Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

HIERARCHICAL HYBRID AUTOMATA: A THEORETICAL BASIS FOR THE DIGITAL TWIN APPROACH

Mamadou K. Traoré[1]

[1]Dept. of Engineering and Digital Sciences, University of Bordeaux, Talence, FRANCE

ABSTRACT

The Digital Twin (DT) concept has garnered significant interest in recent years, with its potential to improve system efficiency through data synchronization and model updates. Unlike traditional modeling techniques, DTs dynamically incorporate operational data to simulate "what-if" scenarios. However, the growing number of definitions and applications of DTs across industries highlights the need for a unified theoretical framework. This paper proposes a system-theoretic approach to formalize the concept of DT, aiming to contribute to a comprehensive DT theory. The paper introduces the DMSμ framework and explores the potential of hierarchical hybrid automata to offer within that framework a formalism that supports symbolic manipulation. It provides a foundation for advancing the formalization and application of DT technology across diverse fields.

1 INTRODUCTION

The concept of Digital Twin (DT) has gained considerable attention in recent years, drawing significant interest from both academic and industrial sectors. The DT paradigm is an innovative model-based approach that, unlike traditional methods, not only utilizes digital models and simulations to support a system's entire lifecycle (Zeigler et al. 2018), but also takes advantage of the vast operational data generated by smart technologies to enhance the models in use. These models allow for the exploration of "what-if" scenarios to identify the optimal operating conditions of the system, offering greater efficiency and accuracy compared to traditional models.

However, this burgeoning field has a multitude of interpretations and definitions originating from various disciplines. As a general definition provided by the Digital Twin Consortium, a DT of a real-world physical entity, process, or product is based on its digital model along with real-time and historical data, synchronized with a specified frequency and fidelity (Budiardjo and Migliori 2021). This synchronization serves to reflect real-world events on the model, enabling the assessment of management initiatives on this ever-updated artifact before implementation. Hence, the model is more than a simple representation, evolving into a digital counterpart intricately linked to the specific system in question. This synchronization principle is also what draws a line between a traditional digital model and a DT (Ali et al. 2024). It then serves as the basis to introduce conceptual and formal elements to contribute to a theory of DT. Such a theory is very needed, both to disambiguate the concept, offer possibilities of symbolic manipulation, and lay down building blocks for structuring a full body of knowledge of DT technology.

Driven by this need, this paper aims to present a unifying theoretical foundation for DT systems by formalizing their structure and behavior using system theory and hierarchical hybrid automata. The intended contribution is to move beyond fragmented definitions and offer a mathematical rigorous framework that can support modeling, symbolic manipulation, and real-time adaptation of DTs.

The remainder of the paper is organized as follows. Section 2 reviews state-of-the-art conceptual and formal approaches to DT. Section 3 introduces our theoretical framework to serve as a basis for the building of a DT theory. Section 4 introduces hierarchical hybrid automata to contribute to such a theory. Section 5 emphasizes the crucial role of the inference mechanism. Section 6 concludes the paper and discusses some current limitations.

2 STATE OF THE ART DIGITAL TWIN MODELS AND STANDARDS

Although the idea of DT has been around for decades, its real impact is only beginning to be fully recognized, especially with its application in various fields such as industry, health and smart cities. This expansion has led to a multitude of definitions without a clear consensus on its characteristics and applications, which can compromise the clarity of the concept and lead to an inefficient use of this technology. Several conceptual and formal modeling approaches of DTs have been proposed to clarify the concept and standardize its implementation.

The International Standardization Organization defines a DT in the context of manufacturing as a "Digital representation of a target entity with data connections that enable convergence between the physical and digital states at an appropriate rate of synchronization" (ISO/IEC 2023). Initially, Glaessgen and Stargel (2012) defined a DT as "an integrated multi-physics, 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". In a series of works centered around Product Lifecycle Management (PLM), the DT was conceptualized as the combination of a physical space, a virtual space and the connection between these two spaces (Grieves 2014). Later, Rosen et al. (2015) proposed a DT architecture focused on the integration of cyber-physical systems for manufacturing applications, emphasizing the importance of real-time feedback between physical and virtual space, allowing continuous process optimization. Grieves and Vickers (2016) defined a DT as "a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level". Negri et al. (2017) presented a reference framework for the implementation of DTs in Industry 4.0, underlining the ability of DTs to interact with manufacturing execution systems and provide real-time supervision. Schleich et al. (2017) showed that current approaches lack a solid conceptual basis, making it difficult to apply DTs to manufacturing engineering, and therefore proposed a reference model for DTs, based on an abstraction to capture and understand the physical twin, its behavior and its environment. Kritzinger et al. (2018) proposed a taxonomy distinguishing the terms Digital Twin, Digital Shadow and Digital Model, depending on the degree of connection between the physical system and the digital model (i.e., two-ways connection for Digital Twin, one-way connection for Digital Shadow, and no connection for Digital Model). Stark et al. (2017) proposed a three-level conceptual model, characterizing the DT as a single instance of a global digital model called Digital Master, associated with its individual Digital Shadow and an intelligent link between these two entities (algorithm, simulation model, etc.). In addition, Ponomarev et al. (2017) developed a five-layer conceptual model, including the cyber-physical layer, the data processing and storage layer, the distributed computing and storage layer, the model and algorithm layer, and the visualization and user interface layer. Tao et al. (2019) proposed a five-dimensional conceptual and formal model, called 5D model, extending Grieves' model (Grieves 2014) by adding digital twin services and connections. Redelinghuys et al. (2019) presented a six-layer conceptual model, focusing on the transmission of data flows from physical systems to the cloud. Fuller et al. (2020) conducted a literature review on DTs, highlighting the different definitions and applications, and proposing a framework to understand DTs according to their complexity and level of integration.

To further formalize the concept and practices related to DTs, several international standards have been published or are under development. The International Organization for Standardization standard (ISO 2020) provides a standardized framework for the development of DTs in an industrial setting, offering a coherent and structured architecture, while encouraging the reuse of common components and data in DTs. The recently published (ISO/IEC 2023) standard provides a conceptual framework for the development and use of DTs, describing a DT as a synchronized digital representation of a physical or virtual entity, thus facilitating the simulation, analysis and optimization of systems. It represents a major step forward in clarifying the terminologies and concepts associated with the DT, allowing for wider adoption in various sectors. Two standards provide additional perspectives: IEC/AWI TS 63526 (ISO/IEC 2023) which focuses on urban information modeling and DTs for cities, and aims to fill the normative gaps related to digital urbanism, and ISO 30186 (ISO 2020), which aims to establish maturity levels for DTs, providing a methodological framework to assess and improve their adoption and deployment.

Despite this very substantial work, there is no formal specification framework for a DT that allows its components to be described in mathematical form and manipulated symbolically (by algebraic structures, logical predicates, etc.). It is with this objective that we introduce the DMSμ framework.

3 GENERIC FORMAL APPROACH TO DIGITAL TWINS: THE DMSμ FRAMEWORK

Figure 1 illustrates the key entities of the DMSμ framework: the system of interest, the set of models developed to manage it, and the data-driven model inference unit.

The system of interest is operating in a given context (environmental conditions, physical constraints, etc.), with data being collected through sensors on the whole operational situation, including system's input, output and state, as well as contextual data. Models of various types are developed using various formalisms to enable the following services:

- Monitoring, i.e., continuous 2D/3D visualization of key indicators (making use of visualization models such as 3D/geometric models);
- Prediction, i.e., simulation and forecasting of key indicators (making use of models of the dynamics of the system of interest, such as multi-agent, system dynamics, and discrete-event models);
- Diagnosis, i.e., evaluation of current key indicators, and detection of abnormalities (making use of learning models such as Artificial Neural Networks, among others);
- Optimization, i.e., exploration of "what-if" scenarios and selection of the best option (making use of operation research models such as heuristics, metaheuristics and larger search algorithms); and
- Control, i.e., feed-back commands to automatically manage the system of interest (making use of control models such as Kalman filters).

Most often, these models are combined/coupled together to produce the DT services. The latter are also combined together to produce more advanced services. As an application example, consider each turbine of a power plant comprising thousands of components, each of which is subject to failure based on metallurgy, load, environment, and other factors. As it is impossible to pinpoint the proper time to take a turbine offline for maintenance using traditional mean time between failure estimates, the plant operators have no choice but to schedule expensive and often unnecessary maintenance cycles based on historical operating experience and the knowledge of a few expert employees, to avoid exposing the plant to the risk of unscheduled outages. By building a DT for every critical turbine assembly, enriching it with situational data such as system load, ambient temperature, and air quality, and continuously diagnosing it as well as simulating it under various assumptions, plant operators can bring turbines down for maintenance predictively, eliminating the costs of unnecessary downtime and mitigating the risks of unplanned outages. In such a case, predictive maintenance is an advanced service that has been obtained from the basic diagnosis and prediction services. The individual DTs can also be leveraged in the whole supply chain of the plant, such that the behavior and dynamics of the supply chain can be simulated based on energy consumption of the plant's customers. Exploration of what-if scenarios enables midterm/short-term decision-making towards optimizing the quality of production processes.

One of the main ideas behind the DT paradigm is that the system of interest is highly variable, which makes its models invalid over time. To maintain this validity, the DT updates these models using data collected on the operating system. The mechanism of assimilating these data to update the models is performed by the inference unit. We refer to "model updating" as the mechanism of modifying at least one specification element of an existing model (e.g., parameter value, initial/current state, etc.) in order to reflect changes detected in the system of interest thanks to the data collected on the system. In this way, re-calibration (or dynamic calibration), reset, and structural change are all forms of model updating. We refer to "model inference" as the way a model is deduced from data collected about the system. In this way, model updating is a special case of model inference, where the inferred model is obtained by updating an existing model. Another special case is the data-driven discovery/design of a new model from scratch.

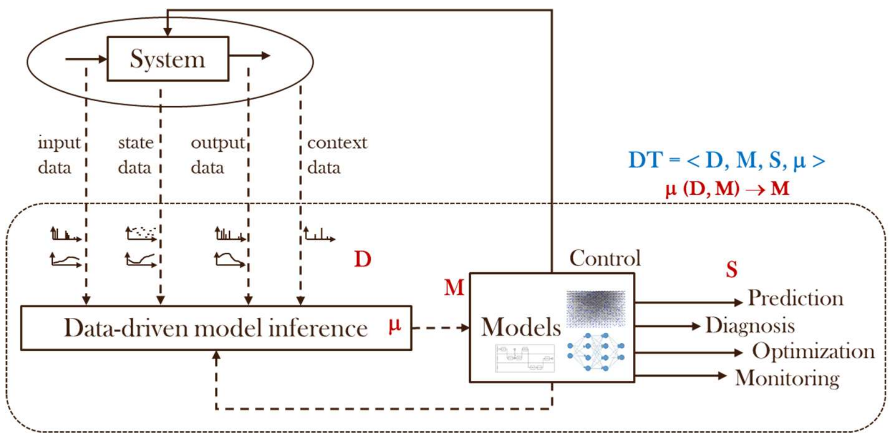

Figure 1: DMSμ framework.

Within this framework, we formally specify a DT system the following way:

$$Sys_{DT} = <D, M, S, \mu> \text{ where:}$$

- D is the set of collected data (including input data, state data, output data and context data);
- M is the set of models of the system of interest within the DT (including multi-agent/discrete-event/system dynamics models, learning/control/visualization models, and search algorithms);
- S is the set of services provided by the DT (including prediction, diagnosis, optimization, monitoring and control), each service making use of elements from M and/or D;
- $\mu: \Omega_D \times \mathscr{M} \rightarrow \mathscr{M}$ is the data assimilation function, where Ω_D is the set of all admissible data segments and $\mathscr{M}$ is the set of all possible models. The detailed specification of the assimilation function requires both a specification of Ω_D's elements and the detailed specification of elements of $\mathscr{M}$, such that they can be symbolically manipulated.

4 HIERARCHICAL HYBRID AUTOMATA FOR DIGITAL TWINS SPECIFICATION

4.1 DT Models' Specification

We need a specification approach that allow details of the model structure and behavior be accessible and modifiable through symbolic manipulation. For this reason, we turn to hybrid automata (which elements are visually presented in Figure 2) as our unifying specification formalism. Such automata are hybrid in the sense that they enable capturing both the discrete and continuous aspects of a dynamic system. We formally specify all dynamic systems as the following 7-tuple:

$$Sys = \langle \Lambda, X, Y, \Phi, \Sigma, \Delta, \Delta^* \rangle, \text{ where:}$$

- Λ is the parameter set, which models the assumptions behind the design context of the DT model.
- X (respectively Y) is the input (respectively output) set, which models the influences received from (respectively exerted on) the DT environment.
- Φ is the phase set; it models the stable discrete/continuous steps of the DT model. Each phase is defined by its properties, as well as the amount of time that expires before it changes to another phase (via an internal transition), and the activities performed during the phase.
- $\forall \varphi \in \Phi, \Phi_\varphi = \langle \pi_\varphi, \theta_\varphi, \tau_\varphi \rangle$ is the phase's definition, with:
  → π_φ is the phase invariant, a predicate on Σ ($\in \wp(\Sigma) \cup \{NIL\}$) giving the semantics of π in Σ;
  → θ_φ is the phase activity, a predicate on Σ modeling the computations performed during φ;
  → τ_φ is the phase's lifetime, a delay ($\in \mathbb{R}_+$) before the model changes from φ to another phase;
- Σ is the semantic domain, i.e. the vector of variables, which the phases are mapped onto;
- $\Delta = \{\Delta_{i?}^{j}, \Delta_{i}^{j!}, \Delta_{i?}^{j!}, i \in \Phi, j \in \Phi\}$ is the phase-to-phase transition set. $\Delta_{i?}^{j}$, $\Delta_{i}^{j!}$, and $\Delta_{i?}^{j!}$ are respectively external, internal, and confluent transitions, from phase to phase (an internal transition

occurs when the lifetime of the current phase elapses; an external transition is triggered by the receipt of an input; a confluent transition occurs when both internal and external transition conditions occur simultaneously), with:

→ $\Delta_{i?}^{j} = \langle \omega_{i?,j}, \varepsilon_{i?,j}, \theta_{i?,j} \rangle$ where $\omega_{i?,j} \in \wp(X)$ is the condition of input receipt, $\varepsilon_{i?,j} \in \wp(\Sigma) \cup \{NIL\}$ is the transition guard, and $\theta_{i?,j} \in \wp(\Sigma) \cup \{NIL\}$ is the jump (i.e., action performed);

→ $\Delta_{i}^{j!} = \langle \varepsilon_{i,j!}, \rho_{i,j!}, \theta_{i,j!} \rangle$ where $\varepsilon_{i,j!} \in \wp(\Sigma) \cup \{NIL\}$ is the transition guard, $\rho_{i,j!} \in \wp(Y) \cup \{NIL\}$ is the output sending predicate, and $\theta_{i,j!} \in \wp(\Sigma) \cup \{NIL\}$ is the transition jump;

→ $\Delta_{i?}^{j!} = \langle \omega_{i?,j!}, \varepsilon_{i?,j!}, \rho_{i?,j!}, \theta_{i?,j!} \rangle$ where $\omega_{i?,j!} \in \wp(X)$ is the condition of input receipt, $\varepsilon_{i?,j!} \in \wp(\Sigma) \cup \{NIL\}$ is the transition guard, $\rho_{i?,j!} \in \wp(Y) \cup \{NIL\}$ is the output predicate, and $\theta_{i?,j!} \in \wp(\Sigma) \cup \{NIL\}$ is the transition jump;

→ $\Delta^{*} = \langle i^{*}, \theta^{*}, \tau^{*} \rangle$ is the initialization, where i^{*} is the initial phase, θ^{*} is the initial jump, and τ^{*} is the time already elapsed in the initial phase.

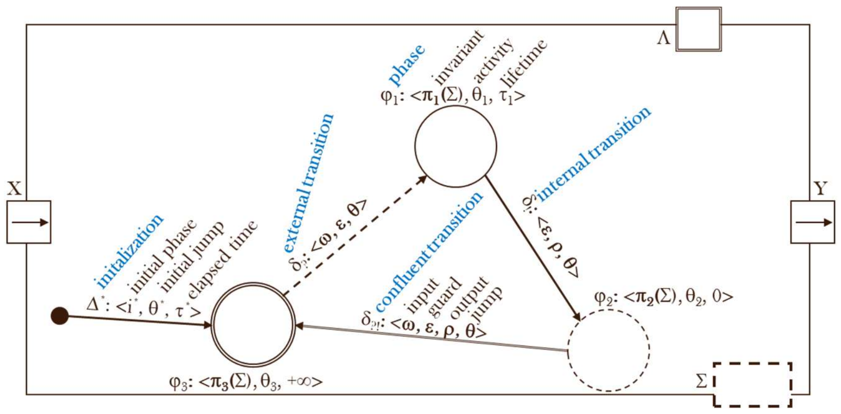

Figure 2: Hybrid automata for DT model's specification.

This formal approach enables the identification of different levels of self-updating capabilities, i.e., the ability of the μ function to detect and respond to changes, either in values (low level of knowledge) or in structure (high level of knowledge) of specific elements within the formal specification of each model in M (Diakité and Traoré 2024), using statistical inference estimation techniques (such as Data fitting, Kalman Filter, etc.). The identified self-updating capabilities include self-initializing (where Δ^{*} is updated), self-coding (where Σ is updated), self-aligning (where Φ is updated), self-interfacing (where X/Y are updated), self-regulating (where Δ is updated), and self-calibrating (where Λ is updated) DTs.

4.2 DT Services' Specification

To formally specify DT services, we adopt the frame paradigm introduced with DEVS (Discrete EVent System specification) by (Zeigler 1976), and later extended by the experimental frame methodology (Traoré and Muzy 2006), which is shown at the left-hand side of Figure 3. In such a paradigm, first class entities are the source system, the model of that system (Md), the experimental frame (EF), and the simulator. The source system (i.e., the system under study) is observed within a given context, leading to the design of a context-based Md and the modeling of its EF. The source system acts as a source of behavioral data. The context is the set of conditions under which the system is observed. Md is a set of rules or mathematical equations that give an abstract representation of the system, which is used to replicate its behavior. Equally, EF, which gives an abstract representation of the context, is a model component to be coupled with Md to produce the data of interest under specified conditions. EF also implicitly expresses the conditions under which Md is considered as a valid representation of the system of interest. The simulator is the automaton that is able to execute the instructions of the resulting coupled model (Md coupled with EF).

In our case (right-hand side of Figure 3), the model (DTM) once built is continually updated with data collected from the system of interest, which we call the Twin of Interest (ToI) as a way to emphasize the twinning situation between the system and its model. Hence, a model inference mechanism is needed to allow such a regular update of DTM, in order to preserve its validity within the same context (therefore realizing the μ function of the DMSμ framework). DTM is parameterized in order to capture both the ToI's properties and the ToI's operational context. Then, the DT service (DTS) describes what is expected from the DT using the DT model, the same way EF describes how experiments are conducted and results are derived (using generators, acceptors and transducers) to provide prediction using Md (Zeigler 1976).

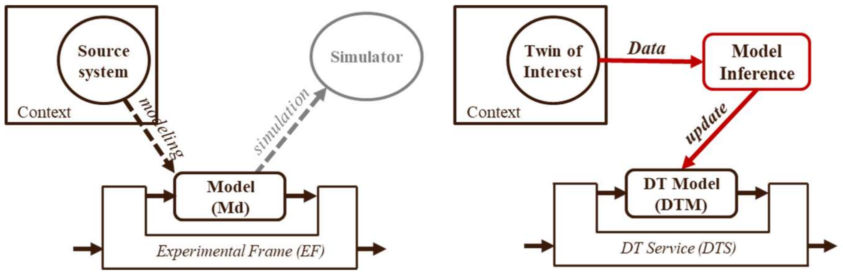

Figure 3: From DEVS paradigm (Zeigler 1976) to DT theory.

It's noteworthy that DTM and DTS both capture the behavior of a dynamic system. Hence, both are formally specified using hybrid automata. More generally, equation (2) is used to specify all elements of M and S of equation (1), as well to specify the whole system defined by equation (1) by coupling elements of M with elements of S. Coupling hybrid automata leads to hierarchical hybrid automata, as depicted by Figure 4. The left-hand side of Figure 4 shows how DTM and DTS feed each other (the same way EF and Md do) to enable DTS provide the expected DT service, using the DT model. The middle of Figure 4 shows the hierarchical nature of such a coupling, with DTM and DTS being children of Sys_{DT} (the resulting DT system). The hybrid automata at the right-hand side of Figure 4 captures this hierarchy, with a semantic domain including DTM and DTS as variables, and phases which invariants capture connections between these components (i.e., the way they feed each other). That way, a phase transition within Sys_{DT} can be used to translate a dynamic change of connections between its components. Thus, the hierarchical hybrid automata can capture dynamic structure situations (notice that in case the structure of the DT system doesn't vary, then Sys_{DT} will possess a single phase with an infinite lifetime, and an external transition from this phase to itself with a jump realizing the μ function).

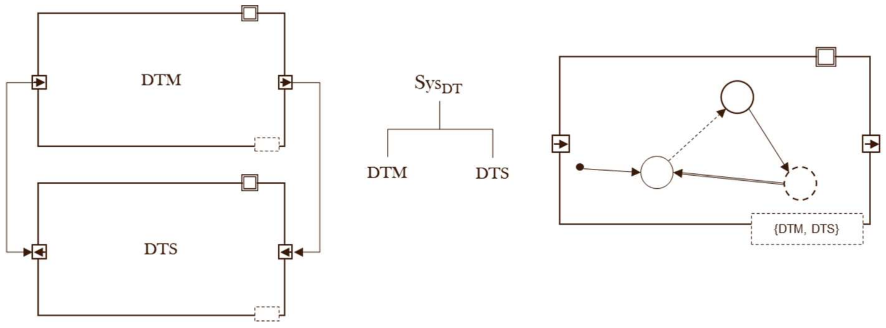

Figure 4: Hierarchical hybrid automata for DT system specification.

It is equally noteworthy that not only one service may be provided by the DT system. A DT system can provide many services, potentially making use of (the same or) different models, as shown in Figure 5. As already explained, various models (e.g., 2D/3D/geometric models, discrete-event/agent/system dynamics models, Bayesian inference models, heuristics/learning models, and feed-back control models) may be developed (and sometimes combined) to enable various services (monitoring, prediction, diagnosis, optimization, and control).

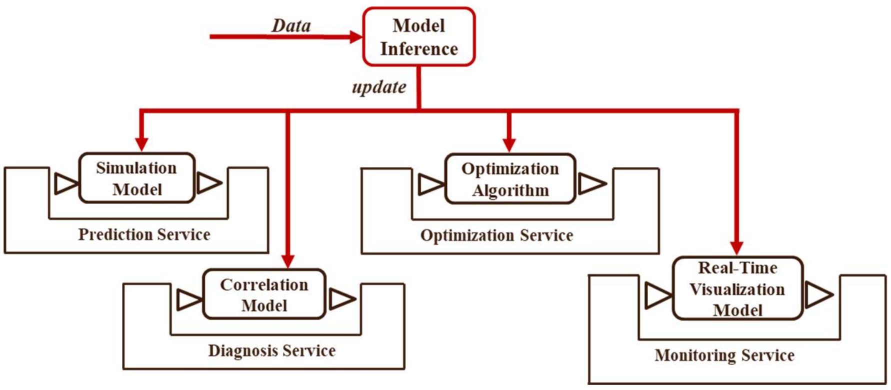

Figure 5: Multi-service multi-model DT system.

4.3 DT Data Specification

Data synchronization between a DT and its corresponding real system is a key feature of the DT concept. This synchronization can be either clock-based or event-based (Barricelli et al. 2020). Clock-based synchronization relies on predefined time intervals to update the DT. It can be real-time (e.g., every millisecond) for applications requiring high responsiveness or cyclic (e.g., hourly) for applications less sensitive to timing. Event-based synchronization is triggered by a change in system state or a specific condition, such as exceeding a temperature threshold. This method is especially useful for applications where the update frequency depends on unexpected events or critical state changes.

We formally capture each element of D as a data segment stamped by a date (i.e., either an interval of time or an instantaneous moment); hence, for each $\omega \in \Omega_D$, we have:

$$\omega: T_U \rightarrow D, \text{ where}$$

- T_U represents the universal/wall-clock time, and $\mathrm{Dom}(\omega) \subset TU$, and $\mathrm{Im}(\omega) \subset D$;
- D is an abstract set with values taken in a set that depends on the unit considered for the study.
- The internal composition law $\oplus$ is defined on Ω_D such that if $\mathrm{dom}(\omega1) \cap \mathrm{dom}(\omega2) = \emptyset$, then $\omega1 \oplus \omega2 = \omega1$ over $\mathrm{dom}(\omega1)$ and $\omega2$ over $\mathrm{dom}(\omega2)$.

5 MODEL INFERENCE

The inference mechanism (the μ function) plays a crucial role in transparently adapting data, allowing to dynamically detect changes in the real system and update the DT model accordingly. A typical way to model the DT system and to capture the inference mechanism is shown by Figure 6. The DT system exhibits two types of input: the ones from users requesting for the DT service, and the ones from sensors and other data sources collecting information (data segments) on the ToI. Users requests are directed to the DTS component inside the DT system, which answers are forwarded to the output of the DT system; this corresponds to the DT in use, where DTS coupled properly with DTM derives the service provided. Meanwhile, data segments coming from the ToI may cause the DTM component to be updated. To handle this, two phases are considered for the DT system: waiting (with infinite lifetime) and memorizing (with zero lifetime). Both phases' invariants carry the DTM-DTS coupling information as already explained. While the waiting phase has no activity, the memorizing phase filters a dataset from the data lake (i.e., the set of all data collected from the ToI from its existence), to be used for the DTM update. When a data segment is received, the DT system transitions from waiting to memorizing, and the jump performed during this transition consists in adding the data segment to the data lake. After the dataset is filtered, the DT system transitions back to the waiting phase, while updating the DTM during the corresponding jump, using the inference function defined. DTM, DTS, data lake and dataset are all variables of the DT system, therefore they all belongs to its semantic domain.

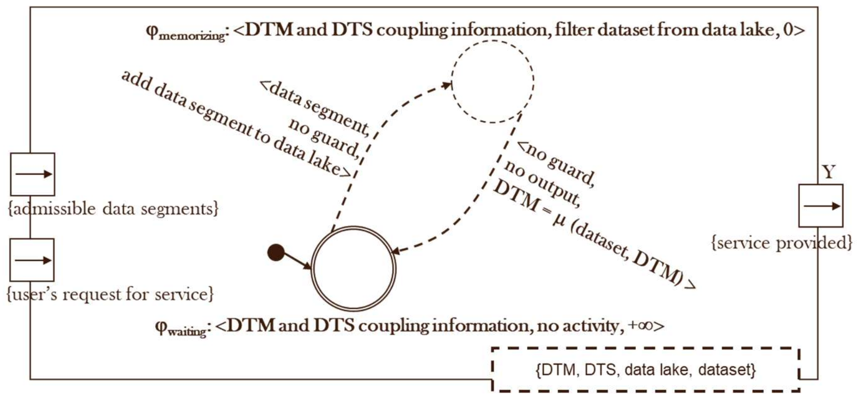
φmemorizing: <DTM and DTS coupling information, filter dataset from data lake, 0>
<data segment, no guard, add data segment to data lake>
<no guard, no output, DTM = μ (dataset, DTM)>
{admissible data segments}
{user's request for service}
Y
{service provided}
φwaiting: <DTM and DTS coupling information, no activity, +∞>
{DTM, DTS, data lake, dataset}

Figure 6: Typical inference mechanism in a DT system.

The inference mechanism can operate at two temporal scales: online and offline. Online inference consists in updating while DTM is being executed. Depending on the type of model, the complexity of the implementation can vary. For example, in the case of a visualization model for a real-time monitoring service, this update is relatively simple. However, for a DT model used for prediction services, the situation becomes more complex, because the simulation time does not always coincide with wall-clock time. When data is received during the execution of the simulation model, it is necessary to manage the gap between the simulated time and the wall-clock time, which can sometimes lead to backtracking in simulation, as done in optimistic Parallel and Distributed Simulation (Fujimoto 2001), in order to synchronize the DT model with events happening in the present of the ToI but in the past of the DT. Offline inference consists in updating the model before the start of an experiment (in other words, between two consecutive experiments). This is to ensure that the DT faithfully reflects the ToI before launching a DT service.

The inference mechanism can also take different forms: Direct mapping, Indirect mapping, and Learning mapping. Direct mapping is used when the captured data directly reflects a state of the system. Indirect mapping occurs when the collected data does not allow the state of the system to be directly deduced. This type of update is often based on estimation techniques. Several statistical estimation techniques exist (Hilbe and Robinson 2013), some of which are very popular, such as Bayesian inference (a method for determining the distribution law of a variable), Data fitting (a type of Bayesian inference in which several distributions are determined, and a distance measure is defined to choose the one deemed closest to the user's needs), and Kalman filters (another type of Bayesian inference that recursively estimates the state of a system at a given time from its previous states, which may come from a series of incomplete or noisy measurements). The use of Learning mapping is particularly relevant when significant changes occur in the system, requiring in-depth knowledge and high detection capacity. For example, if sensor data differs radically from expected data, it may be necessary to use advanced techniques to adjust the model. This form of mapping is often used to manage structural changes, where simple direct or indirect deduction methods are no longer sufficient. In these situations, machine/deep learning algorithms are used to analyze the data and update the model accordingly. These algorithms make it possible to identify complex patterns and learn new relationships within the data to effectively update the DT model.

In all mapping cases, the formal specification that we propose makes it possible to symbolically manipulate the various specification elements of the DT by the aforementioned methods, and to modify their values as needed.

6 CONCLUSION

The concept of DT has emerged as a transformative paradigm in various industries, offering enhanced system modeling and optimization through real-time synchronization with physical systems. Despite its growing popularity, the definition and application of DTs remain varied and sometimes ambiguous, which hinders their broader implementation and effective utilization. To address this, the paper proposes a system-theoretic framework to formalize the DT concept and build a cohesive theory that can be used across different domains.

The framework, called DMSμ, combines data-driven model inference with regular synchronization and updates, allowing the DT model to adapt to changes in the system of interest over time. It also highlights the importance of conceptual clarity and mathematical specification, proposing the use of hierarchical hybrid automata as a unifying formalism to support symbolic manipulation of DT components. Ultimately, this approach provides a foundation for further theoretical development and the standardization of DT technology, enabling more efficient and adaptive use of DTs across a wide range of applications. By formalizing the mechanisms behind model updating and inference, it paves the way for a deeper understanding of the DT concept and its practical deployment in real-world systems.

However, at this stage, let's emphasize two major limitations (among many) to our on-going effort towards a theoretical DT framework. First, the applicability of the framework (and hence, its acceptance) in industrial environments requires a detailed description and illustration of DMSμ in a practical large-scale case study, showing how to specify and implement the sets D, M and S using hybrid automata, as well as how to define the function μ. An example of such a demonstrating real-world application is the smart city DT we're developing in the ACT project (ANR-20-IDES-0001), which complete report is still to be issued. Secondly, we have not addressed here the computational efficiency involved updating a DT model; however, it's important to notice that this is key to the operationality of the DT (e.g., in the case of a DT system offering a control service, real-time constraints may impose the DT model be updated as fast as possible so that the response time of the DT service meets some security requirements). Mitigating these limitations is part of the immediate next steps in our research.

REFERENCES

Ali, Z., R. Biglari, J. Denil, J. Mertens, M. Poursoltan, and M. K. Traoré. 2024. "From Modeling and Simulation to Digital Twin: Evolution or Revolution?". *Simulation* 100(7):751-769.

Barricelli, B. R., E. Casiraghi, and D. Fogli. 2020. "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications". *IEEE Access* 7:167653-167671.

Budiardjo, A. and D. Migliori. 2021. "Digital Twin System Interoperability Framework". Technical Report, Digital Twin Consortium, East Lansing, Michigan.

Diakité, M. and M. K. Traoré. 2024. "Formalizing a Framework of Inference Capabilities for Digital Twin Engineering". *Simulation* 100(9):887-902.

Fujimoto, R. M. 2001. "Parallel and Distributed Simulation Systems". In *2001 Winter Simulation Conference (WSC)*, 147-157.

Fuller, A., Z. Fan, C. Day, and C. Barlow. 2020. "Digital Twin: Enabling Technologies, Challenges and Open Research". *IEEE Access* 8:108952-108971.

Glaessgen E., Stargel D. 2012. "The Digital Twin Paradigm for Future NASA and US Air Force Vehicles". In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1818.

Grieves, M. 2014. "Digital Twin: Manufacturing Excellence Through Virtual Factory Replication". *White Paper* 1:1-7.

Grieves, M. and J. Vickers. 2016. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". In *Trans-Disciplinary Perspectives on System Complexity*, edited by F.-J. Kahlen, S. Flumerfelt and A. Alves, 85-114, Switzerland: Springer.

Hilbe, J. M. and A. P. Robinson. 2013. *Methods of Statistical Model Estimation*. Florida: CRC Press.

ISO 2020. "Automation Systems and Integration – Digital Twin Framework for Manufacturing – Part 1: Overview and General Principles". International Organization for Standardization, Geneva, Switzerland.

ISO/IEC 2023. "Digital Twin – Concepts and Terminology". International Organization for Standardization, Geneva, Switzerland.

Kritzinger, W., M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. "Digital Twin in Manufacturing: A Categorical Literature Review and Classification". *IFAC-PapersOnLine* 51(11):1016–1022.

Negri, E., L. Fumagalli, and M. Macchi. 2017. "A Review of the Roles of Digital Twin in CPS Based Production Systems". *Procedia Manufacturing* 11:939–948.

Ponomarev, S., A. Smirnov, A. Kashevnik, and N. Shilov. 2017. "Smart Product Digital Twin: Conceptual Model and Use Case in the Context of Manufacturing as a Service". In *10th International Conference on Social Computing and Social Media*, 391-401 https://doi.org/10.1007/978-3-319-58559-834.

Redelinghuys, A. J., A. H. Basson, and K. Kruger. 2019. "A Six-Layer Digital Twin Architecture for a Manufacturing Cell". *Procedia CIRP* 81:369-374 https://doi.org/10.1016/j.procir.2019.03.058.

Rosen, R., G. Wichert, G. Lo, and K. D. Bettenhausen. 2015. "About the Importance of Autonomy and Digital Twins for the Future of Manufacturing". *IFAC-PapersOnLine* 48(3):567–572.

Schleich, B., N. Anwer, L. Mathieu, and S. Wartzack. 2017. "Shaping the Digital Twin for Design and Production Engineering". *CIRP Annals* 66(1):141-144.

Stark, R., C. Fresemann, and K. Lindow. 2017. "Development and Operation of Digital Twins for Technical Systems and Services". *CIRP Annals* 66(1):129-132 https://doi.org/10.1016/j.cirp.2017.04.040.

Tao, F., J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. 2019. "Five-Dimension Digital Twin Model and Its Ten Applications". *Computer Integrated Manufacturing Systems* 25(1):1–18 https://doi.org/10.13196/j.cims.2019.01.001.

Traoré, M. K. and A. Muzy. 2006. "Capturing the Dual Relationship Between Simulation Models and their Context". *Simulation Modelling Practice and Theory* 14(2):126-142.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation*. New York: Wiley-Interscience.

Zeigler, B. P., S. Mittal, and M. K. Traore. 2018. "MBSE with/out Simulation: State of the Art and Way Forward". *Systems* 6(4):40.

AUTHOR BIOGRAPHY

MAMADOU KABA TRAORE is a Professor at the University of Bordeaux in France. His research interests include hybrid modeling and simulation, digital twin engineering, and digital twins for system engineering, with applications in industrial systems, supply chains, smart cities, and healthcare systems. His email address is mamadou-kaba.traore@u-bordeaux.fr and his website is https://www.ims-bordeaux.fr/researchers-and-publications/traore-mamadou/.