriented and development teams managing to carry out the required work within the relevant constraints. Carefully defined development processes are not only adhered to but also improved when possible. The inclusion of infrastructure and documentation standards with corresponding documentation enables successful procedures to be utilized even if their creators are no longer present.

The CMM defines five levels of maturity to categorize organizational processes.

- Maturity *level 1* (or "*Initial*") represents a lack of planning and structure in the organizational development process. Issues and crises will be dealt with ad hoc, compromising deadlines and quality, and crucial information is seldom documented.
- An organizational process of maturity level *level 2* ("*Repeatable*") contains structures procedures that are created and updated based on prior experience in similar projects. Such a level 2 process benefits from proper planning and repetition or avoidance of previous successes and failures.
- Upgrading the organizational process to *level 3* ("*Defined*") is achieved by creating organization-wide rule-sets ensuring all projects follow the same structural development approach. This approach is continuously documented and improved based on its results in individual projects.
- Maturity *level 4* ("*Managed*") is unlocked by comparing project processes between all various projects and setting bounds within which new projects are expected to be executed. Known differences between projects may result in altered predictions than solely based on the available comparison data. Exceeding the specified boundaries might require executive action or even updated boundaries.

- Last in line is *level 5* ("Optimized"). This level specifies that an entire organization continuously improves its development process. Defects and other setbacks can be predicted and prevented based on available data from previous projects, whereas knowledge of unique new issues will be propagated into other projects. Additionally, this final level of maturity emphasizes waste reduction, which is less vital for processes in the lower levels.

The maturity levels are visualized in figure 2.15 as provided by the "Capability Maturity Model Integration"- product team of the Software Engineering Institute [56].

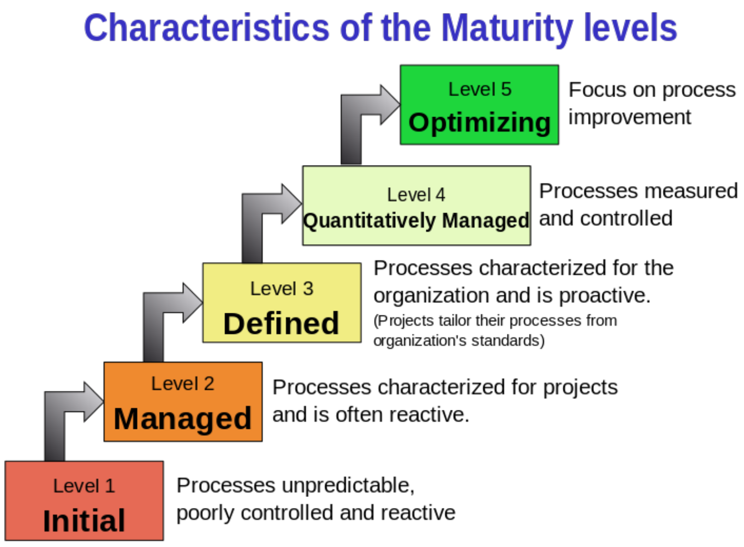

Characteristics of the Maturity levels
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

Figure 2.15: The five maturity levels defined within the Capability Maturity Model

Note that the original CMM further defines detailed rules and guidelines to examine the level of maturity for an organization's software development processes. Each level is accompanied by a list of structure- and planning-related elements that may or may not be present in an organizational process to be considered said level. Considering this paper is about Digital Twin development rather than general software development, these details are outside the scope of this research. Instead, our focus shifts towards a much younger variant of the CMM explicitly tailored to examine the level of maturity of Digital Twin systems. At the time of writing, the definitions for this "CMM for Digital Twins" concept are being prepared for academic review by Barosan, Cleophas & Brand [8]. This CMM adaptation is highly relevant to the young and volatile field of Digital Twin research. Considering its near-completed status and relevance to this thesis, an in-depth explanation of the Digital Twin-specific CMM is provided. Note that slight deviations might occur in the final publicized model, in which case the model's publication document overrules any and all conflicting information.

Resembling the original CMM, the maturity level for Digital Twins is defined using several levels, each having specific hard requirements for a model to be considered to be that level. For a system's level to increase, all upgraded requirements for the next level need to be satisfied while maintaining the requirements for the previous levels.

Level 0

Opposed to the five levels in the original CMM, the CMM for Digital Twins contains an extra level 0. This addition serves to form a basis for expanding and building the capabilities required for the other levels. Level 0 represents a legacy system with machinery attributes such as hydraulics or functional dynamics. The digital side of a level 0 system can be a 3D geometry model containing physical properties, for example. The only form of data analysis such a level 0 system allows for is collecting local data, on site, from the physical or virtual entity (PE or VE, respectively) separately. This data could then be stored for future historical access but is not used on either side of the system.

Level 1

Some minor upgrades can move a system to level 1. Most importantly, the physical entity needs to have fixed sensor devices collecting real-time states of subsystems. These sensors cannot yet be read or monitored remotely. Actuators are still fixed and offer no variable input parameters. This basic, static sensory data is also shared with the virtual entity and digital data space, where it can be used for simple (local) services. As such, the physical and virtual entities now share a fused or linked data space. External monitoring services can analyze the data from this storage space, but only in the form of historical access.

Referring back to section 2.4, the description of *Digital Models* ranks them in either level 0 or level 1, depending on the environment and capabilities. A *Virtual Prototype* remains level 0, while a model-snapshot for an existing system often passes the threshold requirements for level 1.

Level 2

In level 2, the sensory equipment of the physical entity is still fixed in place but can now be read or monitored during runtime and remotely. The virtual entity automatically validates data against the available design models. As such, the shared data space connecting the data from the physical entity to the virtual entity also validates the physical data against its design models. The data space also checks whether physical and virtual entities display consistent and synchronized behavior. External services for monitoring or, when required, manually synchronizing either entity are now enabled.

Note that these capabilities resemble those attributed to the definition of *Digital Shadows* in section 2.6.

Level 3

Maintaining and expanding upon the attributes of the previous levels, a Digital Twin system is considered to have a maturity level 3 when it includes the ability to run simulations to test services based on the virtual entity's data. These test services enable an iterative optimization process through virtual simulation environments.

The inclusion of such a simulation environment resembles the conceptual upgrade from a *Digital Shadow* to an *experimental Digital Twin* (section 2.9).

Level 4

The upgrade to level 4 is an intermediate step towards full-fledged Digital Twin maturity. The physical entity now has adaptable sensors, fixed (parameterized) actuators, and an iterative optimization process similar to that of the virtual entity. All data is fused, automatically analyzed, and the system can be manually adapted or optimized through human intervention based on this analysis. This is the first maturity level where all data can flow both ways between physical and virtual entities, their fused data space, and other services.

Level 5

The final maturity level includes a third iterative optimization loop, directly between the physical and virtual entities. The physical entity also has adaptable sensors, adaptable monitoring, and adaptable (programmed) actuators (i.e., it is dynamic). All data is fused, automatically analyzed, and the system can now be automatically adapted or optimized without human intervention.

Such an autonomous feedback loop and iterative optimization process matches the developed definitions for true Digital Twins. Ultimately, such a system operates and improves on its own within the scope of the installation environment. Of course, the extend of the actuator adaptations that can be autonomously implemented is highly dependent on the specific environment and hardware capabilities and redundancies.

As visible in figure 2.16 representing a level 5 Digital Twin system in the CMM for Digital Twins, there is an unmistakable resemblance between this maturity level and the five-dimensional Digital Twin model visualized in 2.10. Both models reinforce their common definition and capabilities for true Digital Twins.

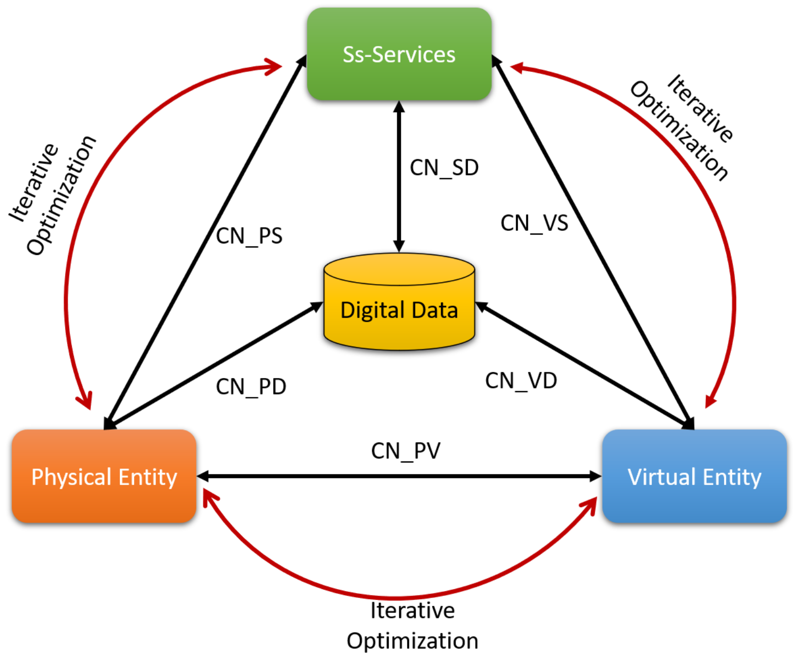


Figure 2.16: Level 5 of the capability maturity model definition for a true Digital Twin, as defined by Barosan, Cleophas & Brand [8]

2.12 Data markets, M2M communication and DLT

Alongside and parallel to the research and development of Digital Twins, several other notions are vital to the global advancements toward Industry 4.0. Examples of such notions are data marketplaces (such as “Fields of the World” [17]) and the Internet of Things. Similar to Digital Twin systems, communication and synchronization of various parameters are cornerstones to these notions.

For data marketplaces to process, analyze and distribute the ever-growing influx of sensory parameters, automation is of utmost importance. Considering the inherent value that often accompanies data from expensive corporate sources, providing this data in an open-source model similar to the “Fields of the World” project is not feasible. Combining this value with the fact that sensory parameters are often time-sensitive and potentially worthless when delayed, data must be brokered between owners continuously while it is created and processed.

This goal can only be reached through Machine-to-Machine (M2M) communications. These communications are another step in the “machine economy” that Industry 4.0 brings us. In such an M2M marketplace, self-sufficient systems can autonomously process transactions for arbitrary data. Such transactions can take place to share and store data for future analysis or live competitor comparison, for example. Additionally, when some factory system detects it is running out of manufacturing materials, it could place financial transactions to expand inventory just as easily.

Smart production and pricing imposed on data and products could control and update individual product prices based on entire chains of supply and demand with negligible delay. Such processes serve to optimize profits and efficiency, as well as reduce resource consumption [37] [28].

Such M2M communication is already implemented in real-life applications. As Stephen Mellor said from his positions of both the CTO of the Industrial Internet Consortium (IIC) [26] and the executive vice president of the Object Management Group (OMG) [43]: "Communication amongst machines to route around failures is already happening. In fact, it's what the internet was invented to do in respect to network nodes. In a factory, say, data would be gathered by nodes in the edge and they would make decisions on how to re-route around the failed machine" [37].

Despite most of the individual concepts enabling Industry 4.0 already being used based on their respective capabilities and benefits, they need to be further interconnected to achieve their full potential. Digital Twins offer part of this interconnection, but standardization is yet missing, especially regarding automated data and value transactions. The IIC, being a program from the global OMG, is a partnership of industry, government, and academia, which has been developing multiple architectures that each aim to standardize important notions surrounding the industrial internet of things, such as security and communication protocols.

One of these standardization movements revolves around the machine data marketplace, where machines can perform transactions of both data and (financial) value autonomously, based on continuously updating parameters. In order to facilitate such a marketplace to be trusted by all parties involved, it should offer decentralized rule-based access. In collaboration with the OMG, the IOTA Foundation (IF) is developing such a standardized, decentralized data marketplace [28]. Using their state-of-the-art distributed ledger technology (DLT) called Tangle, IOTA's data marketplace allows for secure micropayments to be transacted between (autonomous) machines. As per the fundamentals of the Tangle, every network participant automatically takes part in processing and securing other transactions. Using the network is secure, scalable, and free at its core. Currently, the IOTA Foundation is working with the OMG to certify and standardize its protocol as a cornerstone for the upcoming machine economy. Their vision of opening up the locked data silos that currently exist in many corporate environments is shown in figures 2.17 and 2.18.

Figure 2.17: Closed data silos in secluded environments

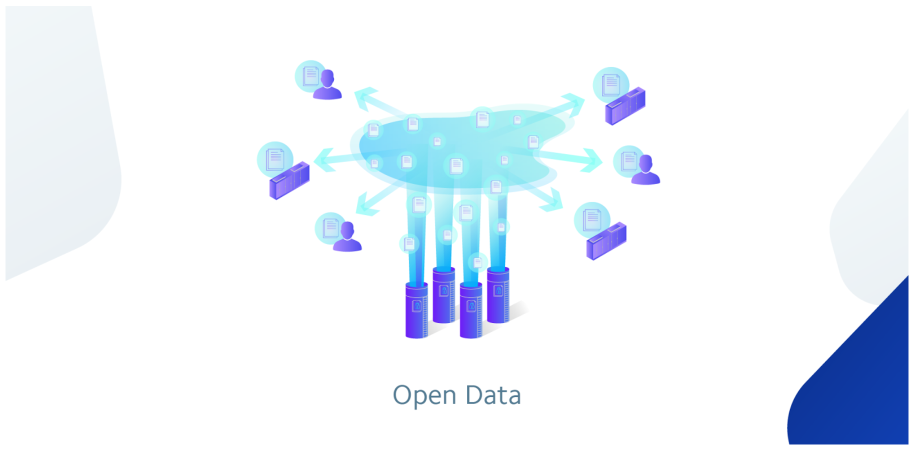


Figure 2.18: The IOTA foundation's vision of accessible, shared data silos

Another invaluable example of Digital Twin application in the near future is the notion of smart cities. In order to improve societal aspects such as waste production and energy consumption, entire cities can be built from the ground up with communication and synchronization being essential aspects. Such cities can, for example, optimize waste collection, energy provision, traffic systems, and lots more. In 2021, the IF has partnered with several entities in Europe, South Korea, and South East Asia to explore design the development of such smart cities [30]. Given the focus on M2M data communication in the IOTA Tangle, their technology is by design intertwined with Digital Twin development.

Combined with machine-to-machine communication protocols such as IOTA's Tangle, Digital Twins and data marketplaces can become essential components of the industrial equation of the future.

Chapter 3

Case Study

Throughout this thesis, various technologies and definitions surrounding digital twins and their varying levels of connectivity have been discussed. Some concepts or differences between them can benefit from showcasing a real system in the form of a case study. To effectively explore the capabilities offered by true digital twins, a hands-on experiment is developed in parallel to this research. This experiment is a case study consisting of a hardware installation combined with a Digital Model built to scale. Using sensors and communication protocols, the physical and virtual entities were synced together to experiment with the possibilities of autonomous optimization decisions by a true Digital Twin. This experiment aims to display and test parts of the theoretical background in practice and highlight the challenges that exist and remain for future projects.

As this work focuses on providing information and discussion with an emphasis on general frameworks and concepts instead of complex environment-specific implementations, only implementations with low complexity are suitable to elaborate the material. High-complexity environments present too many potentially unknown or uncertain variables to be considered as a showcase.

3.1 Preparation

In collaboration with ALTEN [3], a suitable configuration was created in the form of a hardware frame with a 3D scale model. This combination was then modified and improved to serve the Digital Twin-like purposes required for this work. The production and implementation process itself presented valuable domain knowledge as well, strengthening the theoretical foundation built until this point.

3.1.1 Physical System

The available hardware installation concerns an aluminum frame on which two "Makeblock 42BYG" stepper motors are mounted along with rubber belts connecting the motor movements to the movable head of the installation. The head can be moved around over two axes, much like a 2D printer. Previously, the stepper motors were controlled by an industrial-grade PLC (Programmable Logic Controller) and stepper motor drivers to test at which velocity certain items would tip out of the bowl on top of the movable head.

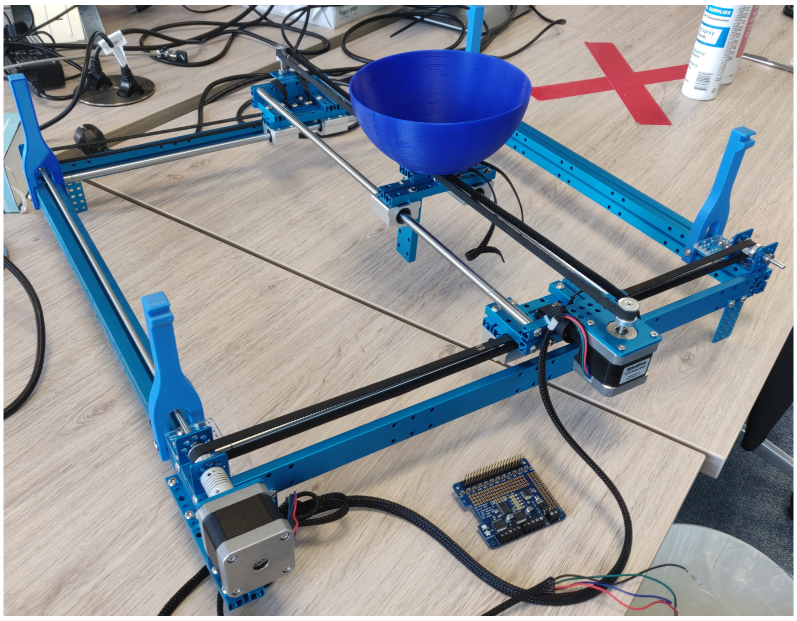


Figure 3.1: The original aluminum frame and stepper motor

Considering neither the PLC nor the stepper motor drivers were available for implementing this case study, suitable alternatives were chosen. In the original build, the choice for an industrial PLC was made in order to simulate a realistic factory environment. For the purpose of this experiment, however, we prioritize conceptual functionality over industrial feasibility in our hardware choices.

3.1.2 Digital Model

In addition to the hardware frame, a Digital Model was provided representing each of the hardware components generated in Unity3D [59]. In this virtual model, all the individual components of the physical frame have been modeled. I.e., the stepper motors, control belts, joints, and the frame pieces themselves. Furthermore, the capability to transfer motor activity towards moving the head had been implemented using Unity's physics engine.

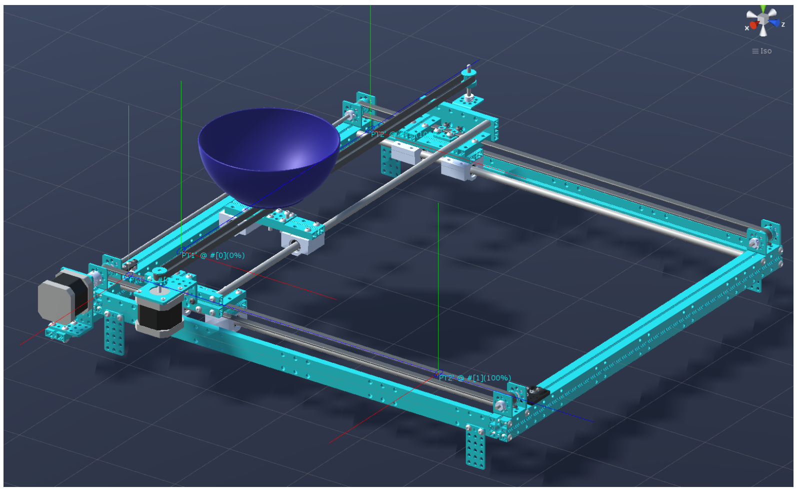


Figure 3.2: The original Unity Model

To ensure realistic interactions between motors, joints, and other moving components, an advanced library of movement scripts was provided by the company Prespective for these conceptual research purposes [45]. Prespective, as part of the Dutch company Unit040, is an experienced developer of Digital Twin simulation software. Prespective's official partnership with Unity3D enables them to build their model components and control software on Unity3D's powerful gaming engine. This engine offers superior physics capabilities compared to most industrial engines [45]. Given their state-of-the-art product development in this quickly evolving field, the library of movement scripts provided by Prespective is proprietary material. As such, it cannot be discussed in detail at the time of writing this thesis. In essence, exerting movement in the stepper motor model rotates its connected wheel-joint, which then transforms this rotational movement into a linear movement in the control belt. As the movable head is attached to the belt, it follows this movement while sliding over the spline to which it is "physically" attached through another script. The relevant physics scripts from the Prespective library are connected to each of the Unity3D models within the development environment. Each model passes on specific input parameters to the library functionality.

In the original system, the physical hardware and its virtual representation in Unity merely existed in parallel. They could be fed comparable input parameters through their respective control environments to execute similar actions (i.e., moving the head to the same coordinates). Both entities, as provided, did not communicate information back and forth, neither directly nor through some shared data space. The head's velocity and position were both hard-coded to match between the entities without synchronization during runtime.

3.1.3 Approach

The goal of this experiment showcase is to assess parts of the theoretical research in this thesis using the available physical and digital assets. For this assessment, the assets are analyzed using the various definitions and attributes at hand. The physical aluminum frame and its Digital Model counterpart will be "connected" by facilitating communication protocols in both directions. Such communications are required to unlock Digital Twin capabilities using the hardware frame and its model as its respective physical and digital entities. With this basic Digital Twin implementation in mind, we can experiment and determine what kind of improvements these capabilities enable in the test environment. Additionally, implementation-specific hurdles and bottlenecks can be examined and discussed. Recall the following research questions from the introduction of this thesis (chapter 1). This case study, combined with and strengthened by the literature in chapter 2, aims to answer these questions regarding the constraints faced when developing two-way communication in a Digital Twin system and the benefits it has to offer.

- **RQ1** – Based on the available literature knowledge, why do we create digital twins, and what benefits do they offer?

  Considerable information relevant to this research question has been provided in chapter 2 with section 2.10 in particular. This case study aims to expand and strengthen the findings in chapter 2 by discussing benefits and constraints after the implementation phase.

- **RQ2** – Can scenarios be derived in which a physical entity can be influenced or optimized based on the data found in its virtual representation counterpart?

  To help answer this research question, this case study aims to strengthen and validate the theory discussed in sections 2.6 and 2.7 by introducing scenarios in a conceptual environment in which the actions of a physical entity are altered using parameters from its virtual entity.

- **RQ3** – If so, what benefits does such influence offer?

  Building on top of the various mentions of Digital Twin benefits throughout chapter 2 and the conceptual scenarios introduced for RQ2, this case study aims to provide extra (hypothetical) scenarios based on its implementation and more generic discussion.

- **RQ4** – What kind of data or components are required in virtual and physical systems or environments to facilitate such interactions between entities?

This research question is vital in Digital Twin development and, at the same time, highly environment-specific. To develop a satisfactory answer, specific hardware for this case study will be analyzed and tested. Considering the environment-specific constraints, however, a more generic discussion approach is provided as well.

3.2 Implementation and Analysis

For the experiment execution, fitting hardware is gathered and installed to replace the previously installed PLC and industrial motor drivers. In parallel, the Digital Model in Unity is expanded and improved. Its movement computations are modified, and extra functionality is developed to enable processing of incoming and outgoing data flows.

3.2.1 Modifications towards Digital Model

In its original form, the provided system consisted of separate physical and virtual entities. Although modeled to scale, no connections or synchronization were present. When considering the theory provided in this work, the setup resembled the combination of a *Digital Model* and the specific hardware environment it was modeled to represent (section 2.4). The Unity3D model would have been considered a *Virtual Prototype* (section 2.5) had it been created as an example before constructing the physical entity, which was not the case.

As the goal for this experiment is to upgrade the system to enable Digital Twin capabilities, modifications were due on both physical and digital entities. First and foremost, the PLC used to control the steppers motors in the original system was to be replaced. The device was no longer available for this setup, and instead of replacing it with a copy, an alternative was sought for a good reason. The PLC, however durable it was, offered a relatively low computational frequency. This constraint could be problematic when trying to implement sensory capabilities later on. An Arduino Uno (offering sufficiently higher computational frequency) was considered the most feasible alternative for this setup. Programming on an Arduino requires little experience, and ample tutorials are available on the internet. Similarly compelling is the large availability of low-cost Arduino-compatible sensory equipment. Finally, its Universal Serial Bus connection enables any required communication between the physical entity and the computer running the digital entity.

The most prevalent parameter to be communicated between both entities is the position of the head within the frame. To implement and utilize communication and synchronization capabilities eventually, both entities should be able to read and mimic movements initiated by their counterpart. In order to measure the position of the head in the physical entity, a sonic distance sensor was selected, installed, and connected to the Arduino. This distance sensor is set up such that it monitors the distance between itself and the inner edge of the aluminum frame. It is mounted under the head, sharing its movements. Using this physical measurement data, the corresponding position coordinates for the virtual head can be computed. This distance sensor functions by creating short bursts of sound on a microsecond timescale in a high-pitched frequency. It then detects the return of those sound waves after they are reflected on some surface or object. The brief time frame of this event is magnitudes shorter than the computational frequency of the originally installed PLC. As such, it is the foremost reason for the incompatibility of that device. In contrast, the Arduino Uno's computational frequency is much more suitable.

With the computational controller and distance measuring devices in place, suitable power delivery and control for the stepper motor is required. The Arduino controller, while capable of distributing commands on high frequencies, is unable to supply the current required to run the motor. Installing an Arduino motor shield is the most obvious choice to overcome this, as it is specifically made to fit an Arduino's interface with minimal spatial footprint. A 12 Volt, 40 Watt power supply is selected to supply continuous current to the stepper motor rated at a phase current of 1.7A for each of its two phases.

The previous paragraph highlighted the (in)compatibility or feasibility for specific controller hardware to emphasize the importance of environments-specific hardware considerations in Digital Twin creation. Both environmental factors and envisioned future capabilities introduce hardware constraints and requirements. With this motor current of 1.7A per phase, such a hardware consideration is required again. Despite its specified ability to deliver the required current at 12 Volt to a connected motor, the Arduino motor shield is prone to overheating when doing so for prolonged durations. Its motor driver *is* theoretically able to supply the specified current, but by design, its heat dissipation capabilities are sub-par. It would not be able to supply the required current for more than a fraction of a second before overheating. A stepper motor by design requires continuous maximum current to function. Therefore a more capable motor driver is needed. Had this construction been built using regular DC motors, this constraint would have been more lenient. Sufficient current delivery and thermal dissipation capability were found in the TB6600 stepper motor driver, as suggested by Drgona and Stefun, who used it in a comparable situation of continuous current delivery [16].

With these new components selected to match the target constraints and hardware environment, the next step is their installation. Given that the original frame contained no more than a plastic bowl on top of the sliding head, some custom work was required to properly mount the hardware expansions. This custom installation is especially relevant given the substantial size of the powerful motor driver and access to the required wiring between controller, motor driver, and distance sensor. Using the digitally available scale models for the individual components, a custom mounting plate (figure 3.3) was constructed using Siemens NX modeling software [52] to be 3D-printed using Ultimaker Cura [58]. This mounting plate replaces the original bowl, with the components mounted on top or below while providing easy access to wiring. The printed mounting pieces and hardware components are shown in figure 3.4.

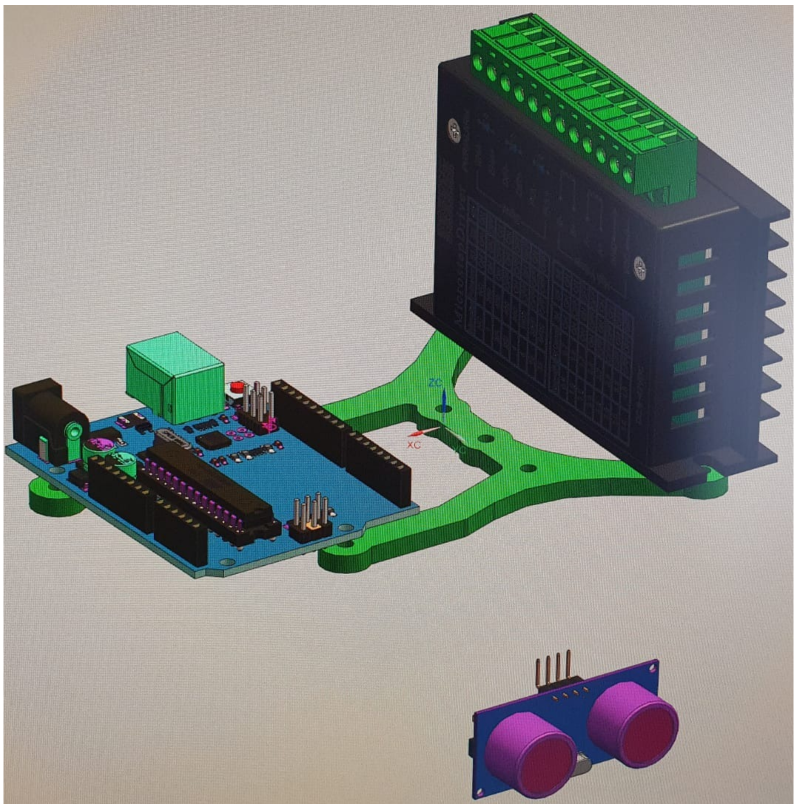


Figure 3.3: 3D model of custom mounting solution and hardware components.

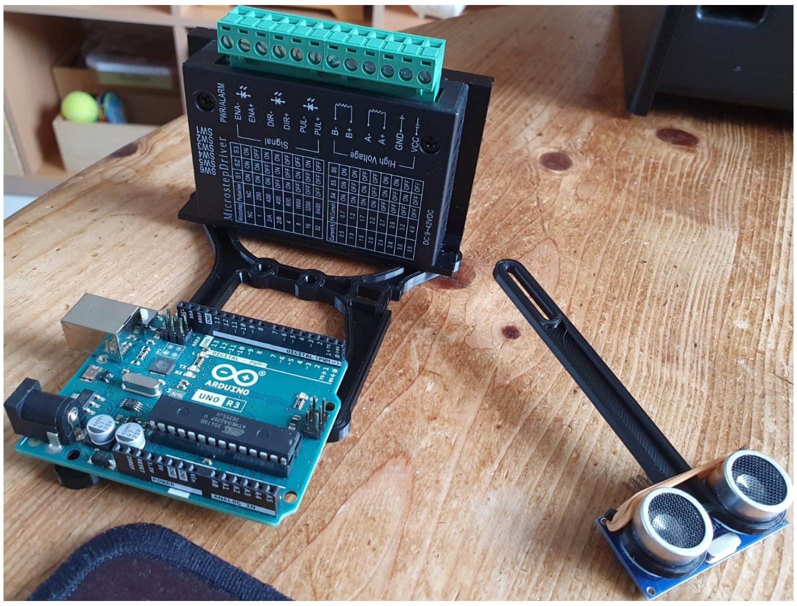


Figure 3.4: Individual pieces of custom mounting solution with the selected hardware components.

With the availability of the required hardware components and constructed mounting plate, the materials were to be installed on the sliding head of the hardware frame, resulting in the device shown in figures 3.5 and 3.6. For the distance sensor to have a continuous line of sight towards the edge of the frame, it was mounted below the head using a custom-printed vertically adjustable arm as shown in figure 3.7.

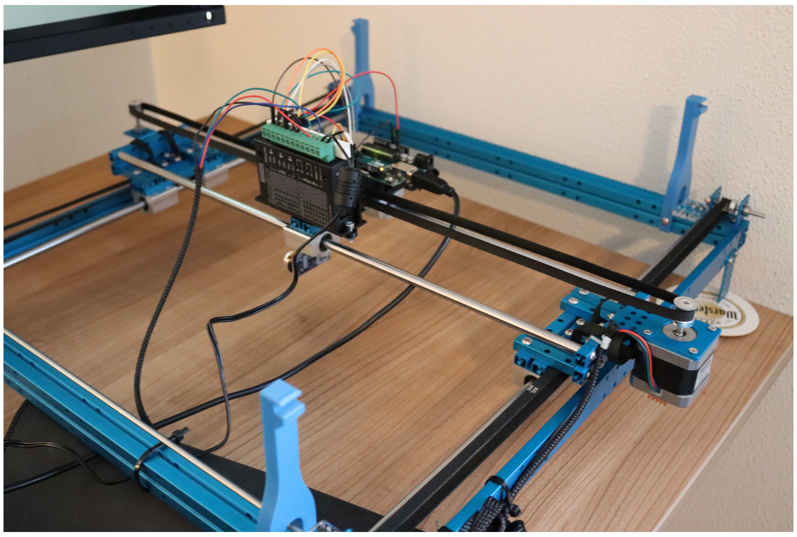


Figure 3.5: The original frame with the modifications required for Digital Twin development

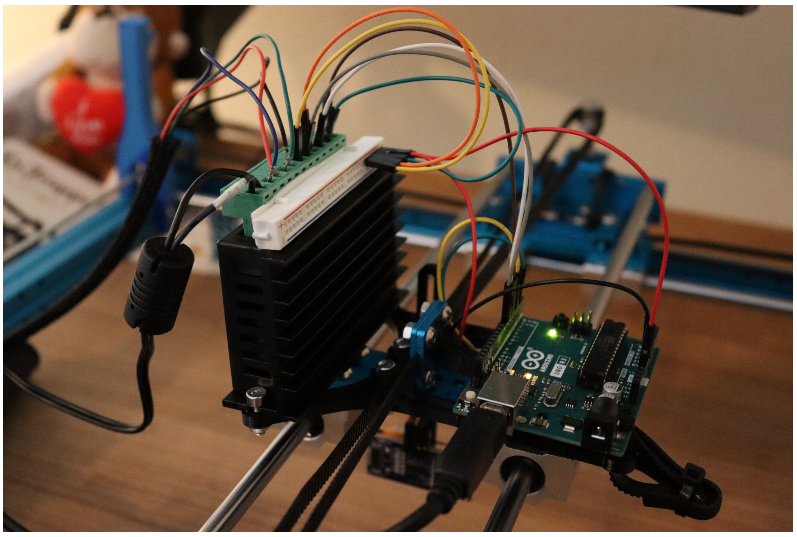


Figure 3.6: A close-up of the installation fixed to the sliding head

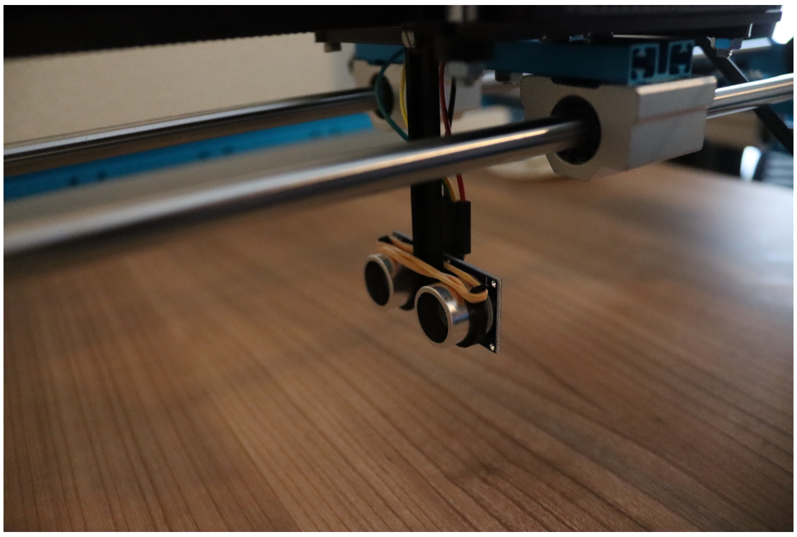


Figure 3.7: The distance sensor as mounted underneath the head

At this point, the physical equipment has been prepared for gathering sensory data and processing communications to and from a connected computer. These preparations need to be implemented in the digital scale model of the hardware setup as well. Considering a Digital Model is no more than a snapshot of a hardware environment at a specific timestamp, a new model should be drawn in order to reflect the physical modifications into the digital world. Exporting the 3D model of the printed frame-mount along with the acquired hardware component models allows us to insert them into the Unity3D model environment (figures 3.8 and 3.9).

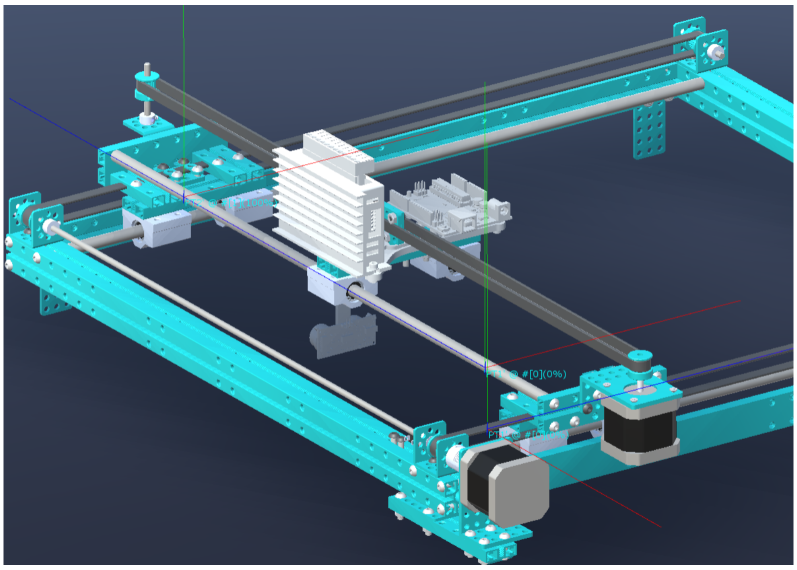


Figure 3.8: Hardware modifications represented by the Unity3D model – Front view

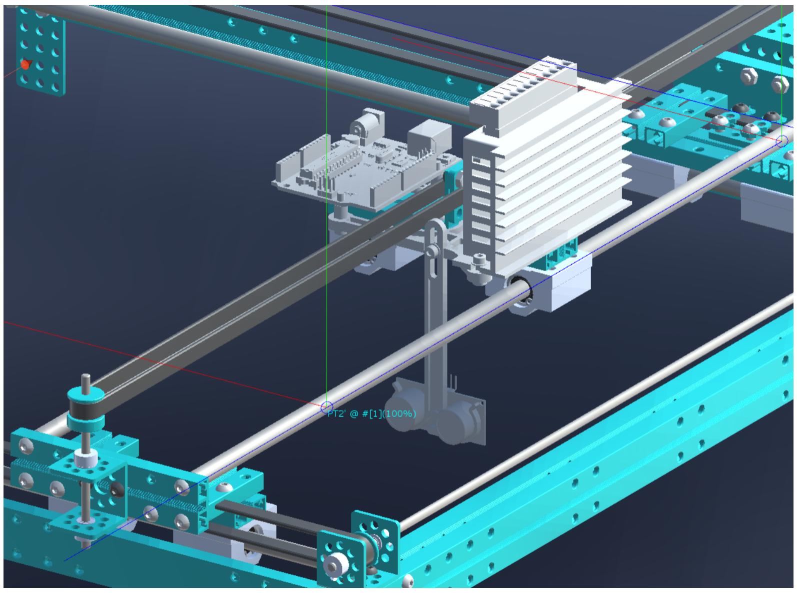


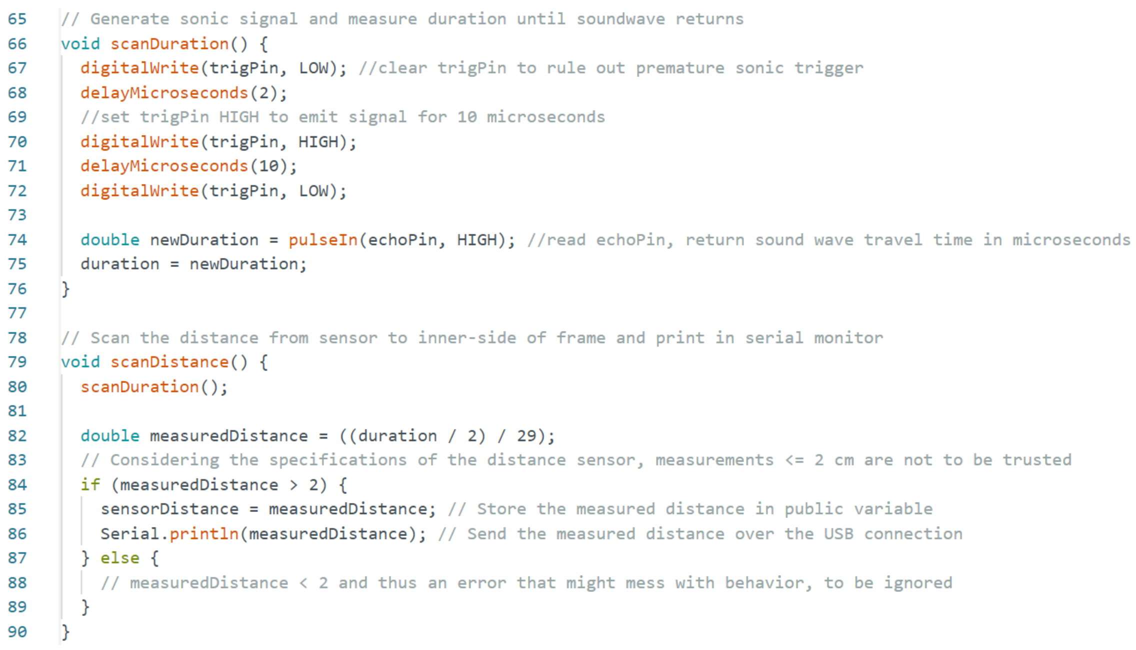
Figure 3.9: Hardware modifications represented by the Unity3D model – Back view

Even with these modifications implemented in the Digital Model representing the hardware environment as well, the setup is still no more than a Digital Model representing a static snapshot of the hardware pieces. Or, alternatively, a level 1 system according to the CMM for Digital Twins (section 2.11). The physical head can be moved by running code on the Arduino, while C# scripts within Unity3D control the digital head. Now, it is time to upgrade the setup to become a *Digital Shadow*.

3.2.2 Modifications towards Digital Shadow

To upgrade a *Digital Model* to a *Digital Shadow* means to include the capability of transferring live sensory data during runtime from the hardware environment into the Digital Model. The Digital Model can then process the same parameters and act accordingly, continuously modeling the latest state of the hardware environment (section 2.6). The first step in this process is to install hardware enabling such live parameter communication. In this case, we have already prepared both our physical hardware and virtual representation as described in the previous section.

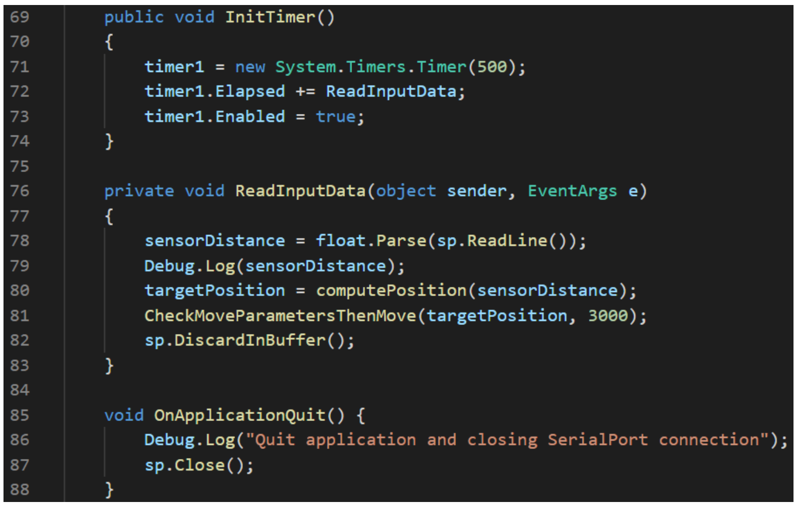
Given that this showcase is constructed to be a generic example, only essential sensory equipment is installed to enable live distance measuring. By continuously transferring the distance between the distance sensor (and by extension, the head) and the inner edge of the frame, the position of the head can be computed with an error margin of several millimeters. Considering the dimensions of the frame, such a margin is sufficient for this case study. This live distance communication was implemented through the Arduino code shown in figure 3.10. A short burst of high-frequency sound is emitted from the distance sensor to keep track of the interval it takes the sound waves to return after reflecting on the first object in their path. Morgan described the exact physical process for those interested [41]. In this case, the sensor is lined up for the reflection object to be the edge of the frame.



```
// Generate sonic signal and measure duration until soundwave returns
void scanDuration() {
  digitalWrite(trigPin, LOW); //clear trigPin to rule out premature sonic trigger
  delayMicroseconds(2);
  //set trigPin HIGH to emit signal for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  double newDuration = pulseIn(echoPin, HIGH); //read echoPin, return sound wave travel time in microseconds
  duration = newDuration;
}

// Scan the distance from sensor to inner-side of frame and print in serial monitor
void scanDistance() {
  scanDuration();

  double measuredDistance = ((duration / 2) / 29);
  // Considering the specifications of the distance sensor, measurements <= 2 cm are not to be trusted
  if (measuredDistance > 2) {
    sensorDistance = measuredDistance; // Store the measured distance in public variable
    Serial.println(measuredDistance); // Send the measured distance over the USB connection
  } else {
    // measuredDistance < 2 and thus an error that might mess with behavior, to be ignored
  }
}
```

Figure 3.10: Arduino code used to compute the distance (cm) between sensor and frame

To communicate data from the Arduino to the Unity3D scripts controlling the digital motor propulsion, a serial port connection was initiated through the USB cable connecting the Arduino to the computer running the digital environment. The measured distance is then sent over this connection at specific intervals. Next, the processed parameter is received in Unity3D on the same interval to set the target destination for the digital head and initiate its movement towards that location (figure 3.11). By reducing the timer interval for distance measuring and processing, sudden changes in movement can be accounted for with reasonable accuracy. In this conceptual implementation, the stepper motor's rotational speed, and with that, the velocity of the moving head is fixed. However, that velocity can be computed on the Arduino similar to the distance measurement. As such, it can be transferred to Unity3D and accounted for in the digital movement controller script on the same interval.



```
public void InitTimer()
{
    timer1 = new System.Timers.Timer(500);
    timer1.Elapsed += ReadInputData;
    timer1.Enabled = true;
}

private void ReadInputData(object sender, EventArgs e)
{
    sensorDistance = float.Parse(sp.ReadLine());
    Debug.Log(sensorDistance);
    targetPosition = computePosition(sensorDistance);
    CheckMoveParametersThenMove(targetPosition, 3000);
    sp.DiscardInBuffer();
}

void OnApplicationQuit() {
    Debug.Log("Quit application and closing SerialPort connection");
    sp.Close();
}
```

Figure 3.11: Unity C# Code translating the communicated physical distance measurement into digital movement

At this point, our setup offers continuous live parameter synchronization from the physical entity towards the digital entity. As such, its capabilities upgraded from those of a Digital Model to a Digital Shadow, and with that, its definition. It is now possible to monitor the movement of the hardware head during runtime without requiring direct access to the physical system. Would this system be performing some kind of meaningful task, potential inefficiencies could be more easily detected by comparing the broadcast parameters over time. The system's maturity passed all thresholds for level 2 and even partly level 3 in the CMM for digital twins.

3.2.3 Modifications towards Digital Twin

As noted, having the abilities of a Digital Shadow offers clear benefits over standalone hardware and Digital Model environments. It enables continuous (remote) monitoring of parameter data and system status. To improve our system even further, however, consider the following train of thought. After monitoring the system for a while, the system's analyst has written a script that actively maps the velocity for each type of movement (i.e., distributed by direction, distance, frequency). Using this script, the analyst can decide to change the Arduino control software that is powering the physical motor driver. They could do so for a number of reasons, among which lowering energy consumption, reducing hardware strain and degradation, or improving task performance. As is the case with most scripts and optimization code, there is an inherent benefit in striving towards automation. If the system could process, analyze, and implement suggested optimizations in the control software by itself, the analyst would no longer be required to perform this task.

At this point, our Digital Shadow does not facilitate autonomous optimization processes. To enable such processes, we need to upgrade the Digital Shadow to a "true" *Digital Twin*. Note the emphasis on "true" to stress that *only* systems with bi-directional synchronization capabilities between physical and virtual entities through a shared data space could be defined as "digital twins". Our system already facilitates parts of the capabilities offered by digital twins, as is the case for any Digital Shadow. To implement the remaining capabilities, we need to enable a reversed data flow from the digital entity to the physical entity. This reversed data flow can then be used to alter input parameters for the Arduino's motor control functions.

As every step in this development process was taken with the implementation of a true Digital Twin in mind, including reversed communication is a straightforward task. In other cases, incompatible legacy code might need to be updated first. The existing serial port connection already set up to transmit measurements from the Arduino during runtime can be utilized to write data from Unity3D scripts to the physical controller as well. It is capable of handling simultaneous data streams both ways. This way, the concept of autonomous optimization scenarios explained above can be implemented without further delay. Not only does the system facilitate changing physical parameters manually through the digital entity, it can also suggest and implement such changes by itself, for itself. As of now, we successfully modified, expanded, and upgraded a static Digital Model representing a snapshot of a hardware environment into a true Digital Twin system. As such, its maturity increased to level 5. Note that, although available, hardware and budget constraints result in an insignificant amount of actuator adaptability.

In order to preserve the generic nature of this implementation and reduce conceptual complexity, the various available forms of data storage and external services have not been discussed. Data storage in our implementation's environment takes place in the form of a simple buffer storing chronologically ordered distance measurement strings. For other more complex and sensitive systems, such a rudimentary form of storage might not be sufficient. Structured means of storage for various data parameters in a real database would increase the computational overhead and processing power requirements. Such considerations, however, are highly environment-specific and not required for this conceptual discussion. The same holds for services. The implemented services in the system at hand revolved entirely around the core functionality of both entities and their communication. However, third-party services like data visualization dashboards are possible extensions for digital shadows and digital twins as well.

3.2.4 Experimentable Digital Twin

Recalling section 2.11, our system also offers compatibility to implement an *Experimentable Digital Twin*. As discussed, implementing an EDT means expanding the functionality of either a Digital Shadow or Digital Twin by feeding live sensory parameters to custom simulation environments. For example, an optimization algorithm in our conceptual system might recommend accelerating motor rotations. An EDT could be used to test the safety and feasibility of this recommendation in an environment maintaining accuracy to the real system as closely as possible.

However, considering location (and with that velocity) data is the only available data parameter in this showcase, implementing EDT capabilities would offer no improvements. If only one (or few) parameters can be modified, none (or few) are left with which to combine and analyze these modifications. Modifying our sliding head's velocity in a simulation would overrule all live sensory data, discarding the possibility to analyze inter-parameter effects.

3.2.5 Anomaly detection example

Following the execution goal for RQ2 in section 3.1.3, an anomaly scenario is created and analyzed in this case-study environment. It serves both as a test for the implementation at hand and as an example of Digital Twin usage in low complexity environments.

For this scenario, imagine the system to be moving up and down its axis at a specific interval. For example, to move materials between other machines in a larger factory environment. At some point, the digital entity and its underlying services decide upon increasing the velocity at which the machine moves up and down its fixed path. This change could, for example, increase the number of processed items and offer financial or temporal benefits. In this scenario, a human operator could have suggested such a change as well. The system at hand supports changes by humans as well as software solutions.

However, despite the beneficial intentions driving the suggested changes, an on-site human operator was unaware of the sudden change in velocity and found himself on a collision course with the system. For visual aid, think about a commercial car wash environment. Often, human operators handle part of the cleaning efforts while cars pass through them and several machines in a steady rhythm.

The Digital Twin detects the sudden proximity of the operator through the continuous incoming data stream from the distance sensor (considering the scale of the system, the operator was simulated by fingers walking into the machine environment). To mediate the risk of the situation, the Digital Twin triggers an emergency stop procedure. This procedure results in a shutdown of any movements in the physical system and provides visual and auditory cues in the virtual entity to emphasize the state of emergency. This state, as implemented in the system at hand, is shown in figure 3.12. The motor was driving the head up and down its sliders until the “operator” walked into its environment (figure 3.13). Given the lack of additional actuators in this system, the only possible hardware clue for this emergency state is implemented in the form of the built-in Arduino LED. Upon detecting an unobstructed path, the Digital Twin resumes regular operation by restarting the physical system and continuing its detailed monitoring capabilities.

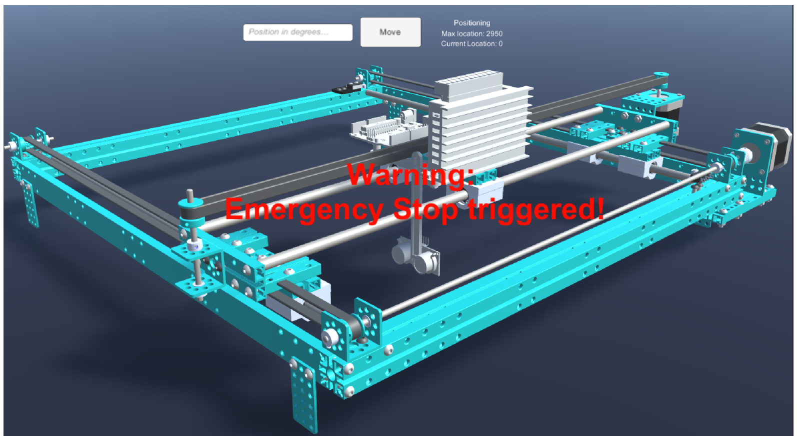


Figure 3.12: Emergency stop scenario triggered by the digital entity for a sudden blockade in the field of movement of the physical entity

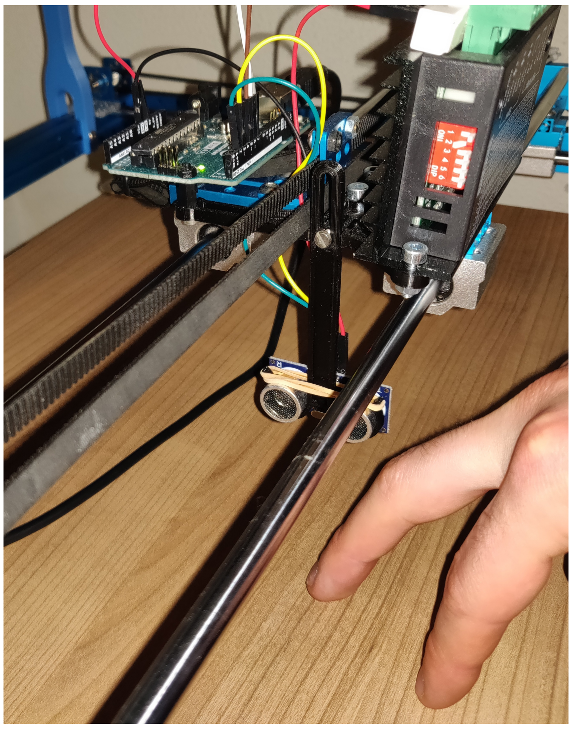


Figure 3.13: Emergency stop scenario trigger viewed from the physical entity

Alternatively, more sophisticated solutions can be implemented in a slightly different scenario.

Consider the hardware at hand to be part of a cleaning mechanism scrubbing up and down a surface. As this scenario still allows surface scrubbing to continue in the case of partial blockades, the initially implemented emergency shutdown procedure might be excessive. In such a simple, low-risk example, the Digital Twin can temporarily alter the edge parameters for its current cleaning activities. As soon as the blockade is lifted, it automatically moves on to scrub the remaining areas.

In our implementation, altering the edges of a repetitive up-and-down movement is a delicate challenge. The distance sensor is the only available sensor and is simultaneously responsible for the main activity and anomaly detection. As such, the software behind the digital entity needs to perform extra processing of incoming data streams. Computing expected positions based on a combination of previous positions and initiated movements can single out anomalies in the stream of distance data. For example, in the emergency shutdown scenario explained above, the Digital Twin continuously computes the distance traveled by the head. As soon as any new distance measurement between the sensor and the edge of the frame deviates from its expected current position, it triggers the shutdown procedure. Something, or someone, must have positioned itself between the sensor and the edge of the frame. Without extra sensors to confirm or dispel the situation, there is no room for errors.

The setup can be made significantly more resilient by including a second distance sensor facing the other end of the hardware frame. Doing so would allow for synchronized computations between the different sensor data streams, which would improve parameter reliability during regular system flows and expand anomaly detection capabilities.

3.3 Recap

During the implementation and evaluation of this case study, the remaining research questions were answered. The final implemented system offers a scenario in which the physical entity can be influenced and/or optimized based on data found in the virtual entity. Some potential benefits of such influence were discussed as well. An an environment-specific overview of the data and components required in both the virtual and physical system is provided.

Even though the functionality of the system implemented in this case study proved sufficient for the questions at hand, note that its limits were quickly reached. The inclusion of extra (distance) sensors as well as actuators (e.g., lights or speakers that trigger with the emergency procedure) can solve some of the remaining challenges. Chapter 4 offers an in-depth discussion for each research question and other important notions.

Chapter 4

Conclusion and Discussion

4.1 Conclusion

The field of Digital Twins is growing and developing across various sectors, highlighting the importance of proper academic definitions, frameworks, and model standardization. In this work, we shed light on the history, capabilities, benefits, and constraints of the Digital Twin concept and its numerous variations.

The universal importance of standardized, structured development is introduced using model design engineering. Then, the historical necessity for Digital Twins is introduced using the concept of models@run.time. The evolution from Grieves' three-dimensional Digital Twin model to the five-dimensional model by Qi et al. was traversed in detail. Alongside these concepts, the variations and subsystems of Digital Twins were discussed and compared. This discussion was followed by strengthening and structuring the provided definitions and explanations using the capability maturity model adaptation for Digital Twins. Several recent examples of Digital Twins in different industry fields offered an overview of the benefits they can offer.

It is time to look back at the initial motivation for this research thesis (section 1). First, consider RQ1: "Based on the available literature knowledge, why do we create digital twins, and what benefits do they offer?". Chapter 2, with section 2.10 in particular, has provided ample examples of the reasons for which Digital Twins are built-in industry environments. Digital Twins offer analysis, optimization, and prediction capabilities unmatched by human operators. Digital Twin application can provide benefits in the form of reduced operational and material costs, minimized downtime, optimized production output, and many others.

With the need for, and benefits of, Digital Twins explored, a case study scenario was implemented to validate and experiment with the Digital Twin variants discussed during the research phase of this work. Our initially basic hardware and software environments were sufficiently modified and expanded to explore and connect the acquired definitions and capabilities.

Combining the theoretical research and case study, consider RQ2: "Can scenarios be derived in which a physical entity can be influenced or optimized based on the data found in its virtual representation counterpart?". After providing and discussing existing and conceptual examples of such scenarios, the case study implementation validated and confirmed its feasibility. It successfully represented a scenario in which a virtual entity can influence its physical counterpart based on arbitrary parameter processing.

Next, consider RQ3: "If so (i.e., if the scenarios mentioned by RQ2 are feasible), what benefits does such influence offer?". Note the similarity between RQ1 and RQ3. Where RQ1 fueled the research and discussion comparing all variants of Digital Twins, RQ3 is explicitly aimed towards *true* Digital Twins. As most true Digital Twins are yet highly conceptual, the benefits offered by such a system are detailed by expanding scenarios for lesser variants and by introducing hypothetical scenarios.

Finally, RQ4: "What kind of data or components are required in virtual and physical systems or environments to facilitate such interactions between entities?". As mentioned in section 3.1.3, this question is vital in Digital Twin development but at the same time highly environment-specific. Throughout chapter 2, multiple real-world examples have been discussed in detail to list and compare crucial elements of Digital Twin development. The case study implementation of chapter 3 provides extra insights in this matter, both generic as specific to the implementation environment. Besides the generic basis that is ultimately described in the CMM for Digital Twins (section 2.11), specific answers to this question always require environment-specific knowledge.

Considering the scale of the global digital evolution, several key technologies and research fields are highlighted and described in the research part of this thesis as well. Progression in Digital Twin development will depend on and benefit by advances made in these other technologies and fields. Digital Twins have shown capable of improving industry systems in various sectors from all angles, and the field will continue to grow in importance while the world moves towards complete industry 4.0 expansion.

4.2 Discussion and observations

Having successfully converted the available system into a Digital Twin (chapter 3) and carefully analyzing its possibilities, some elements of uncertainty remain. These need to be considered for any implementation scenario and might hold varying importance depending on the purpose, materials, price, and size of the system.

First, consider the following hypothetical case for the implementation presented in this work. One of the optimization algorithms is programmed so that it monitors the temperature of the high-current motors. To do so, it uses a temperature sensor integrated similarly to the distance sensor (i.e., continuously synchronizing). This temperature optimization algorithm detects a drop in energy usage when the motor temperature drops below a certain threshold. In order to pursue its inherently desired energy reduction, the algorithm could broadcast commands slowing down the motor and machine movement to reduce the temperature. However, this could, in turn, trigger different optimization algorithms geared towards maximizing total movement distance or minimizing the operation time. In other words, when implementing a Digital Twin system with multiple sensors and actuators, a careful selection of optimization algorithms is required. Optimization purposes should be ordered according to their relative importance, and compromises are inevitable.

Additionally, when both the physical and virtual entities can share information back and forth with the ability to alter execution parameters on both sides, conflicts may arise. Depending on the environment in question, which entity should have the leading vote in times of disagreement? For example, in a factory environment shared by machines and human employees, it is objectively beneficial for hardware safety triggers to overrule all digital optimization suggestions. For example, introducing a higher velocity in some system processes might be beneficial for production capacity. However, if such an increase surpasses the legal noise level, the optimization should be overruled by some hard-coded limit or hardware safety trigger. In such a hybrid environment shared by humans and machines, detailed modeling of the entire environment surrounding the system is required to enable automated optimizations. Noise levels, air quality, ambient temperatures, and other sensory information are to be monitored, controlled, and optimized by the Digital Twin system as a priority. Optimizations can still be performed automatically, yet within specified boundaries for such crucial parameters.

Expanding this line of reasoning, consider a sensor noticing an unexpected object in its path for a split second. In a sealed, high-performance hardware environment, such an observation could be blamed on a sensory malfunction and is best ignored to maintain production capacity. However, in an environment shared with humans or animals, even the slightest chance of physical harm should immediately trigger a shutdown event for the surrounding machinery until further inspection dismisses the emergency procedure. To increase safety without compromising performance, high-performance systems at risk of physically harming humans or other organisms should always be contained within a monitored area. With such an area contained by borders or doorways, the system can detect hazardous entry and apply the desired safety procedures immediately, without continuously over-analyzing any anomaly during regular runtime. After all, no unauthorized personnel is present if none are detected by such a contained system. As systems tend to be built based on the assumption that environmental factors behave as specified, engineers need to decide to which extend a Digital Twin system should manage and resolve bad weather behavior. Note that bad weather behaviors, such as a person walking into a hazardous area, can be simulated in digital prototypes or models to trigger and observe the system's safety procedures [42].

Combining the previous notions that should be considered when creating a Digital Twin, another question arises. Which entity is (or should be) leading? Revisiting the theory and showcase implementation in this work, we can formulate a generic recommendation to work with. The best approach to authority division in Digital Twin systems seems to consider the physical entity as the leading data source and only propagate suggested parameter changes back from the digital entity after validating their feasibility and safety. Such validation could be performed safely by generating a snapshot for the system at a timestamp right before the suggested optimization would take form. This snapshot becomes a temporary, separate, Digital Model environment with the new parameters in place.

Finally, optimizations can go beyond modifying software parameters. Consider again the situation explained before, where a temperature sensor notices an efficiency increase below certain temperature thresholds. What if, instead of modifying functional parameters in an attempt to lower the temperature, the algorithm could suggest hardware installments such as a cooling fan? When designing a complex Digital Twin system from scratch, there is a significant chance that not every edge case will be considered by human developers or engineers. There are benefits in installing generic extra hardware components such as cooling fans and various other sensors and actuators. Unpredicted optimization scenarios might pop up after extensive system runtime and continuously changing operational sequences and processes. Of course, this approach is a double-edged sword, as installing potentially redundant hardware can add considerable financial cost that might just as well outweigh potential optimization benefits. As emphasized before, these types of reasoning are a necessity when developing any environment-specific Digital Twin system.

4.3 Future work

Some notions were found yet too immature to be included in the research and definition stage of this thesis. This section provides a brief overview of subjects that are not yet discussed but are interesting nonetheless.

4.3.1 Generative Models

Although highly conceptual and difficult to realize, Generative models are an interesting thought experiment. Recall that the ultimate goal of true Digital Twins is to automate processes and their optimization as much as possible. Also recall section 2.1 which highlighted the importance of (partly) automated transitioning from platform *independent* models to platform *specific* models. Both these goals can be reached by implementing the notion of generative models. A generative model as proposed can consist of a template digital representation for some basic system structure. By feeding live input and output parameters to this representation, the software model should be able to determine what exactly it is modeling by separating accurate measurements and outliers. Specific hardware implementations might differ across factories, and the availability of a generic Digital Model able to adjust to varying scenarios is highly beneficial. It removes the requirement of developing separate models for comparable environments.

For example, if some template digital entity receives input data of a format yet unknown (e.g., temperature data despite a lack of known thermometers), the model should detect or "realize" that it now also contains a thermometer. Inclusion of this new component results in the ability to process and utilize the, up until now, unknown data type.

4.3.2 CMM for Digital Twins

The Capability Maturity Model adaptation for Digital Twins discussed in section 2.11 is not yet formally published at the time of writing this thesis. To properly incorporate it into the common notions of Digital Twin research, other research should validate and expand the explanation provided here. Such a framework can prove a valuable asset in the strive toward industry-wide standardization of definitions and applications, as it has for other Software Engineering disciplines.

4.3.3 IOTA and the M2M economy

As mentioned in section 2.12, other industries components vital to progression in Digital Twin development are data marketplaces and machine-to-machine communication frameworks. Both these notions might well be intertwined with the Digital Twins of the future, further bolstering autonomous optimization and evolution in cyber-physical systems. Given the expected future growth in the number of entities creating and transferring data and financial value, a universal data communication protocol could be established. Such a protocol allows international parties to participate in shared networks and data stores without fear of intellectual property theft or data manipulation. Of course, to enable such a grand vision, many potential issues and risks need to be accounted for. The German IOTA Foundation (IF) is well on its way to deliver such a standardized protocol [27][28]. Their secure, scalable, and immutable DLT-based IOTA Tangle can process continuous shared data streams and micro-payments in various forms without compromising data security and validity. The IF is, in addition to several joint ventures with digital industry leaders [4][61][29], collaborating with the OMG [43]. They aim to standardize their Tangle as the communication protocol for machine-to-machine communications in the industry of the future. Given the promising results their technology has thus far delivered and its obvious valuable compatibility with (autonomous) Digital Twin systems, further research and standardization efforts are encouraged.

Bibliography

[1] Akselos. https://akselos.com/. Accessed: 13-07-2021. 20

[2] Alexopoulos, K., Nikolakis, N. and Chryssolouris, G. Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. *International Journal of Computer Integrated Manufacturing*, 33(5):429–439, 2020. vi, 19, 20

[3] Alten Nederland. https://www.alten.nl/. Accessed: 30-06-2021. 27

[4] Project Alvarium – The Backbone of Trustworthy Data. https://blog.iota.org/together-iota-and-dell-technologies-demonstrate-project-alvarium/. Accessed: 30-07-2021. 43

[5] An, J., Chua, C.K. and Mironov, V. Application of Machine Learning in 3D Bioprinting: Focus on Development of Big Data and Digital Twin. *International Journal of Bioprinting*, 7(1), 2021. 16, 19

[6] Andersson, J., De Lemos, R., Malek, S. and Weyns, D. Modeling dimensions of self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems*, pages 27–47. Springer, 2009. 7

[7] Autodesk Maya. https://www.autodesk.nl/products/maya/overview. Accessed: 19-08-2021. 11

[8] Barosan, I., Cleophas, L.G.W.A. and van den Brand, M. Capability Maturity Model for Digital Twins (work in progress, subject to change). 2021. vi, 22, 24

[9] Bauernhansl, T., Hartleif, S. and Felix, T. The Digital Shadow of production–A concept for the effective and efficient information supply in dynamic industrial environments. *Procedia CIRP*, 72:69–74, 2018. 12

[10] Blair, G., Bencomo, N. and France, R.B. Models@run.time. *Computer*, 42(10):22–27, 2009. 1, 7

[11] Blender. https://www.blender.org/. Accessed: 19-08-2021. 11

[12] Boardman, J. and Sauser, B. System of Systems-the meaning of of. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6–pp. IEEE, 2006. 16

[13] Boeykens, S., Santana Quintero, M. and Neuckermans, H. Improving architectural design analysis using 3D modeling and visualization techniques. In *Digital Heritage: Proceedings of the 14th International Conference on Virtual Systems and Multimedia*, pages 67–73. Archeolingua; Hungary, 2008. vi, 9, 10, 11

[14] Brettel, M., Friederichsen, N., Keller, M.. and Rosenberg, M. How virtualization, decentralization and network building change the manufacturing landscape: an industry 4.0 perspective. *FormaMente*, 12, 2017. 18

[15] Chen, YH., Lin, TP. and Yen, D.C. How to facilitate inter-organizational knowledge sharing: The impact of trust. *Information & management*, 51(5):568–578, 2014. 14

[16] Drgona, P. and Stefun, R. Application of Stepper Motors in CNC Device. In *2018 International Conference and Exposition on Electrical and Power Engineering (EPE)*, pages 241–246. IEEE, 2018. 31

[17] Fields of the World, open-access dataset created with support of Microsoft's AI for Earth initiative. `https://agrimetrics.co.uk/2021/06/04/fields-of-the-world/`. Accessed: 13-07-2021. 20, 24

[18] Franzen, J., Stecken, J., Pfaff, R. and Kuhlenkötter, B. Using the Digital Shadow for a Prescriptive Optimization of Maintenance and Operation. In *Interdisciplinary Conference on Production, Logistics and Traffic*, pages 265–276. Springer, 2019. 2

[19] Frimpong, S. and Li, Y. Virtual prototype simulation of hydraulic shovel kinematics for spatial characterization in surface mining operations. *International Journal of Surface Mining, Reclamation and Environment*, 19(4):238–250, 2005. vi, 11

[20] Fuller, A., Fan, Z., Day, C. and Barlow, C. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971, 2020. vii, 1, 48

[21] Godess project: Digital twin helps cut material and financial costs for offshore wind jacket foundations by up to 30%. `https://renews.biz/70920/digital-twin-helps-cut-jacket-costs-up-to-30/`. Accessed: 13-07-2021. vi, 20

[22] Grieves, M. *Virtually perfect: Driving innovative and lean products through product lifecycle management.* Space Coast Press, 2011. 8

[23] Grieves, M. Digital Twin: manufacturing excellence through virtual factory replication. *White Paper*, 1:1–7, 2014. 8

[24] Hancock, G.R., Martinez, C., Evans, K.G. and Moliere, D.R. A comparison of SRTM and high-resolution digital elevation models and their use in catchment geomorphology and hydrology: Australian examples. *Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group*, 31(11):1394–1412, 2006. vi, 9

[25] Ibrahim, M.S., Fan, J., Yung, W.K.C., Prisacaru, A., van Driel, W., Fan, X. and Zhang, G. Machine learning and digital twin driven diagnostics and prognostics of light-emitting diodes. *Laser & Photonics Reviews*, 14(12):2000254, 2020. 19

[26] Industrial Internet Consortium. `https://www.iiconsortium.org/`. Accessed: 13-07-2021. 25

[27] IOTA. `https://www.iota.org/`. Accessed: 30-07-2021. 43

[28] IOTA Data Marketplace. `https://blog.iota.org/iota-data-marketplace-cb6be463ac7f/`. Accessed: 13-07-2021. 14, 25, 43

[29] IOTA Partnerships – Building Digital Trust Together. `https://www.iota.org/solutions/partnerships`. Accessed: 30-07-2021. 43

[30] IOTA – Towards Open & Transparent Cities. `https://blog.iota.org/iota-foundation-signs-strategic-agreement-with-south-korean-obsr-foundation-and-tanglehub/`. Accessed: 30-07-2021. 26

[31] Kaur, M.J., Mishra, V.P. and Maheshwari, P. The convergence of digital twin, IoT, and machine learning: transforming data into action. In *Digital Twin Technologies and Smart Cities*, pages 3–17. Springer, 2020. 15

[32] Kent, S. Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer, 2002. 5, 6

[33] Kienzle, J., Mussbacher, G., Combemale, B., Bastin, L., Bencomo, N., Bruel, JM., Becker, C., Betz, S., Chitchyan, R., Cheng, B.HC. and others. Toward model-driven sustainability evaluation. *Communications of the ACM*, 63(3):80–91, 2020. 1

[34] Kuhn, W. Digital factory-simulation enhancing the product and production engineering process. In *Proceedings of the 2006 Winter Simulation Conference*, pages 1899–1906. IEEE, 2006. 2

[35] Ladj, A., Wang, Z., Meski, O., Belkadi, F., Ritou, M. and Da Cunha, C. A knowledge-based Digital Shadow for machining industry in a Digital Twin perspective. *Journal of Manufacturing Systems*, 58:168–179, 2021. 12, 18

[36] Lu, Y., Liu, C., Kevin, I., Wang, K., Huang, H. and Xu, X. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, 61:101837, 2020. 18

[37] The Machine-To-Machine Marketplace. `https://www.automationworld.com/factory/robotics/article/21485606/evolution-of-machine-autonomy-in-factory-transactions`. Accessed: 13-07-2021. 25

[38] Maes, P. Concepts and experiments in computational reflection. *ACM Sigplan Notices*, 22(12):147–155, 1987. 7

[39] Microsoft's AI for Earth initiative. `https://www.microsoft.com/en-us/ai/ai-for-earth`. Accessed: 13-07-2021. 20

[40] Min, Q., Lu, Y., Liu, Z., Su, C. and Wang, B. Machine learning based digital twin framework for production optimization in petrochemical industry. *International Journal of Information Management*, 49:502–519, 2019. 15, 18

[41] Morgan, E.J. HC-SR04 ultrasonic sensor, 2014. 35

[42] Nagy, I., Cleophas, L., van den Brand, M., Engelen, L., Raulea, L. and Mithun, E.X.L. VPDSL: A DSL for software in the loop simulations covering material flow. In *2012 IEEE 17th International Conference on Engineering of Complex Computer Systems*, pages 318–327. IEEE, 2012. 6, 42

[43] Object Management Group. `https://www.omg.org/`. Accessed: 13-07-2021. 5, 25, 43

[44] Paulk, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V. Capability maturity model, version 1.1. *IEEE Software*, 10(4):18–27, 1993. 21

[45] Prespective, powered by and partner of Unity. `https://prespective-software.com/`. Accessed: 27-07-2021. 29

[46] Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L. and Nee, AYC. Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 2019. vi, vi, vi, 1, 2, 3, 14, 16, 18

[47] Rauh, J. Virtual development of ride and handling characteristics for advanced passenger cars. *Vehicle System Dynamics*, 40(1-3):135–155, 2003. vi, 10, 11, 12

[48] Riesener, M., Schuh, G., Dölle, C. and Tönnes, C. The digital shadow as enabler for data analytics in product life cycle management. *Procedia CIRP*, 80:729–734, 2019. 14

[49] Schluse, M., Priggemeyer, M., Atorf, L. and Rossmann, J. Experimentable digital twins for model-based systems engineering and simulation-based development. In *2017 Annual IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2017. 17

[50] Schluse, M., Priggemeyer, M., Atorf, L. and Rossmann, J. Experimentable digital twins—Streamlining simulation-based systems engineering for industry 4.0. *IEEE Transactions on industrial informatics*, 14(4):1722–1731, 2018. vi, 2, 17

[51] Schütze, A., Helwig, N. and Schneider, T. Sensors 4.0–smart sensors and measurement technology enable Industry 4.0. *Journal of Sensors and Sensor Systems*, 7(1):359–371, 2018. 1

[52] Siemens NX for Design. `https://www.plm.automation.siemens.com/global/en/products/nx/nx-for-design.html`. Accessed: 01-07-2021. 31

[53] Steindl, G., Stagl, M., Kasper, L., Kastner, W. and Hofmann, R. Generic Digital Twin Architecture for Industrial Energy Systems. *Applied Sciences*, 10(24):8903, 2020. vi, 15

[54] Tao, F., Liu, W., Zhang, M., Hu, T., Qi, Q., Zhang, H., Sui, F., Wang, T., Xu, H., Huang, Z. and others. Five-dimension digital twin model and its ten applications. *Comput Integr Manuf Syst*, 25(1):1–18, 2019. 2

[55] Tao, F., Zhang, M. and Nee, A. Five-Dimension Digital Twin Modeling and Its Key Technologies. *Digit. Twin Driven Smart Manuf*, pages 63–81, 2019. vi, 8, 15

[56] Team, CMMI Product. CMMI for acquisition, version 1.3. *CMU SEI, Nov-2010*, 2010. 22

[57] Uhlemann, T.HJ., Lehmann, C. and Steinhilper, R. The digital twin: Realizing the cyber-physical production system for industry 4.0. *Procedia Cirp*, 61:335–340, 2017. 18

[58] Ultimaker Cura. `https://ultimaker.com/nl/software/ultimaker-cura`. Accessed: 01-07-2021. 31

[59] Unity3D. `https://unity.com/`. Accessed: 30-06-2021. 11, 28

[60] Wollschlaeger, M., Sauter, T. and Jasperneite, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017. 18

[61] Zebra Savanna with IOTA track and trace ledger API. `https://developer.zebra.com/blog/introducing-zebra-savanna-and-iota-track-and-trace-ledger-api`. Accessed: 30-07-2021. 43

[62] Zhang, H., Liu, Q., Chen, X., Zhang, D. and Leng, J. A digital twin-based approach for designing and multi-objective optimization of hollow glass production line. *IEEE Access*, 5:26901–26911, 2017. 2

Appendix A

Illustrations

A.1 Chapter 1

The image from Fuller et al. that was used as inspiration in the introduction (figure 1.1) contains an error; the arrows from digital object to physical object are pointing the wrong way.

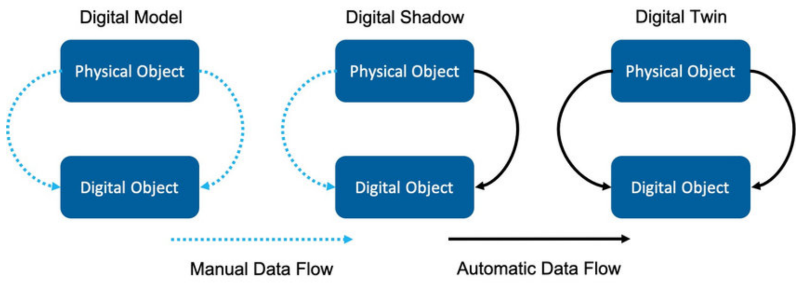


Figure A.1: Difference in data flow for Digital Models, Digital Shadows and Digital Twins [20]

A.2 Section 2.1

Some of the illustrations used to highlight or explain material discussed in this thesis were hand-crafted from multiple icons, which are all available freely available on the web. The following icons were used to create **Figure 2.1**, visualizing the flow from PIMs to PSMs, through implementations towards results:

- **PIM:** `https://img.flaticon.com/icons/png/512/2103/2103639.png?size=1200x630f&pad=10,10,10,10&ext=png&bg=FFFFFFFF`
- **PSM:** `https://icon-library.com/images/model-icon/model-icon-28.jpg`
- **Implementation:** `http://cdn.onlinewebfonts.com/svg/img_414428.png`
- **Result:** `https://static.thenounproject.com/png/3630202-200.png`

For the five-dimensional digital twin model in **Figure 2.10**, the following icons were used:

- **Physical Entity:** `https://clipartstation.com/robot-arm-clipart/`
- **Virtual Model:** `https://img.pngio.com/computer-screen-icon-png-239841-free-icons-library-com` `1600.jpg`

- **Data:** `https://cdn.icon-icons.com/icons2/508/PNG/512/database_icon-icons.com_49871.png`
- **Services:** `https://img.icons8.com/ios/452/service.png`

Appendix B

Implementation Hurdles

B.1 Hardware

During the development phase of the digital twin implementation complementing this research paper, an Arduino motorshield was the first motor driver to be tested. It overheated in a matter of seconds while tasked with supplying continuous current well within its specified limits. As such, for Digital Twin environments and their desired autonomous operation scenarios and running times, more professional equipment is a necessity. Figure B.1 displays this preliminary testing phase.

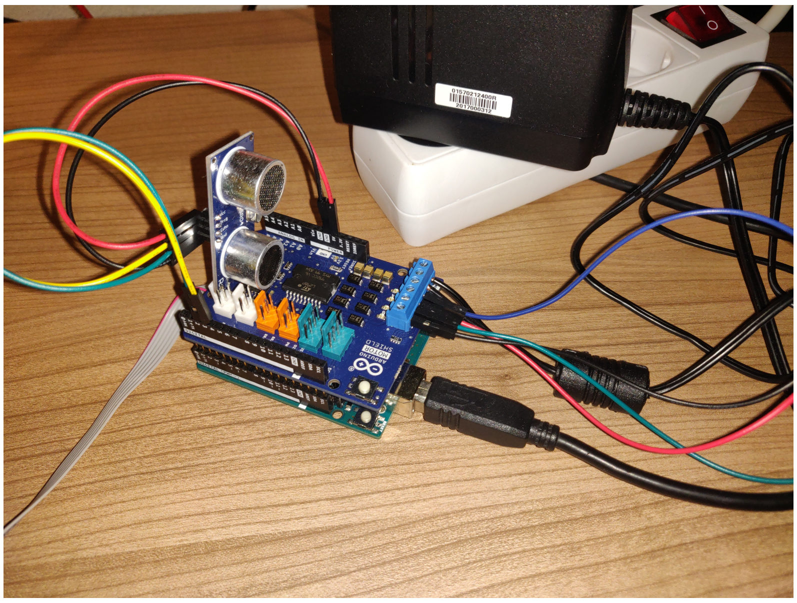


Figure B.1: Arduino Uno with external 12V input to the motor shield and distance sensor connected