# Linking Digital Twin Design and Ontologies with Model-Driven Engineering: Application to Railway Infrastructure

Alexis Chartrain[1,2], Gilles Dessagne[1], Noël Haddad[1] and David R. C. Hill[2]
[1]*SNCF* RÉSEAU, *Direction Générale Industrielle & Ingénierie, F-93210 Saint-Denis, France*
[2]*Université Clermont – Auvergne, CNRS, Clermont – Auvergne INP, Mines de Saint-Étienne, LIMOS UMR CNRS 6158, F-63000 Clermont-Ferrand, France*
{*alexis.chartrain, gilles.dessagne, noel.haddad*}*@reseau.sncf.fr, david.hill@uca.fr*

Keywords: Digital Twin, Information System, Metamodels, Meta-Object Facility (MOF), Model-Driven Architecture (MDA), Model-Driven Engineering (MDE), Object-Oriented Approach, Ontologies, Railway System.

Abstract: In this position paper, we argue that ontologies, combined with Model-Driven Engineering (MDE) approach and the object-oriented approach, could be leveraged to produce model-driven Digital Twins. This paper presents our framework for the production of model-driven Digital Twins using the aforementioned approach. Despite the interest in existing frameworks, this paper offers a new perspective that could help pave the way towards a future standardized, generic framework and shows insights of application at the French Railway Infrastructure Manager, SNCF RÉSEAU.

## 1 INTRODUCTION

The concept of "Digital Twin" first emerged in the early 2000's and has seen significant growth since 2015, as evidence by the increasing number of annual publications on the topic, particularly from the period between 2015 and 2017 (Barricelli et al., 2019; Semeraro et al., 2021). Indeed, Digital Twins (DTs) are seen as promising tools for monitoring, simulating, optimizing, and predicting the behaviour of a real-world system or product, remotely from a virtual counterpart that mirrors the actual system or product. These capabilities are highly valued in cutting-edge sectors, which is why DTs are gaining traction in manufacturing (e.g., Industry 4.0, aerospace), healthcare (e.g., hospitals), transportation (shipping, aviation, maritime, railways) (Barricelli et al., 2019; Semeraro et al., 2021; De Donato et al., 2023).

However, defining, designing, and developing a DT is by no means straightforward. One major challenge is to deal with the numerous definitions stated for the DT concept, which vary across sectors, applications, and use-cases. As a result, there is little consensus on a universal definition of this concept within the community, both in industry and academia (Tao and Qi, 2019; Adamenko et al., 2020; Zhang et al., 2021; VanDerHorn and Mahadevan, 2021; De Donato et al., 2023; Chartrain et al., 2024). Nevertheless, a clear definition is crucial to determine *what* needs to be designed and developed. Furthermore, creating a DT requires models (Tao and Qi, 2019; Chartrain et al., 2024); however despite the existence of a few frameworks, standards and methods for designing and producing DTs (Segovia and Garcia-Alfaro, 2022; Haße et al., 2022), even fewer of these approaches are model-driven (Zhang et al., 2021).

In computer science, ontologies are considered as specific, often formalized, conceptualizations accounting for a particular view or vision of the world (e.g., reality, domain of discourse) which are shared among a group of people (Gruber, 1995; Guarino, 1998; Maedche, 2002; Gruber, 2008). They are presented as structured corpus of concepts in relationship (i.e., semantic or taxonomic relations), modeled in a language allowing further seamless exploitation by computers. They are often represented by knowledge graphs and formalized through languages such as the Web Ontology Language (OWL), the Resource Description Framework (RDF), or the Unified Modeling Language (UML). A computer ontology is always built for a domain, but less frequently for a set of domains of knowledge. Ontologies offer high-potential solutions for analyzing domains, systems, and products by formalizing their inherent concepts and interrelationships, thereby providing the foundation for related unifying models. DTs of these systems or products, considered as their *digital representations* (i.e., a specific type of model) conform to



Chartrain, A., Dessagne, G., Haddad, N. and Hill, D.
Linking Digital Twin Design and Ontologies with Model-Driven Engineering: Application to Railway Infrastructure.
DOI: 10.5220/0013090200003838

In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 2: KEOD*, pages 265-276
ISBN: 978-989-758-716-0; ISSN: 2184-3228

their related meta-models, following Bézivin's works on Model-Driven Engineering (Bézivin and Gerbé, 2001; Bézivin, 2004; Bézivin, 2005).

In this paper, we argue that ontologies combined with Model-Driven Engineering (MDE) standards and techniques, and the object-oriented approach could be leveraged to produce model-driven DTs. This paper aims at presenting our proposal of framework for the production of model-driven DTs with the previously mentioned approach. Our goal is not to dismiss existing frameworks, but to offer a new perspective that could help pave the way toward a future standardized, generic framework.

In section 2, we present an overview of DTs and ontologies as part of the state of the art. We also precise the questions addressed in this paper regarding the production of DTs. Then, in section 3, we introduce our framework which aimed at facilitating the production of DTs. This framework includes definitions of the key concepts, basic object-oriented principles, Model-Driven Engineering (MDE) standards, ontologies, and the strong links between these elements. We also discuss the application of this framework at SNCF RÉSEAU, the French Railway Infrastructure Manager, within the context of producing a railway system DT as part of the company Information System. We give a very concrete example of application in railways dealing with turnouts and associated measurements, especially from the perspective of monitoring point machines current consumption with our DT. Finally, we conclude this paper in section 4.

# 2 DIGITAL TWIN AND ONTOLOGIES: AN OVERVIEW

In this section, we present a brief state of the art regarding the two main topics of our paper: on the one hand the concept of DT and, on the other hand ontologies in computer science. For both of these topics, we proceed to a short overview of their historical background and a review of existing definitions in the literature including their main characteristics.

## 2.1 Where Does the Concept of Digital Twin Comes From?

### 2.1.1 Historical Background and Definitions

The scientific community acknowledges Michael Grieves as one of the first – if not the first – author that introduced the concept of "Digital Twin" in the literature (Kritzinger et al., 2018; Barricelli et al., 2019; Madni et al., 2019; Semeraro et al., 2021; Segovia and Garcia-Alfaro, 2022). Roots of this concept were introduced in 2002 by Grieves, in his presentation slide show for the formation of a Product Lifecycle Management (PLM) center, and more precisely in the well known slide "Conceptual Ideal for PLM" (Grieves and Vickers, 2017; Barricelli et al., 2019). Although the concept was not yet designated as "Digital Twin" at that time, all the elements of Grieves' definition were already stated (i.e., a real space, a virtual space, and a data flow that connects the two spaces together) (Grieves and Vickers, 2017). These elements, referred as "Mirrored Spaces Model", were then presented in Grieves' courses on PLM at the University of Michigan in 2003. Grieves himself attributes to John Vickers the origin of the term "Digital Twin". According to the former, the latter has first proposed it in the framework of their joint research works. The authors have refined their concepts over the years from "Conceptual Ideal for PLM" to "Digital Twin", by the way of "Mirrored Spaces Model" and "Information Mirroring Model". However, the essence of these concepts remained mostly the same with the following three main aspects: (1) a *real space* containing physical products, (2) a *virtual space* containing virtual products, and (3) a connection between the two spaces, tying them together through an *information exchange* achieved by a data flow (Grieves, 2015; Grieves and Vickers, 2017).

In 2012, ten years after Grieves' preliminary works on the DT concept, NASA proposed their definition in regards of their own needs: "*A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin.*" (Glaessgen and Stargel, 2012; Barricelli et al., 2019).

In 2018, Kritzinger et al. refined the definition of the DT concept with a proposal for a categorization of the different integration levels of a physical system with a corresponding virtual system (i.e., the twin). According to the latter, three levels could be distinguished, respectively named *digital model*, *digital shadow*, and *digital twin* (Kritzinger et al., 2018). We detail each one of them thereafter.

**A digital model** is a *digital representation* of an existing or planned physical system without any automated information flow between the physical system and its corresponding representation. In this case, digital data regarding the physical system could only be set manually in the virtual system; consequently a change of state in one of the system has no effect on the other.

**A digital shadow** is a virtual system (or digital model) being automatically updated with a data flow containing information about the physical system (e.g., gathered with sensors) allowing a possible control – in the monitoring sense – of the physical system through the corresponding virtual system.

**A digital twin** is a virtual system being both capable of commanding (i.e., transmission of instructions) and controlling (i.e., monitoring of up-to-date statuses such as in the digital shadow case) a physical system.

To sum up Kritzinger's position: a digital model is a digital representation disconnected from any physical system; a digital shadow imply a one-way information flow from a physical system to a corresponding digital representation. Finally only the digital twin runs a double-way information flow between a physical system and a corresponding digital representation (Kritzinger et al., 2018; Segovia and Garcia-Alfaro, 2022). In addition, (Aheleroff et al., 2021) based their reasoning on the categorization proposed in (Kritzinger et al., 2018), but add one more kind: the *digital twin predictive* that uses the digital representation to process data and perform simulations or machine learning applications to make predictions.

Following from this position, (Wright and Davidson, 2020), defended in 2020 a clear distinction between the two notions of "model" and "Digital Twin". They argued that the existence of a physical system is an indispensable condition for a related digital representation to be designated by the term "Digital Twin". Otherwise, without any physical counterpart, the digital representation is simply designated as a "plain digital model". However, this point of view should be put in perspective with the PLM vision proposed by (Grieves and Vickers, 2017) with different types of DTs (e.g., Digital Twin Prototype, Digital Twin Instance) and the approach with further propose, both embracing the whole lifecycle of the system within the DT.

#### 2.1.2 Characteristics and Functionalities

According to our previous analysis stated in (Chartrain et al., 2024), we propose to arrange the most common characteristics and functionalities of DTs described in the existing literature within the three following aspects:

1. *A digital representation* of a precise, designated, system or product – often assumed as physical – that interest a group of stakeholders, aiming at *digitally represent* this system/product, possibly in an up-to-date or pseudo real-time manner (Barricelli et al., 2019; Madni et al., 2019).
2. *An information flow* between an actual system/product and a corresponding digital representation aiming at *controlling* (i.e., monitoring) and eventually *commanding* the actual system from its associated virtual counterpart (Grieves, 2015; Kritzinger et al., 2018; Tao and Qi, 2019; Segovia and Garcia-Alfaro, 2022).
3. *Simulations, predictions, and decisions* achieved based on the current available knowledge of the actual system (e.g., digital models, data) aiming at *anticipate the behaviour* of this actual system/product by predicting its future likely statuses as effectively as possible. This is performed thanks to statistics, simulations, optimizations, and even artificial intelligence tools, based on the digital representation of the actual system/product (Barricelli et al., 2019; Segovia and Garcia-Alfaro, 2022).

In this paper, we argue that the integration of the actual system within the corresponding DT and the predictions achieved thanks to the latter about the former, both rely on the digital representation containing all the necessary information to describe, manage, monitor, control, and predict the behaviour of this actual system of interest throughout its whole lifecycle. As a result, the digital representation is the cornerstone of the DT and will constitute our main focus from now on.

### 2.2 Ontology or Ontologies?

#### 2.2.1 Historical Background and Definitions

The term "Ontology" is a combination of the Ancient Greek words *ontos* and *logos*, the former meaning "being" or "what is", and the latter meaning "treatise" or "discourse". Ontology is a branch of philosophy that originates from Aristotle's metaphysics, a discipline interested in the study of the *being* considering its intrinsic nature, characteristics, and organization; broadly, the study of existence and reality focusing on "what exists" (Maedche, 2002; Gruber, 2008).

We consider that the field of Ontology allows practitioners of the discipline (e.g., philosophers) to produce ontologies. Guarino, then Maedche highlighted the paramount importance of differentiating the *Ontology* (i.e., the philosophical discipline) from an *ontology* – or *ontologies* in its plural form – that designate "*a particular system of categories accounting for a certain vision of the world*" (Guarino, 1998; Maedche, 2002). As for Gruber, he thinks an ontol-

ogy in philosophy in a similar manner, that is as "a systematic account of Existence" (Gruber, 1995).

Furthermore, nowadays ontologies are not only contained to the field of philosophy anymore; it has indeed considerably expanded to the field of computer science over the past three to four decades. We have recently seen various applications in Artificial Intelligence (e.g., machine learning, deep learning, Large Language Models), Computational Linguistics, Knowledge Engineering (e.g., knowledge representation, data/information modeling, object-oriented analysis, shared digital libraries of reusable concepts/knowledge components, Information Systems) and Database Theory (e.g., heterogeneous data management, multi-database systems interoperability) (Guarino, 1998; Gruber, 1995; Gruber, 2008). Like philosophers, computer scientists are also producing ontologies to organize and formalize their knowledge, for example to better understand a domain. Computer scientists first and foremost use them to make high-level languages computable. In addition, Knowledge Engineering and Database Theory are promising fields that should be considered for the design and the production of DTs, especially in the framework of Information Systems. In the light of the above, the need of better approaching the concept of an ontology with a focus on computer science clearly appears; therefore we establish a corpus of definitions of *an ontology in computer science* in the next section.

#### 2.2.2 The Concept of Ontology in Computer Science

Probably one of the most famous definitions of an ontology in computer science, is the following one stated by Gruber: "*an ontology is an explicit specification of a conceptualization*" (Gruber, 1995; Gruber, 2008). The so-called "conceptualization" is "*an abstract, simplified view of the world that we wish to represent for some purpose*" or "*an abstract, simplified view of a domain of discourse*" (Gruber, 1995; Gruber, 2008; Aßmann et al., 2006; Valiente et al., 2011). According to Guarino, then Maedche: "*an ontology refers to an* engineering artifact*, constituted by a specific* vocabulary *used to describe a certain reality, plus a set of explicit assumptions regarding the* intended meaning *of the vocabulary words*" (Guarino, 1998; Maedche, 2002). As for Ben Sta et al., they define an ontology as "*an explicit and formal shared abstract view of a part of the real world. This view is described by a whole* [set] *of tools as a vocabulary formed of concepts, relations, axioms and rules of inference*" (Ben Sta et al., 2005).

The nature and the characteristics of an ontology in computer science are for the most part, acknowledged in the literature: it is a *conceptualization* with *shared*, *explicit*, and possibly *formal* aspects. The different kinds of components of ontologies are: *concepts*, *relations*, and *axioms* (Gruber, 1995; Ben Sta et al., 2005; Aßmann et al., 2006; Gruber, 2008; Valiente et al., 2011).

Furthermore, ontologies are classified in several levels depending on their scopes. According to (Guarino, 1998) then (Maedche, 2002), there are: (1) *top-level ontologies*, also known as *upper-level ontologies*, which describe common high-level concepts of a general vocabulary (e.g., space, time, matter, object, event, action) independent from any domain, task, or problem; (2) *domain ontologies* and *tasks ontologies*, which describe concepts related to the vocabulary of a specific domain (e.g., railways) or a given task, by specializing concepts from a top-level ontology; and (3) *application ontologies*, linking domain and task ontologies by describing the roles played by domain entities through the performance of a given process, activity, or task.

### 2.3 Still Problems

As stated in the introduction of this paper and already highlighted in one of our previous publication, there are still remaining open questions about DTs (Chartrain et al., 2024). First, it appears that hardly any clear consensus on a generic, universal, definition of the concept of Digital Twin exist to date among researchers and practitioners in the community (Tao and Qi, 2019; Barricelli et al., 2019; Adamenko et al., 2020; Zhang et al., 2021; VanDerHorn and Mahadevan, 2021; De Donato et al., 2023). To illustrate this problem, let's take the following example: among the 75 papers reviewed in (Barricelli et al., 2019), only 31 provided a definition of the DT concept, counting 29 different definitions in total. Depending on the definitions provided in the papers, these papers were then categorized in the 6 following topics: (1) "*Integrated system*" (2) "*Clone, counterpart*" (3) "*Ties, links*" (4) "*Description, construct, information*" (5) "*Simulation, test, prediction*" (6) "*Virtual, mirror, replica*" (Barricelli et al., 2019). In addition, based on a corpus of 150 papers and an analysis of 30 definitions of the DT concept, (Semeraro et al., 2021) proposed a classification of these definitions following five clusters: (1) "*Lifecycle*", (2) "*Cyber-Physical Systems*", (3) "*Real and virtual spaces in loop*", (4) "*Behaviour modeling of a physical space*", (5) "*Virtual system*" i.e., replication of a physical system. In (Chartrain et al., 2024), we conjectured that numerous definitions of the DT concept might exist because it is foremost approached with use-cases (e.g., lifecycle man-

agement, control/command, predictive maintenance) and not by defining its very nature; according to Barricelli et al.: "*literature works have never described in detail the characteristics of a generic DT. Indeed, each state-of-the-art paper concentrates on the development of few components of DTs*" (Barricelli et al., 2019). In the next section, we will propose a generic definition that we hope broad enough to embrace as many cases as possible, and as reusable as possible. We will then refine this definition to suit our objective regarding the production of DTs with our framework.

Secondly, it appears as well that any DT requires models to be produced (Tao and Qi, 2019; Tao et al., 2022). In (Chartrain et al., 2024), we quoted Tao and Qi: "*To build a digital twin of an object or system, researchers must model its parts*" (Tao and Qi, 2019). We also underlined the importance of conceptual models (i.e., ontologies) since they contain all the necessary semantics (i.e., concepts and relationships) to create digital representations of systems accurately from the desired point of view in DTs. Pieces of information within digital representations are stored through data, according to the structure of concepts in related conceptual models.

A few frameworks and methods currently exist to produce DTs: we could for instance cite the DT Architecture Reference Model (Aheleroff et al., 2021), the design principles for shared DTs in distributed systems (Haße et al., 2022), the proposal for design, modeling, and implementation (Segovia and Garcia-Alfaro, 2022). Also, ISO standards were recently published: ISO 23247:2021 focused on automation & manufacturing, then ISO/IEC 30173:2023 with a larger scope. However, these frameworks, methods, and norms are only on early stages of maturation; therefore there is hardly any consensus among the community about which ones should be used in each case in order to design and produce a DT (Tao and Qi, 2019; Zhang et al., 2021; De Donato et al., 2023; Segovia and Garcia-Alfaro, 2022). Regarding model-driven DTs, only Zhang et al. describes a model-based approach using MDE techniques to design a DT considering its whole lifecycle (Zhang et al., 2021). However, the latter doesn't focus much on *modeling choices* i.e., *what* should be represented and *how*. Based on requirements, the production of models constitutes just a step in the whole framework proposed to design a complete DT in (Zhang et al., 2021). Our framework presented in the next section is our contribution to mitigate this lack and initiate a bridge between DT designing and ontology engineering.

# 3 LINKING DIGITAL TWIN DESIGN AND ONTOLOGIES WITH MODEL-DRIVEN ENGINEERING

In this section, we present our proposal of metamodeling framework regarding the design and the production of a Digital Twin as part of an Information System, using ontologies, the object-oriented approach, and existing standards coming from Model-Driven Engineering (MDE). We first introduce the object-oriented approach and the standards we are using, secondly we propose our definitions of the key concepts given this context and, thirdly we present our complete proposal. Finally, we provide an example of application at SNCF RÉSEAU, the French Railway Infrastructure Manager.

## 3.1 World Modeling Approaches

### 3.1.1 Object-Oriented Approach

The object-oriented approach relies on the concepts of *instances* and *classes* (Dahl and Nygaard, 1966). The instance is constituted of a finite set of attributes and a finite set of methods; each attribute is defined by its name (i.e., a property name) and its associated value; similarly, each method is defined by its name (i.e., a function name) and its returned value.

In Object-Oriented Programming (OOP), an instance is generated from a *class* that is, a common structure containing generic attributes and methods for instances. For that matter, a class could be considered as a model for instances: "*A class forms a model for the creation of instances which are only individual representations of this model.*" (Hill, 1996). Also, we could draw a strong link between a class and a *concept* in an ontology, considering their close definitions and features. The concepts and relationships expressed in computer science ontologies matches perfectly what we need to represent through models, and more particularly when going to an upper modeling level, further introduced with the concept of metamodel.

In the DT engineering perspective of our proposal, our goal is to instantiate the digital representation from classes and to design the classes in accordance with the concepts inherent to the actual system/product with an ontological approach. To do so, we use existing Model-Driven Engineering (MDE) standards provided by the Object Management Group (OMG), further discussed thereafter.

#### 3.1.2 Object Management Group Model-Driven Standards

Our framework requires the use of two OMG standards: the Meta-Object Facility (MOF) on the top of the metamodeling architecture and the Model-Driven Architecture (MDA). It also requires the famous four-layer metamodeling architecture, commonly illustrated with the "meta-pyramid".

This metamodeling architecture contains four levels respectively called M0, M1, M2, and M3 (Bézivin and Gerbé, 2001). M0 is the ground level in which no models are involved (i.e., 0 model = M0), it simply designates the "real world" in which real systems or products are contained. M1 is the first level of modeling, embracing *models* of the real world. M2 is the second level of modeling, concerned with *metamodels* as models of languages to produce models. Finally, M3 is the last level of modeling which contains the *unique metametamodel* able to describe itself. It is the MOF within the OMG standard (Bézivin and Gerbé, 2001; Bézivin, 2005). The M3 layer allows all metamodels in M2 to be compatible with each other. In our framework, we stick to Bézivin's *3+1 revisited organization* in which models in M1 represent a real system in M0, models in M1 conform to metamodels in M2, and lastly metamodels in M2 conform to the MOF in M3, the MOF being self-conformant (Bézivin, 2005). This reflexivity is known as metacircularity by specialists.

The MDA is a standard enabling the object-oriented analysis, design, and programming in software development, allowing an uncoupling of the results of these activities in three different kinds of models: (1) Computation Independent Models (CIM) that are requirement models presenting an analysis of a domain or a real-world system, through the formalization of related concepts which could be drawn with an ontology; (2) Platform Independent Models (PIM) stating software design choices and solutions based on the analysis in the CIM; and (3) Platform Specific Models (PSM) constituting object-oriented implementation of the former models. We position all these models at level M1 in the metamodeling architecture, this view is shared (Aßmann et al., 2006; Valiente et al., 2011). During the whole development process, we consider that a CIM is transformed in one or several PIMs, and that each PIM is transformed in one or several PSMs, depending on the project needs. This transformation relationship between models is well documented in Favre's works on MDE (Favre and Nguyen, 2005).

### 3.2 Our Definitions of the Key Concepts

#### 3.2.1 Digital Twin

To define what we have to design and produce, we first start by defining the DT concept in a broad sense that is, as an *up-to-date digital representation* of a system of interest. This representation is shared among a group of people (e.g., stakeholders working in collaboration). Furthermore, it can embrace all the necessary pieces of information for describing (e.g., functional models), monitoring/controlling, (e.g., statuses, measures, alerts, warnings), commanding (e.g., orders, instructions), and simulating (e.g., simulation models, data) the actual system.

We consider that the integration of the physical system within the corresponding digital representation depend on our use-cases and therefore may even change over time. For example, the DT could contain information about a system not yet produced, consequently without an immediate physical reality e.g., Grieves' Digital Twin Prototype (Grieves and Vickers, 2017). We give another example: a DT could first be designed to monitor a system, then upgraded later to perform control/command, and finally augmented afterwards to perform simulations based on the digital representation to inform decisions related to the actual real-world system. The digital representation should be capable of addressing all use-cases related to every phase of the actual system lifecycle during which we aim to work with the DT.

Hence, we argue that the heart of a DT lies in the digital representation in the first place. As a result, we propose to define a DT as follows:

**Definition 1.** Generic definition. *A Digital Twin (DT) is a shared up-to-date digital representation of a system of interest (Chartrain et al., 2024).*

Considering now an implementation perspective, and using the object-oriented approach we have previously introduced:

**Definition 2.** Object-oriented technical definition. *A Digital Twin (i.e., as defined in 1) could be implemented with a set of instances generated from a global and systemic class model. This representation could be stored in repositories and be accessible through services (Chartrain et al., 2024).*

Of course, this second definition is less generic and may certainly not be reusable in every case; it assumes that the design of the considered DT is achieved with the object-oriented approach. When using a model-driven approach, choices of other technical spaces are possible.

#### 3.2.2 Concepts of (Meta-)Model, Megamodel, and Ontology

To clearly grasp our framework, it is useful to precise definitions of the concepts of model, metamodel, megamodel and ontology to remove any ambiguity.

We fit the definitions provided by Minsky and Bézivin for the concept of "model". According to the former author: "*To an observer B, an object $A^*$ is a model of an object A to the extent that B can use $A^*$ to answer questions that interest him about A*" (Minsky, 1965). As for Hill, Bézivin and Gerbé: "*a model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system.*" (Hill, 1996; Bézivin and Gerbé, 2001). Moreover, a function of representation mapping models in M1 with the actual systems they represent in M0 is defined in (Bézivin and Gerbé, 2001; Bézivin, 2005), then refined in (Favre and Nguyen, 2005; Favre, 2006). This function, often written "$\mu$" and read "represents", is the essence of modeling; it is used from M1 to M0. These definitions are sufficient regarding the scope of this paper.

We rely on Favre's contributions regarding the definition of the concept of "metamodel". According to the latter, a metamodel is a model of language (i.e., a grammar) through which downer-level models could be expressed. To be more specific, a metamodel could either be defined as "*a model of a language of models*" or "*a model of a modeling language*" (Favre and Nguyen, 2005; Favre, 2006). Bézivin and Favre et al. both define a function of conformance, written $\chi$ and read "conformsTo" or "conformantTo". This conformance relationship is used to link models to metamodels from levels M1 to M2, metamodels to the MOF from levels M2 to M3, and the MOF to itself at level M3 thanks to metacircularity (Bézivin, 2005; Favre and Nguyen, 2005; Favre, 2006). The conformance relationship maps a model with the corresponding grammar through which the model is expressed (Favre and Nguyen, 2005). Furthermore, Bézivin draw a parallel between the concept of metamodel and the notion of ontology (Bézivin, 2005).

Our framework requires the concept of "megamodel", first introduced by Bézivin to define "*a model which elements represent models, metamodels and other global entities*" according to (Favre, 2006). We make use of this concept to express the major relationships (i.e., representation, conformance, and transformation) between systems, DT, models, metamodels, and the MOF within the four metamodeling layers. In addition, models could be transformed into other models within these layers thanks to a common metamodel; for that matter (Favre, 2006) introduced a function of transformation, written $\tau$ and read "transformedIn".

Based on (Gruber, 1995; Guarino, 1998; Maedche, 2002; Gruber, 2008), we consider an ontology in the computer science meaning. This supposed that, an ontology is a set of concepts and axioms in relationship with each other through semantic and taxonomic relations. It represents a specific view of the world (e.g., reality, particular domain) shared among a community. Concepts and relations in an ontology are explicitly expressed through a more or less formalized language (e.g., OWL, UML) and axioms could be expressed with a mathematical or first-order logic formalism.

### 3.3 Our Generic Proposal

#### 3.3.1 Synoptic Megamodel

We illustrate our proposal of framework for producing a DT with MDE through our synoptic megamodel presented in Figure 1.

In our megamodel, the DT is a digital system at level M1 that `represents` (i.e., $\mu$) the actual real-world system at level M0. Stakeholders can interact with the actual system through its virtual counterpart (e.g., system management, monitoring, control/command). The virtual system can also address other use-cases regarding the actual system, such as predictions through statistics or simulations. The digital representation is constituted of a set of digital objects, `instanceOf` a digital library of objects (i.e., concepts describing the actual system). A chosen object-oriented PSM is embedded in this library. The digital representation and the digital library from which it is instantiated are both contained at level M1. As stated in (Bézivin, 2005), there is a programming conformance relationship between instances and classes. The same apply here between the digital representation and the digital library, however it is not explicitly shown in the diagrams since only metamodeling conformance relationships are drawn.

This library programmed in a given PSM is the last product of chain of transformations following the MDA: first a CIM is defined to formalize all the necessary concepts and relationships related to the real-world system. The production of the CIM is comparable to the establishment of a domain ontology. This CIM is then `transformedIn` (i.e., $\tau$) into one or several PIMs in order to satisfy development constraints and choices. Each PIM is a software solution that is then `transformedIn` into one or several PSMs that are implementation models. CIM, PIMs, and PSMs are all contained at level M1.

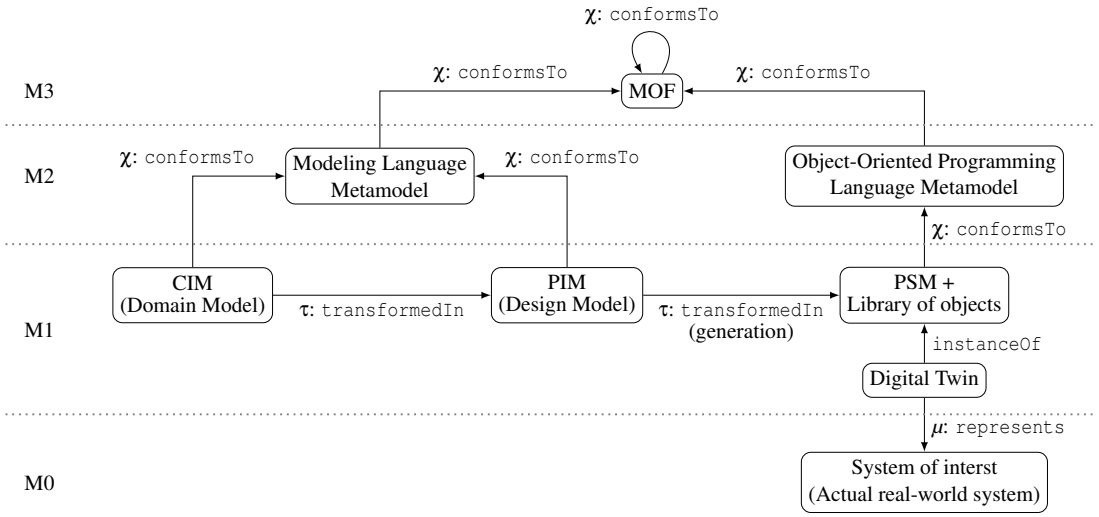


Figure 1: Synoptic megamodel of our framework for producing a DT with MDA and the MOF.

The CIM and the PIMs `conformsTo` the metamodel of the modeling languages they are expressed with (e.g., OWL, UML). CIM and PIMs could be expressed with the same or different modeling languages. The digital library of objects `conformsTo` the metamodel of the object-oriented programming language it is coded with. All these metamodels are contained at level M2 and each of them `conformsTo` the MOF at level M3.

#### 3.3.2 Leveraging Ontologies to Formalize Models

Following from the above, it clearly appears that the very starting point of the production of the real-world system digital representation, cornerstone of the DT, is its corresponding domain model. The latter is expressed through as CIM using the MDA, as mentioned previously. Then, one more question remains: how could we produce this domain model? Since the domain is a formalization of a set of concepts in relationship describing the real-world, and more specifically the scope concerned with the aforementioned real-world system, it could be established such as an ontology. This ontology would be a domain ontology, according to Guarino's well known classification of ontologies (Guarino, 1998; Maedche, 2002).

Furthermore, this domain ontology will require to commit from an upper-level ontology (i.e., also known as top-level ontology) to reuse, refine, and specialize really high-level concepts (Guarino, 1998; Maedche, 2002). Several upper-level ontologies exist to date, one of the most popular is the Unified Foundation Ontology (UFO) (Guizzardi, 2005). Others could be mentioned such as *Basic Formal Ontology* (BFO), *Cyc*, *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE), *General Formal Ontology* (GFO), *PROTo ONtology* (PROTON), Sowa's Ontology, and Suggested Upper Merged Ontology (SUMO). All these seven upper-level ontologies are compared in (Mascardi et al., 2007). We conjecture that the choice or the production of a top-level ontology depends on ontological and epistemological assumptions about *what is reality* and *how do we know what we know*, including our knowledge about reality. However, the scope of this paper is not to address this question, but rather to open the discussion on this subject.

### 3.4 Application at SNCF RÉSEAU

#### 3.4.1 Applied Synoptic Megamodel

We present in Figure 2 the application of our proposal of framework for producing a DT with MDE, previously presented in a generic way in Figure 1.

In our company, the actual system we aim at twining is the railway system. The Digital Twin, seen in our framework as *a shared up-to-date digital representation of the railway system* among the stakeholders working at and with the company, embraces:

- Functional aspect of the infrastructure (e.g., requirements, description of the French Railway Network within schematics and/or Building Information Modeling (BIM) mock-ups) considering its complete lifecycle i.e., the *as-designed* phase embracing the history of all the different versions of each design study, the *as-built* phase considering the actual construction of the infrastructure and the possible gaps with the *as-designed* requirements, and finally the *as-is* phase focusing

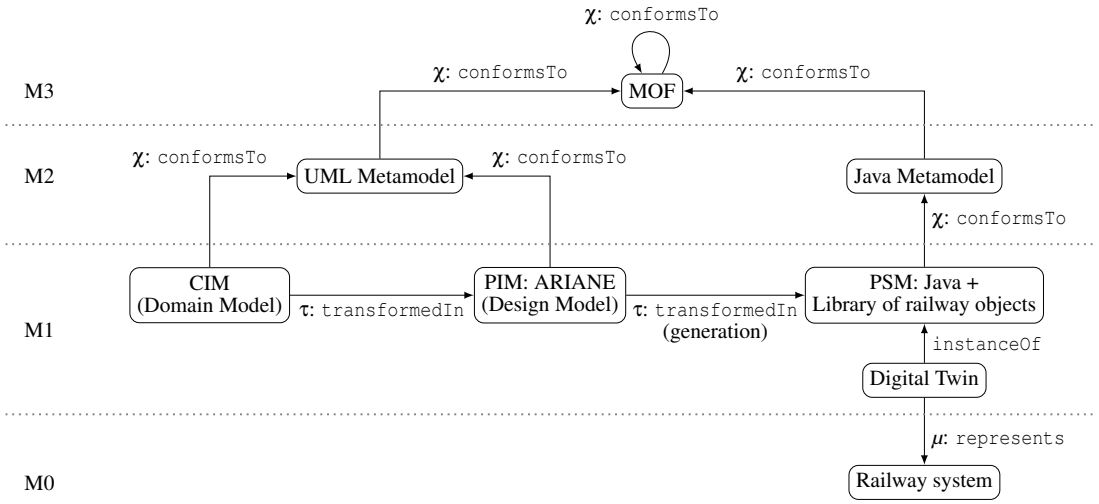


Figure 2: Synoptic megamodel of our framework for producing a DT, applied at SNCF RÉSEAU.

on the current configuration of the infrastructure in which it is operated. This functional aspect include the description of field elements of the infrastructure e.g., tracks, tunnels, bridges, electrical substations, catenaries, signaling items;

- Physical aspect of the infrastructure that is, actual real-world field elements deployed on the French Railway Network considering their complete lifecycle i.e., the *as-built* and the *as-is* phases. Specific components of deployed field elements are supervised for asset management and maintenance purposes e.g., turnout frogs;
- Up-to-date measures and statuses related to the physical infrastructure e.g., sensors controlling the position of turnout points, sensors monitoring the current consumption of point machines, sensors ensuring rock-fall, flood, and on-track hotbox detectors. Each measurement and status is saved through an history for further analysis;
- Railway capacity allocation of the rolling stock e.g., scheduled train paths, real-time running train circulations, working zones.

The digital representation is instantiated in repositories, thanks to our library of objects (i.e., classes describing railway concepts) in which the Java PSM is embedded. Java is the main PSM used, since it has been adopted as a coding standard in our company. Therefore, the object-oriented programming metamodel in our framework applied at SNCF RÉSEAU is the Java metamodel. However, other object-oriented languages could be used as well, such as C++ for simulation applications. In this case, the digital library would be written in several object-oriented programming languages, each one conforming to its corresponding metamodel, all metamodels still conforming to the MOF.

The digital library is produced based on our PIM named "ARIANE" which is a software solution (i.e., a design model containing roughly a thousand classes at the moment) based on the requirements of the CIM (i.e., a railway domain model). All the concepts related to the railway system are contained in the CIM, produced with a global, systemic, and ontological approach. "ARIANE" is a reference PIM for our company which has been chosen as input, especially its topology package, for the establishment of the International Union of Railways (UIC) standard called "Rail System Model" (RSM) (Tane et al., 2022; Chartrain et al., 2024). At SNCF RÉSEAU, the CIM and the PIM are both produced in UML, consequently they both conform to the UML metamodel.

### 3.4.2 Digital Twin and Information System

At SNCF RÉSEAU, our DT is on the one hand, automatically updated by the IoT (e.g., sensors) and, on the other hand also manually updated by end-users and data managers (Chartrain et al., 2024). It is our reference source of information regarding the railway system, guaranteeing both the unicity and the accuracy of data and therefore information (Issa et al., 2024). As a result, the DT is a major component of our Information System in the company.

We previously published our DT reference technical architecture in (Issa et al., 2024). It is achieved with a Service-Oriented Architecture (SOA) and more specifically a Representational State Transfer (REST) Architecture, allowing end-users (e.g., stakeholders working directly in our company or in partnership with our company) to access whole or part of the digital representation contained in the repositories

through RESTful web-services (Issa et al., 2024; Chartrain et al., 2024). Software applications call web-services and shape the information in the required format depending on the needs of each end-user.

#### 3.4.3 Example in Railways: Turnouts and Measurements

A turnout, also called switch, is a railway field element made up of a set of points (i.e., movable elements of the turnout) and a frog (i.e., central crossing piece of rail within the turnout). Turnouts enable the setting of a given direction at a local bifurcation within the route of a train. In this example we are particularly interested in electrically commanded and controlled turnouts, respectively by the way of point machines and sensors that check the validity of the commanded position after the movement of the points.

The following aspects of turnouts are currently stored within our DT repositories:

- A *physical* aspect including: (1) each identified frog of each turnout within the scope of the French Railway Network and (2) rails, sleepers, and ballast related to the track section within which the turnout is embraced. These latter elements are only instantiated at the scale of the track section and not at the scale of the turnout;
- A *functional* aspect including: (1) the network topology ensured by crossings and turnouts (2) a precise description of each turnout with especially its corresponding orientation (e.g., left, right, or symmetric/Y-shaped), number of branches, number of points. This description is usually achieved through track and signaling schematics. The integration of turnouts models contained in BIM mock-ups within our DT is foreseen;
- *Measurements* related to the position of the points and to the current consumption of point machines.

The execution of our framework regarding turnouts and their related measurements (e.g., points position, point machines current consumption) requires the establishment of a domain model for both of these notions i.e., models of "turnout" and "measurement". These are components of a CIM, further transformed into corresponding packages in our PIM that is, ARIANE at SNCF RÉSEAU. Based on the latter, corresponding Java classes related to the concepts of "turnout" and "measurement" are generated. These Java classes are then instantiated in our DT repositories with actual data about the functional and physical aspects of turnouts, along with related measurements.

Figure 3 shows a simplified domain model related to a generic concept of "measurement" through a UML class diagram. It is adapted for the case of SNCF RÉSEAU in order to fit the needs of our company. However, this model is not complete: it has been reduced given the informative context provided in this paper. In our approach, this generic model is specialized for each context and measured item.

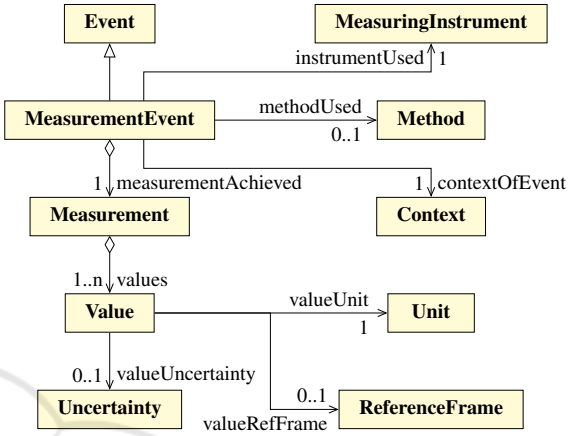


Figure 3: Simplified generic measurement domain model.

Given that the access to our PIM ARIANE is restricted for intellectual property reasons, we cannot show a corresponding class diagram extracted from our PIM. Nevertheless, we provide a link towards RSM[1], the UIC standard railway PIM that we have previously mentioned above. In this latter model, a model of the concepts of "turnout" and "measurement" can be respectively found in the `<<Domain>> Track` and the `ObservationAndMeasure` packages. However, ARIANE is still more detailed at the moment and therefore our PIM related to the concept of "measurement" is closer to the domain model presented in Figure 3 than the model presented in RSM.

To complete this example, we provide in Figure 4, an instance diagram (i.e., a UML objects diagram) that shows how Java classes could be instantiated in the DT repositories at SNCF RÉSEAU. In this example, instances are related to the monitoring of a point machine current consumption, when the points are maneuvered from one side of the turnout to the other by the point machines.

## 4 CONCLUSIONS

The concept of Digital Twin (DT) have first appeared in the early 2000's. Since 2015, there has been a sig-

[1] https://rsm.uic.org/doc/rsm/rsm-1-2/

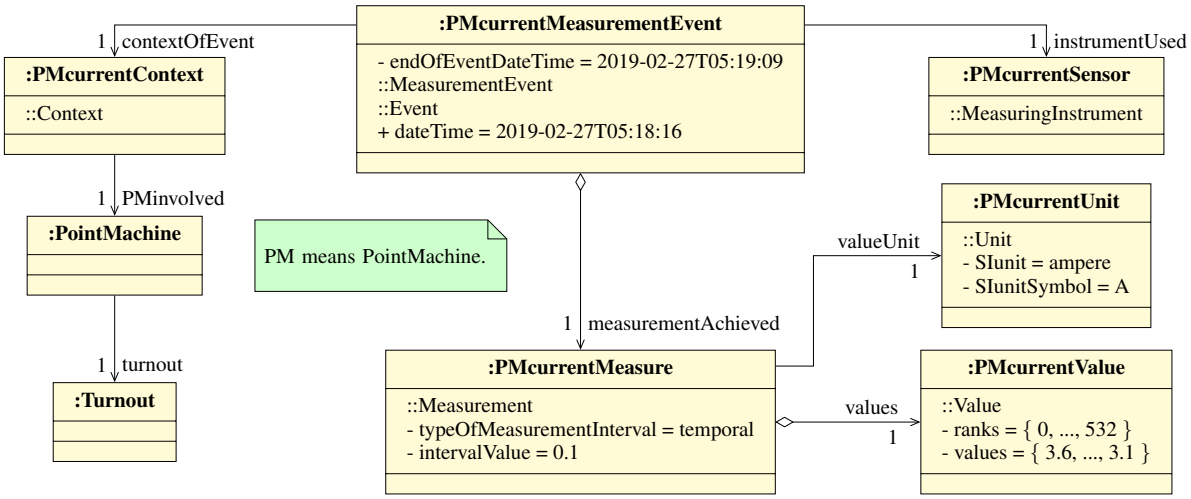


Figure 4: Simplified instantiation example illustrating the measurement of point machines current consumption.

nificant growth in scientific literature on this topic, and numerous concrete applications have emerged in various sectors such as manufacturing, healthcare, and transportation. DTs are promising tools to manage, monitor, control, simulate, and predict the behaviour of a real-world system.

In this paper, we first provided an overview of these concepts by studying their historical backgrounds and conducted a state of the art of their definitions in literature. Based on our analysis, there is little consensus regarding a generic, universal definition of the DT concept across industry and academia. However, we emphasized that a clear definition is nevertheless required to guide the design and the production of a DT. In addition, there are some frameworks, methods, and standards to produce DTs but there is currently no widely accepted generic or standardized framework.

Furthermore, we underlined that a DT requires models in order to be built, however among the existing frameworks, very few focus on the production of model-driven DTs. Ontologies which are set of concepts and axioms related to each other through semantic and taxonomic relations, represent a specific view of the world shared by a community. They offer relevant solutions for analyzing domains, systems, and products and formalize related computable models. The latter are of paramount importance to further produce the digital representation of the system or product to be digitally twined.

In this position paper, we argue that ontologies, Model-Driven Engineering, and object-oriented principles can be leveraged to produce model-driven Digital Twins. We propose a framework designed to assist in the creation and development of DTs based on these latter methods, hoping that it could help paving the way towards a future standardized, generic framework. We provide insights into the application of our framework at SNCF RÉSEAU, the French Railway Infrastructure Manager.

# REFERENCES

Adamenko, D., Kunnen, S., Pluhnau, R., Loibl, A., and Nagarajah, A. (2020). Review and comparison of the methods of designing the Digital Twin. *Procedia CIRP*, 91:27–32.

Aheleroff, S., Xu, X., Zhong, R. Y., and Lu, Y. (2021). Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model. *Advanced Engineering Informatics*, 47:1–15.

Aßmann, U., Zschaler, S., and Wagner, G. (2006). Ontologies, Meta-models, and the Model-Driven Paradigm. In Calero, C., Ruiz, F., and Piattini, M., editors, *Ontologies for Software Engineering and Software Technology*, pages 249–273. Springer, Berlin, Heidelberg, Allemagne.

Barricelli, B. R., Casiraghi, E., and Fogli, D. (2019). A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access*, 7:167653–167671.

Ben Sta, H., Ben Said, L., Ghédira, K., Bigand, M., and Bourey, J.-P. (2005). Cartographies of Ontology Concepts. In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 486–494, Miami, FL, USA. SciTePress - Science and and Technology Publications.

Bézivin, J. (2004). In Search of a Basic Principle for Model Driven Engineering. *The European Journal for the Informatics Professional, Novatica*, 5(2):21–24.

Bézivin, J. (2005). On the Unification Power of Models. *Softw Syst Model*, 4(2):171–188.

Bézivin, J. and Gerbé, O. (2001). Towards a Precise Definition of the OMG/MDA Framework. In *Proceed-*

*ings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 273–280, San Diego, CA, USA. IEEE.

Chartrain, A., Dessagne, G., Haddad, N., and Hill, D. R. (2024). Retrospective on the Digital Twin concept and perspectives for railways: the case of SNCF Réseau. In *2ème Congrès Annuel de la SAGIP (SAGIP '24)*, Lyon, France. Société d'Automatique, de Génie Industriel et de Productique (SAGIP).

Dahl, O.-J. and Nygaard, K. (1966). SIMULA: An ALGOL-based simulation language. *Comm. ACM*, 9(9):671–678.

De Donato, L., Dirnfeld Turocy, R., Somma, A., De Benedictis, A., Flammini, F., Marrone, S., Samanazari, M., and Vittorini, V. (2023). Towards AI-assisted digital twins for smart railways: Preliminary guideline and reference architecture. *Journal of Reliable Intelligent Environments*, 9:303–317.

Favre, J.-M. (2006). Megamodelling and Etymology. In *Transformation Techniques in Software Engineering*, volume 5161 of *Dagstuhl Seminar Proceedings*, Wadern, Germany. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Favre, J.-M. and Nguyen, T. (2005). Towards a Megamodel to Model Software Evolution Through Transformations. *Electr. Notes Theor. Comput. Sci.*, 127:59–74.

Glaessgen, E. and Stargel, D. (2012). The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In *53rd Structures, Structural Dynamics, and Materials Conference: Special Session on the Digital Twin*, pages 1–14, Honolulu, Hawaii. American Institute of Aeronautics and Astronautics.

Grieves, M. (2015). Digital Twin: Manufacturing Excellence through Virtual Factory Replication. *White Paper*, 1:1–7.

Grieves, M. and Vickers, J. (2017). Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In Kahlen, F.-J., Flumerfelt, S., and Alves, A., editors, *Transdisciplinary Perspectives on Complex Systems*, pages 85–113. Springer International Publishing, Cham.

Gruber, T. (2008). Definition of Ontology. https://tomgruber.org/writing/definition-of-ontology.

Gruber, T. R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing? *International Journal of Human-Computer Studies*, 43(5-6):907–928.

Guarino, N. (1998). Formal Ontology and Information Systems. In *Formal Ontology in Information Systems. Proceedings of the First International Conference (FOIS '98)*, Frontiers in Artificial Intelligence and Applications, pages 3–15, Trento, Italy. IOS Press.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. PhD thesis, University of Twente, Enschede, The Netherlands.

Haße, H., van der Valk, H., Möller, F., and Otto, B. (2022). Design Principles for Shared Digital Twins in Distributed Systems. *Bus Inf Syst Eng*, 64(6):751–772.

Hill, D. R. C. (1996). *Object-Oriented Analysis and Simulation*. Addison Wesley, Harlow, England ; Reading, MA, USA, 1st edition.

Issa, M., Chartrain, A., Viguier, F., Landes, B., Dessagne, G., Haddad, N., and Hill, D. R. (2024). Railway system Digital Twin: A tool for extended enterprises to perform multimodal transportation in a decarbonization context. In *Transport Research Arena 2024 (TRA '24)*, Dublin, Ireland.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., and Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022.

Madni, A. M., Madni, C. C., and Lucero, S. D. (2019). Leveraging Digital Twin Technology in Model-Based Systems Engineering. *Systems*, 7(1):1–13.

Maedche, A. (2002). Ontology — Definition & Overview. In Maedche, A., editor, *Ontology Learning for the Semantic Web*, volume 665 of *The Kluwer International Series in Engineering and Computer Science*, pages 11–27. Springer, Boston, MA, USA, 1 edition.

Mascardi, V., Cordì, V., and Rosso, P. (2007). A Comparison of Upper Ontologies. In *Dagli Oggetti Agli Agenti. Agents and Industry: Technological Applications of Software Agents*, pages 55–64, Genova, Italy. Seneca Edizioni.

Minsky, M. L. (1965). Matter, Mind and Models. In *Proeedings of International Federation of Information Processing Congress 1965*, volume 1, pages 45–49, New-York City, NY, USA.

Segovia, M. and Garcia-Alfaro, J. (2022). Design, Modeling and Implementation of Digital Twins. *Sensors*, 22(14):1–30.

Semeraro, C., Lezoche, M., Panetto, H., and Dassisti, M. (2021). Digital twin paradigm: A systematic literature review. *Computers in Industry*, 130:1–23.

Tane, P., Dessagne, G., Janssen, B., and Magnien, A. (2022). The case for a federated digital model of the rail system. *Global Railway Review*.

Tao, F. and Qi, Q. (2019). Make more digital twins. *Nature*, 573(7775):490–491.

Tao, F., Xiao, B., Qi, Q., Cheng, J., and Ji, P. (2022). Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389.

Valiente, M.-C., Vicente-Chicote, C., and Rodríguez, D. (2011). An Ontology-Based and Model-Driven Approach for Designing IT Service Management Systems. *International Journal of Service Science, Management, Engineering, and Technology*, 2(2):65–81.

VanDerHorn, E. and Mahadevan, S. (2021). Digital Twin: Generalization, characterization and implementation. *Decision Support Systems*, 145:1–11.

Wright, L. and Davidson, S. (2020). How to tell the difference between a model and a digital twin. *Adv. Model. and Simul. in Eng. Sci.*, 7:1–13.

Zhang, L., Zhou, L., and Horn, B. K. P. (2021). Building a right digital twin with model engineering. *Journal of Manufacturing Systems*, 59:151–164.