A Service-Oriented Digital Twin Framework for Dynamic and Robust Distributed Systems

Rong Gu[1], Tiberiu Seceleanu[1], Ning Xiong[1], Muhammad Naeem[1]
[1] Mälardalen University, Västerås, Sweden
{rong.gu, tiberiu.seceleanu, ning.xiong, muhammad.naeem}@mdu.se

Abstract—**Digital Twins (DTs) are virtual representations of physical products in many dimensions, such as geometry and behaviour. As a backbone of Industry 4.0, DTs help interpret and even predict the behaviour of physical processes, provide a virtual testbed for maintenance and upgrade, and enable automatic decision-making supported by artificial intelligence. Despite the promising future, challenges exist, such as the absence of a framework that facilitates the development and application of DTs in industrial contexts. We propose a service-oriented architecture (SOA) DT framework for dynamic and robust distributed systems. The framework contains two types of services. One includes the services provided to the users and is supported by an orchestration mechanism to ensure a quality of service (QoS). The other one refers to the common functions of all DTs. Further, we describe the DT-based decision-making enabled by our QoS-oriented learning of the framework and a Hoare-logic-based verification of QoS.**

Index Terms—**Digital Twins, SOA, System design, Machine learning, Formal verification.**

I. Introduction

As the essential step towards Industry 4.0 (I4.0), digitalization in industries is largely involved in various scenarios, such as automation, data exchange, cloud computing, robotics, Artificial Intelligence (AI) / machine learning (ML), and cyber-physical systems. However, it is very challenging for the modern industry to adapt to I4.0 contexts, which cover the whole lifecycle of products. For example, in the inclusion of legacy systems and technologies, finding optimal deployment solutions, and achieving overall performance and robustness of complex systems, problems regarding the enormous amount of data generated and decision-making in crucial yet complex situations hinder the development of digitalization.

One possible solution to address those challenges is the virtual representation of physical products, which is also known as Digital Twins (DTs) [1]. Hence, DTs now bear the promise of enabling I4.0 initiatives and have been applied in many fields as conceptual models for I4.0, such as smart cities [2] and healthcare [3]. Essentially, DTs simulate a real system of interest (SoI) digitally in multiple dimensions, e.g., geometry and behaviour, such that they capture the attributes and functions of the SoI accurately enough for specific purposes, e.g., communication, system interpretation and maintenance. Hence, fidelity is a crucial property of DTs, which requires techniques for verification and validation (V&V) of DTs. However, the inherent complexity of DTs makes the V&V of such models extremely difficult and thus brings a usability barrier in the industry [4]. One of the obstacles to realizing verifiable DTs is the "barely emerging" standards of DTs [5].

Previously [6], a 3-layered approach to an intended domain agnostic framework for the creation, development and operation of DT systems was proposed, called *D-RODS* (A Digital Twin Framework for Dynamic and Robust Distributed Systems). *D-RODS* addresses problems related to system integration, performance, organization, data volume and quality, and challenges of distributed system automation: integration and compatibility of legacy systems, continuous improvement, lack of skilled labour, etc. *D-RODS* aims to advance the level of digitalization towards autonomous operations, validated via use cases coming from major Swedish companies in the domains of industrial automation, transportation and telecommunication.

In this paper, we progress by proposing a Service-Oriented Architecture (SOA) framework for *D-RODS*. The reason for adopting SOA is twofold. Firstly, DTs are often provided alongside the actual system of a product, such as the wind turbines of ABB or the base radio stations of Ericsson. When a complex system or a system of systems is composed of products from different manufacturers, having DTs of the sub-systems as services offers a clear separation between the service providers and consumers, thus creating a separation of concerns. This advantage of SOA is highly demanded for I4.0. Moreover, one of our goals is to provide complex DT systems composed of a potentially large number of sub-system DTs. The SOA approach comes, as intended and proven along years of development and employment, as the best solution. Secondly, auxiliary functions, e.g., monitors of the DTs, may not necessarily run consistently throughout the life-cycle of DTs. Therefore, it is efficient and environmentally friendly to have these functions as services that are invoked only when needed. Additionally, some support functions for DT operation, such as the verification module, can be reused with various DTs and thus not attaching them to particular DTs is a wise design choice.

In a nutshell, in this paper, we report the current progress and envision a SOA framework of *D-RODS*. We also intend to answer to a few research questions that we consider critical in our approach, identified as follows.

- To what extent is it possible to build a DT system that correctly matches the original?
- What solutions can be employed to alleviate the inherent inexact predictions of ML algorithms?

Our path towards finding answers to the above questions starts with the proposal of a service orchestration mechanism that is aware of the quality of service (QoS) demanded by the users. Further, we explore how ML and formal verification can be adopted for generating and verifying QoS-aware service orchestration, respectively.

The remainder of this paper is organized as follows. Section II gives the necessary background knowledge for reading this paper. Section III gives the related studies that inspired our research. Section IV presents the challenges of developing a DT framework before Section V describes our current status and a roadmap for developing methods that overcome those challenges. Section VI show the use case where we apply our methods. Section VII discusses the current limits that we have observed in *D-RODS*. Section VIII concludes that paper and envisions the future work.

II. Preliminaries

A. Digital Twins

DTs are cutting-edge technology revolutionizing industries worldwide [7]. They are virtual representations of physical objects, processes, or systems that enable real-time monitoring, analysis, and optimization. By seamlessly integrating the physical and virtual realms, DTs provide a powerful platform for simulation, decision-making, and control. Their practical applications span industries such as manufacturing, healthcare, transportation, and energy, showcasing their transformative potential yet to be fully explored.

DTs are rooted in the convergence of advanced technologies like IoT, sensor tech, big data analytics, and simulation tools. They consist of five key dimensions: the physical entity, the virtual entity, services for both, and interconnections among components [8]. However, the data, the core element, truly fuels insights and innovation, driving efficiency and productivity enhancements in engineered systems.

While DTs have a relatively short history, significant advancements since the last decade have propelled them to widespread adoption and application across industries [7]. As they evolve rapidly, addressing challenges such as unified modelling methods and cyber-physical fusion is essential to fully harness their practical utility and benefits in various sectors.

B. Service-Oriented Architecture

Service-oriented architecture (SOA) is defined as a paradigm for the realization and maintenance of processes that span large distributed systems [9]. There are three major concepts in SOA, namely services, interoperability, and loose coupling. A service is a self-contained function/act that is provided by one party to another. The function/act might be connected to a physical process but the service essentially refers to an intangible performance. Interoperability is a characteristic of a product whose interface is completely understandable and accessible by other products without any restrictions. Loose coupling is simple as its name indicates, a means of component composition that requires low dependency on each other.

In this paper, we propose two types of services, namely, services supported by DTs and services for DTs. The former are functions provided to users of *D-RODS* and supported by DTs via an orchestration mechanism, which adaptively invokes different DTs or physical processes to guarantee a desired level of quality of service (QoS). The latter are common functions that can be invoked by DTs, such as verification and validation of DTs. We aim to design an SOA framework for our DTs, where DTs communicate with each other via an infrastructure such that they can be easily configured and replaced throughout the life cycle of *D-RODS*. The framework supports verifying the QoS of *D-RODS* under various assumptions and offering suggestions for system operation when changes occur.

III. Related Work

Digital twins and Service-Oriented Architectures are naturally good partners as the data captured by DTs can be presented, analysed, and visualized by services or microservices with adaptability. The combination of DTs and SOAs has been carried out in three ways: i) DTs separated from services [10], ii) DTs and services are integrated [11], [12], and iii) hybrid, meaning the services can be either provided by DTs or SOAs [13] [14].

When DTs and services are separated, DTs are constructed individually and services are functions that are provided to the users of the DTs via a dashboard. If these functions are closely coupled with DTs, then the separated DTs and services introduce overhead, and thus an integrated combination is proposed, where services are contained in the DTs such that they share the memory and resources and the QoS is supposed to be increased. However, the integrated DTs and services bring high maintenance costs as an update or upgrade of DTs would result in a significant change in the services that are exposed to the users, which are unnecessary in some cases, such as general services that mostly require data or resources outside DTs. Hence, the hybrid way of combining DTs and SOAs is proposed. Our method belongs to the hybrid combination as we have services exposed to users (i.e., the user interface in Figure 2) and services that are common functions among DTs. Additionally, our method provides QoS and the ability of real-time operation, which are not considered in any of the current frameworks.

IV. Challenges

We have faced several challenges while trying to improve the performance of our service-oriented DT system. We interpret here the DT *performance*, as the accuracy with which the execution of the DT system follows the execution of the actual system, in terms of event sequence and respective signal values. A certain tolerance is thought to be observed.

The following identified challenges are unique to our context but provide valuable insights into the broader considerations in developing and deploying DT systems.

A. Aggregation with separated data records

DT systems often accumulate records representing diverse operational conditions, resulting in fragmented data sets. These records are generated from sequential executions under varying conditions and do not form a continuous time series. Instead, they manifest as distinct groups of individual time series. This fragmented nature of data presents a significant obstacle to precise model construction, requiring innovative approaches for data aggregation. One possible approach to overcome this challenge is sequential model learning using data records generated from successive system executions. However, this method has a downside: it introduces the risk of overwriting valuable data from prior executions while accommodating new data.

B. Regression with Sparse Target Values

Regression tasks involve building models for predicting system behaviour using neural network architectures, which are expected to implement a nonlinear, continuous and smooth mapping between inputs and outputs. However, achieving smooth behaviour in regression models becomes challenging when the training data are limited by only containing sparse target values. The lack of sufficient target values makes it difficult to find model parameters that can accurately capture the underlying behavior patterns under a variety of situations.

C. Determining Optimal Time Delays

A time delay refers to the number of sampling periods that one traces back when predicting process outputs. Setting the appropriate values for inputs and outputs in DT models is essential to ensure effective learning. However, determining this can be hard, since assuming a short delay may cause the model to miss crucial previous information while using excessively long delays can introduce irrelevant information and unnecessarily complicat the model. Overcoming this challenge requires a comprehensive understanding of the system's dynamics and domain expertise to identify the optimal time delays that will facilitate effective model learning.

D. Heterogeneous Digital Twins

DTs of a system of systems, such as those considered in *D-RODS*, are heterogeneous because they have different dynamics and interfaces, and are even manufactured by different companies. Composing these DTs into one system and maintaining them regularly is a taunting task as a change in one component could influence other components and thus change the entire system's behaviour. One solution is to treat DTs as services and loosely couple them via an infrastructure as what it is in *D-RODS*. However, the overhead caused by the coupling and communication among services may drop the quality of service and a suitable orchestration mechanism is needed in a service-oriented DT framework.

Next, we propose the *D-RODS* approach to overcome these challenges. Note that the approach is still under development. Therefore, some innovations are conceptual. However, we present the ideas here to describe the research roadmap and inspire researchers who are facing similar challenges.

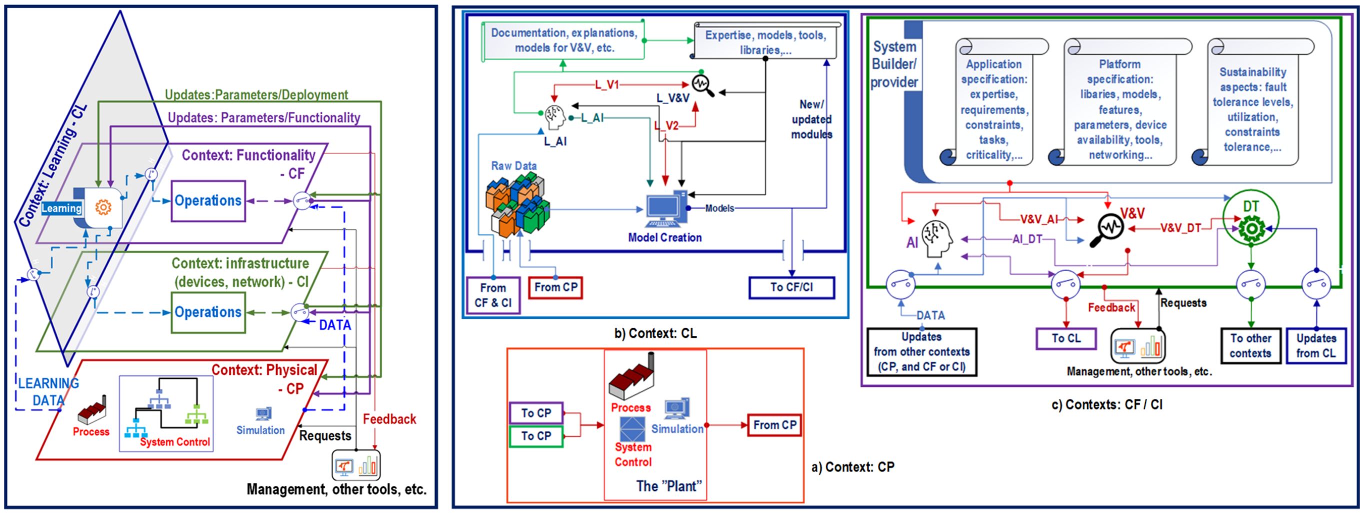
V. The *D-RODS* Approach

A. Approach Description

D-RODS aims to provide verified and verifiable AI-based approaches for constructing DTs. The approaches must be adapted to the size and features of system instances, continuously evolving through the operational stages, and improving with respect to their targeted goals. The architecture of *D-RODS* is organized on several *contexts* or *layers*, as depicted in Figure 1 A., and briefly described as follows.

Context: Physical (CP). This layer corresponds to the physical world (Figure 1 B.a), containing the plant and especially focusing on the processes within the system controlling it.

Context: Learning (CL). This layer (Figure 1 B.b) focuses on the creation of the DT models corresponding to the relevant parts of the other layers. Unsupervised learning, validated


Context: Learning - CL
Updates:Parameters/Deployment
Updates: Parameters/Functionality
Context: Functionality - CF
Learning
Operations
Context: infrastructure (devices, network) - CI
Operations
DATA
LEARNING DATA
Context: Physical - CP
Process
System Control
Simulation
Feedback
Requests
Management, other tools, etc.
Documentation, explanations, models for V&V, etc.
Expertise, models, tools, libraries,...
L_V1
L_V&V
L_AI
L_V2
L_AI
New/ updated modules
Raw Data
Models
Model Creation
From CF & CI
From CP
To CF/CI
b) Context: CL
To CP
To CP
Process
Simulation
System Control
The "Plant"
From CP
a) Context: CP
System Builder/ provider
Application specification: expertise, requirements, constraints, tasks, criticality,...
Platform specification: libaries, models, features, parameters, device availability, tools, networking...
Sustainability aspects: fault tolerance levels, utilization, constraints tolerance,...
DT
V&V
V&V_AI
V&V_DT
AI
AI_DT
DATA
Updates from other contexts (CP, and CF or CI)
Feedback
Requests
To CL
Management, other tools, etc.
To other contexts
Updates from CL
c) Contexts: CF / CI

A. The D-RODS layered perspective

B. The D-RODS contexts (layers) and respective connectivity

Figure 1. The *D-RODS* approach.

by suitable methods for verification and validation (V&V) extracts a filtered data set from a long data history. Our goal is to have the learning process and results transparent, explainable, and understandable, which would make the solution trustworthy for the industry. Based on domain expertise and models from libraries, this layer creates the set of DT models to be employed in other contexts introduced in the following paragraph. Once a complete version of the models is accepted, the **CL** goes offline, for a reduction in energy consumption.

Context: Functionality (CF) and **Context: Infrastructure (CI).** These two layers are similar (Figure 1 B.c) as they both contain an AI and a V&V block, which control and enhance the functionality of a complex DT (models from **CL**). A collection of AI algorithms and V&V procedures supervises the DT execution. Continuous learning, optimization and behaviour evaluation are in place. Focusing on machine-learning methods, *D-RODS* is set to refine system performance prediction and resource utilization optimization.

Apparently, *D-RODS* is a complex system of systems, in which sub-systems are coupled loosely and communicate asynchronously. The purpose of DTs is to provide various functions to users at the endpoint, such as monitoring and diagnosing the system of interest (SoI). Hence, we design a SOA framework that provides those functions as services to the users of *D-RODS* and treat modules in DTs as services such that they can communicate in various manners and be reused by multiple DTs. In the next section, we introduce the SOA framework in detail.

B. A SOA Framework of D-RODS

Figure 2 depicts the SOA framework of *D-RODS*, where functions at the user interface and common functions at the backend are implemented as services. Note that we only enumerate a part of the services as our framework allows an extension of services at any point in its life cycle. First, we go through the modules of the framework.

- *User interface*: an interface to users that contains services supported by the DTs, such as monitoring the SoI via DTs, reports of system execution, diagnosis of the SoI, and decision-making based on the status of DTs.
- *Digital twins*: virtual representation of real products, which are constructed by **CL** of *D-RODS*.
- *Common functions*: services for DTs including the **CF** and **CI** of *D-RODS*.
- *Plant*: The real products including the physical processes.
- *Orchestrator 1*: A module that performs the execution and coordination of services. It adapts the usage of DTs and the plant such that the framework guarantees users a quality of service (QoS) even when the plant is shut down for a period.
- *Orchestrator 2*: A module that coordinates the functions for training, verifying, and monitoring DTs.

QoS is levelled and adaptive as the orchestration adapts based on the statuses of the plant and DTs. For example, when a service has no restrictive end-to-end deadline but requires high data accuracy, the orchestrator requests data directly from the plant. When the request for service needs to be responded to within a time frame, i.e., a hard deadline, the orchestrator turns to the corresponding DTs, which offer the demanded data much faster than the plant because the former simulates the result instead of running the actual physical process. However, the data accuracy would drop, which should be informed to the users. In this example, we mention two dimensions of QoS, i.e., response time and data accuracy. QoS has many other dimensions, such as the rate of packet loss. Given different situations and priorities, QoS is adjusted and informed to the users such that they can make a decision on their own, or rely on the decision-making service of *D-RODS*.

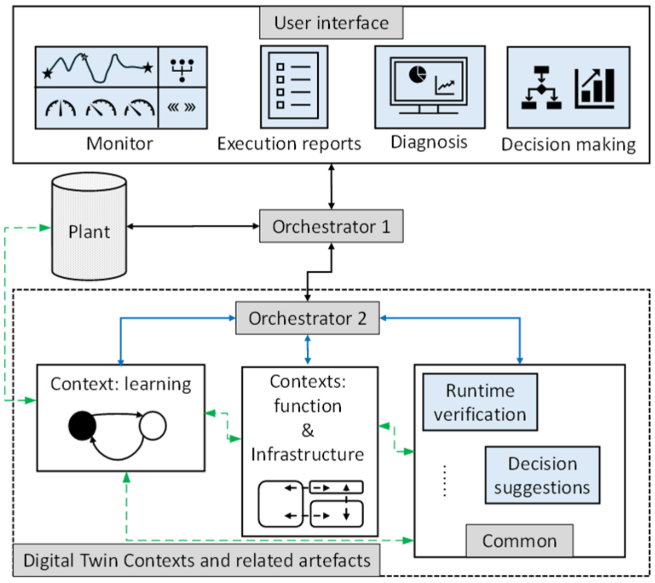


Figure 2. The SOA frame work of *D-RODS*. Blue boxes are services.

The orchestration within the DT contexts (implemented by e.g. Orchestrator 2 in Figure 2) is (additionally) meant to support various communication and data exchange policies between connected DTs, where such policies are allowed by time constraints of the system. We exemplify such a communication further down, within our use case description.

Considering the complexity of the SOA framework of *D-RODS*, balancing the dimensions of QoS is nontrivial. One obstacle is that DTs are often provided alongside their corresponding real products manufactured by different companies. The lack of documentation and standards of DTs would make the orchestration mechanism exponentially complex as subsystems increase. Moreover, we can only treat those subsystems and their DTs as black boxes, which means analysis of the system for QoS guarantee is extremely difficult.

QoS-aware Service Orchestration. Inspired by the concept of responsibility-sensitive safety (RSS) [15], a rule-based approach for autonomous driving safety, we propose a QoS-aware mechanism of service orchestration. Essentially, a level of QoS is expected to be fulfilled if and only if its constraints on the framework are satisfied. When the constraints are partially violated, the orchestration mechanism must adapt the services accordingly and give the users options combining the dimensions of QoS differently. Let us explain how this mechanism works with an example.

Real time aspects. Over the large body of literature, SOA approaches have been pronounced capable of real-time operations, however, mostly relating to exchanges at the *Orchestrator 1* (Figure 2) levels. This goes along with accepting a "relaxed" interpretation of real-time, mostly understood as *soft* real-time considerations. The selection of suitable protocol(s), of suitable communication pattern(s), etc. are additional tweaking aspects to be considered for answering real-time responses of a SOA-based system. Potentially, a transition into *microservices* may also provide good responses, at the cost of finer granularity of the composing services (and hence, an increase of the system size in terms of component numbers).

QoS-oriented Learning. Machine learning is crucial to monitor and predict the quality of services and dynamically control the orchestration and resource utilization to meet complex requirements in various applications. We envisage the following learning issues to be investigated in the development of our service-oriented digital twin framework:

a) Anomaly detection: to support the identification of any anomaly or degraded performance in real-time monitoring of the service system. The particular focus of the research will be learning the model for detection in an unsupervised manner using unlabelled data. Self-supervised learning algorithms need to be developed to autonomously distinguish normal and anomalous status with minimization of both false negative and false positive rates.

b) Service quality prediction is required to find the optimal orchestration of services in the design stage to achieve the QoS that best satisfies all customers. Robust machine learning will be a crucial issue here to ensure accurate and reliable prediction results that are scarcely affected by the condition deviating from what is anticipated in the design stage.

c) Distributed learning: to support the distributed architecture of our service-oriented digital twin framework. The whole system comprises many DTs that can collect local data and then build models based on the local data. It is desired that these individual DTs communicate with each other to share information and enhance performance. Distributed and federated learning provides a powerful means for the DTs to augment their capability by building an aggregated knowledge model while without requiring data transfer.

d) Reinforcement learning of mitigation policy: to acquire the optimal policy of reallocating computing and networking services when a failure occurs in the infrastructure or there is a drop in the quality of service. Offline learning of the mitigation policy will utilize the model for service quality prediction as stated in b). The learning can be subject to two objectives. On one hand, we aim to shift to a new allocation or orchestration optimizing the quality of services. On the other hand, we also desire to minimize the energy and computing resources in the updated allocation scheme.

Verification of QoS. In line with the QoS-aware service orchestration, we propose a Hoare-logic-based verification of the levelled QoS. Intuitively, constraints are modelled as logic expressions, which serve as the preconditions (denoted as P) of the services (denoted as S). Each level of QoS is modelled as a logic expression, which are the postconditions (denoted as Q) of services. Now, services of *D-RODS* can be expressed by the well-known Hoare triple: $\{P\}S\{Q\}$, and we use axioms and inference rules to demonstrate the level of QoS.

DT development flow. Collecting the aspects we discussed above, a generic development flow of DTs is illustrated in Figure 3. We identify a "usual" development path, based on tools of choice, grouped under the *SotA* "path", which produces the *Functional DT*(s), and an additional path implementing a (formal) V&V approach, providing the *Formal DT*(s) of the system. In our developments, we used Simulink[1] - for collecting (simulation) data, SIMPPAAL [16] - a proprietary tool performing the necessary translation into UPPAAL [17] models - the tool we use for V&V. We employed various machine learning algorithms and architectures to obtain the DT models on the "SotA" path, while specific support is provided by formal verifications of the machine-learning process and result (a.k.a. formal DTs) on the *V&V* path.

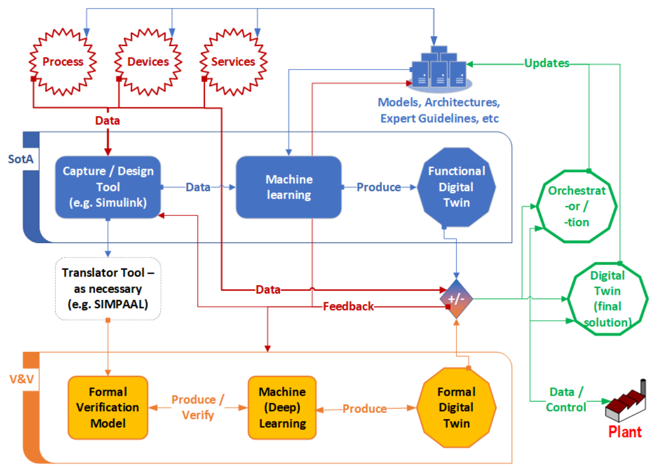


Figure 3. The *D-RODS* development framework.

The V&V approach. Our use case model requires a translation, from the Simulink into the Uppaal environments. The tool further identifies the state flow of the execution, and compares it to the results of the actual execution (Simulink simulation), as described in Figure 4. In Uppaal, actions, which are represented by arrows in Figure 4, are categorized as *controllable* and *uncontrollable*. Normally, controllable actions (solid arrows) are the system's actions that learning algorithms (a.k.a., learners) can observe and uncontrollable actions are the environment's actions (dashed arrows) that are invisible to learners. A learner's goal is to synthesize a controller for the system to function correctly and safely. However, when we model the inputs and outputs of a Simulink model as controllable actions, and the internal function of the Simulink model as uncontrollable actions, the internal function is treated as a

[1]www.mathworks.com

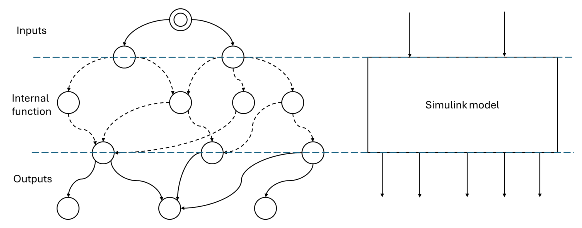


Figure 4. V&V activities in Uppaal.

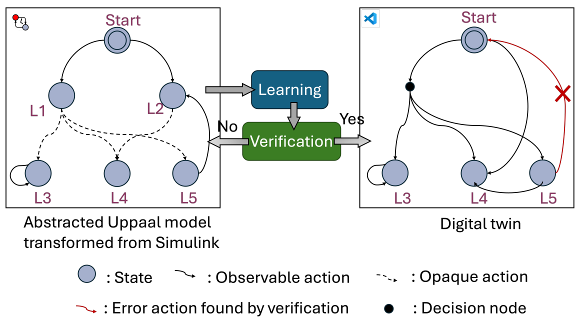


Figure 5. DT result from V&V activities.

black box by the learner. After obtaining the respective DT, we can run exhaustive verification on the Uppaal model against a liveness property, e.g., A◇ goal, which means the model must always reach the goal regardless of how the uncontrollable actions take place. When running such a verification, we can verify, for instance, if the DT covers all the branches of the Uppaal model, including the controllable and uncontrollable ones. In this way, we can claim the training data has a full coverage of the target system, i.e., the Simulink model.

VI. Use Case - The Heating System

We choose to illustrate our approach with a synthetic use case of heating various liquids to evaporation. We model a *wood provider* delivering combustible to a *warehouse*, from where a *burner system* is fed for raising the temperature of a *boiler* supporting cans made from various materials that contain different liquids. The variants of the system are based on randomly chosen values for different coefficients related to the wood-burning efficiency, liquid heating, pot heat transfer, etc. We also randomly choose values for the initial volumes of wood and liquid, sizes of wood deliveries, the volume of wood requests, evaporation temperatures, etc. A high-level depiction of the system with its functional elements and with some simple information flow is presented in Figure 6.

We define a system (the *System of Interest* - *SoI*) composed of three sub-systems (the *Warehouse* - *WH*, the *Burner System* - *BuS* and the *Boiler System* - *BoS*) and an external one (the *Wood Delivery System* - *WDS*). We intend to build digital twins corresponding to the SoI and some parts of its sub-systems. We have to note that the WDS is not part of our platform development, it is considered an environment system; we do modify its operating conditions (when, and how much wood it delivers), but we do not create any digital twin based on this.

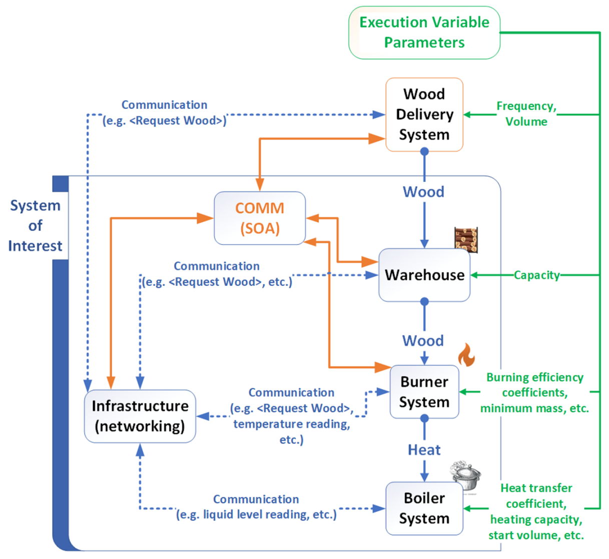

Execution Variable Parameters
Wood Delivery System
Frequency, Volume
Communication (e.g. <Request Wood>)
System of Interest
COMM (SOA)
Wood
Warehouse
Capacity
Communication (e.g. <Request Wood>, etc.)
Wood
Burner System
Burning efficiency coefficients, minimum mass, etc.
Infrastructure (networking)
Communication (e.g. <Request Wood>, temperature reading, etc.)
Heat
Boiler System
Heat transfer coefficient, heating capacity, start volume, etc.
Communication (e.g. liquid level reading, etc.)

Figure 6. The heating system.

We build the heating system in Simulink, where (on the *SotA* path) we also develop our ML models, trained on 100 simulated executions, with the results collected as time series.

Operation scenarios. The WH delivers requested amounts of wood, as much as available, to the BuS. Following the dotted lines in Figure 6, whenever the requested mass of wood is not available in the WH, a request is sent to the WDS. The WDS delivers a fixed quantity of wood, repeatedly, following requests from the WH, which receives and stores the wood. When the required quantity of wood has been delivered, the BuS releases the request.

Once the system is started, the wood available within the BuS helps raise its temperature (with a specific slope). The temperature range is limited to a maximum given value (e.g. 300), and it starts decreasing (with a different slope) when the system is off.

The heat is transferred to a *pot* containing a certain liquid - both with specific heat transfer coefficients and heat capacities, respectively. When the liquid reaches its vaporization temperature, its temperature remains constant, and its mass decreases. In case this mass of liquid decreases under a specified volume, the system automatically turns off.

Implementing and executing the use case. The heating system described above consists mainly of physical modelling components (the BuS and the BoS). A potential SOA approach can only be applied between the WDS, the WH and the BuS. The SOA management of the exchanges across these three units has been implemented with a distinct block, COMM (Figure 6) playing the role of the *Orchestrator 2* (Figure 2), and along the attached connectors. We observe an asynchronous communication pattern within the sub-system.

As the system is small and only the WH interacts with the other components, we did not implement a service registry or the related functionality. The communication is conducted via the network, and the delivery of services (“wood” quantities)

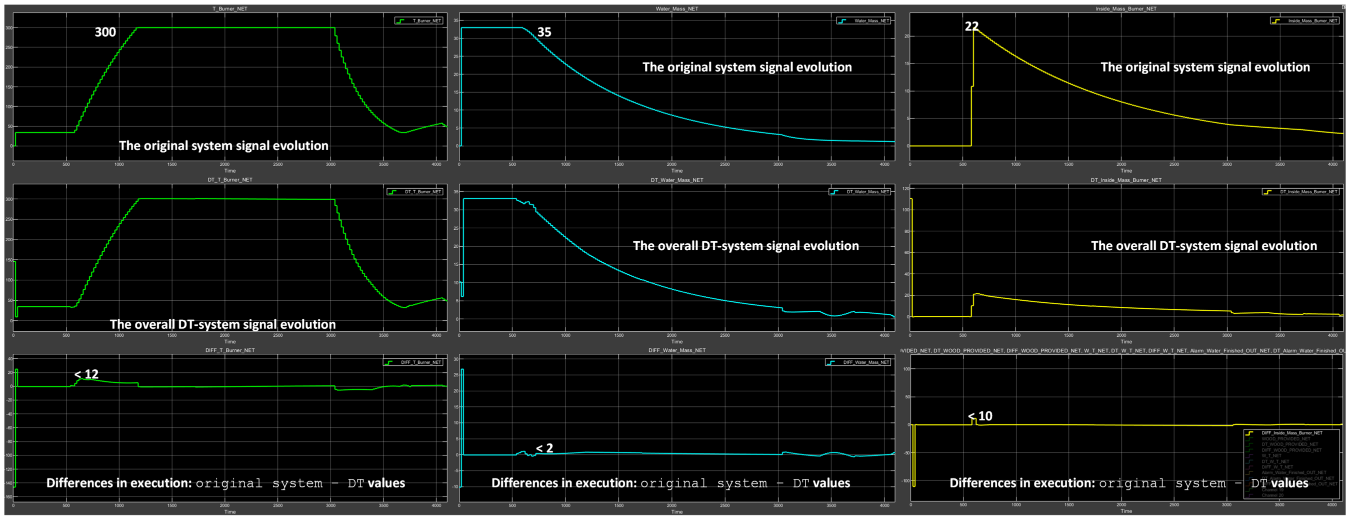


Figure 7. The heating system execution.

is directly handed to the respective recipient (the WH or the BuS). We obtain, based on simulation data, several DTs for the components of interest. For the purpose of this study, we observe first the original behaviour of the system (top of Figure 7) and the result of an overall DT execution (middle of Figure 7) and we also depict the differences between these at the bottom of Figure 7[2]. The graphs refer to the variables storing the mass of water in BoS, the mass of wood in BuS and the temperature of the BuS.

Please note in the mentioned figure, that, ignoring the initial system start moments, the maximal error values are kept within quite suitable tolerance limits. An exception here is the `Wood mass` signal. All such errors are due to a certain time misalignment, for which we had no immediate solution.

Further, we build a specific DT for the COMM system (DT-COMM) and include it as a replacement for the actual COMM bloc in the original system. We are interested in how the important signals are managed by DT-COMM and what is the effect on the overall execution. For this, we select the `Wood_Request` signal, with an original shape as in Figure 8 a). The respective un-processed shape of the DT-COMM signal is illustrated in Figure 8 b), while Figure 8 c) shows the corrected execution of the DT-COMM.

Further, the consideration of DT-COMM does not change the overall DT-based system execution (at least concerning the selected variables), the simulation presenting indistinguishable differences to the signals in Figure 7.

VII. Discussion

We've shared here several steps ahead towards our goal of building a DT development and operational framework. We included SOA aspects and implemented them within our use case. We also identified and applied V&V tools to correct the "raw" DTs, obtained in an usual manner.

One may observe that the errors identified in the raw DTs are not fully eliminated by the V&V solutions: our tools identified the prediction anomaly (and accordingly "filtered" it out), but the value levels were not corrected. We believe that, with the current implementation of the system, this may actually not be possible at all: the `Wood_Request` signal is defined with the help of a *random* assignment, which may neither be suitably identified by any machine learning algorithm nor corrected by any V&V method. Therefore, in future system realisations, we will increase determinism by assigning a fixed value to define the specified variable. This will lead to a limited variation of the signal, which, in turn, may provide either sufficient information for a more accurate machine learning prediction, or the correction(s) will become more efficient.

However, to our surprise (however pleasant), the errors identified in the raw DTs did not have a large impact on the execution of the use case, considering high-level variables, at least. At the moment, it is not possible to state if this is a specific effect only valid given the selected use case, or if this is a more generic aspect.

The advantages that we see with the SOA solution, partly implemented in our use case, are the expected ones when discussing design choices. The modularity of the SOA subsystem allowed us a "clean" replacement of the original component with the respective DT, with no additional impact on the rest of the system. This is one major goal of our targeted platform [18].

The real-time aspects here were not critical. However, by measuring the timing in answers, we may still have to reconsider the implemented SOA solution for (much) shorter time requirements. Also, various selections of machine learning approaches may add to the timed execution. These aspects

[2]Please note for both Figures 7 and 8, the metrics (e.g. degrees, litres, etc.) on the y-axis data, plotted against time, are of no relevance; the important aspect is the *shape* of the graph and the range of pairwise values.

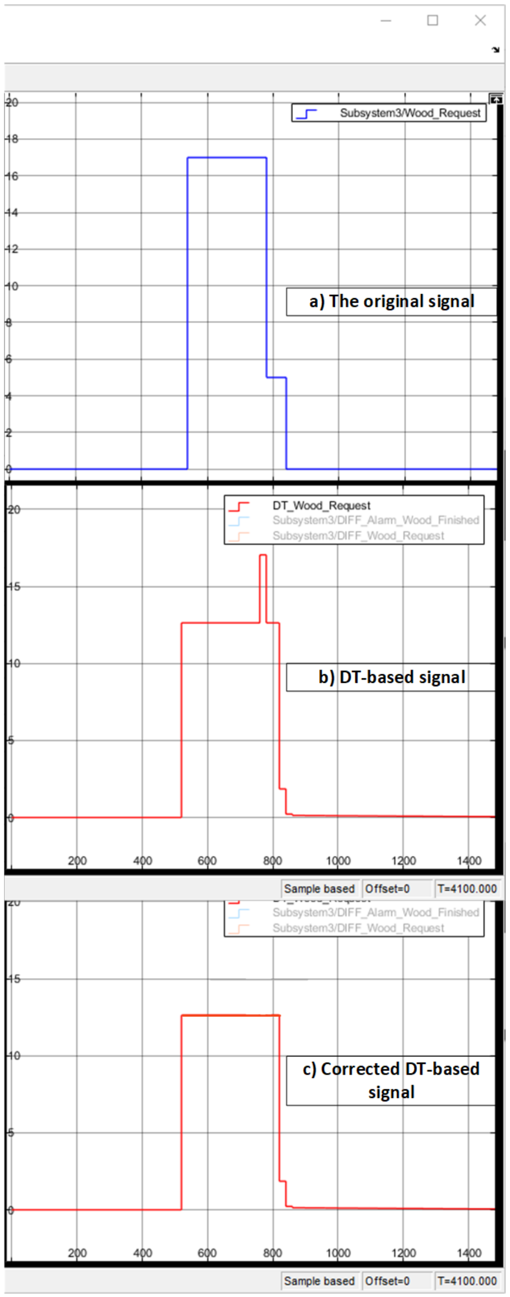


Figure 8. The *Wood_Request* signal.

will be addressed in further developments on our framework.

VIII. Conclusions and Future Work

We have introduced here our views on employing DT solutions for (partly) SOA-organized systems. While the results are encouraging, there are indications that more research is necessary in order to reach our targeted development and operational framework for DTs.

We also included a first view on V&V solutions meant to improve the raw versions of our DTs. Certain corrections are possible but with some incomplete coverage. Potential other approaches can also be tested in the context, trying to improve the results of verification activities. However, the selected V&V tools provided further insights in the operational aspects of the DT-based system.

Acknowledgements. The authors are partly supported by Vinnova's *Advanced digitalization* programme in the project D-RODS (ID: 2023-00244), and by a grant from the Swedish Knowledge Foundation in project PerFlex (ID: 20180150).

References

[1] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP journal of manufacturing science and technology*, vol. 29, pp. 36–52, 2020.
[2] K. Lamb, "Principle-based digital twins: a scoping review," *Centre for Digital Built Britain: Cambridge, UK*, 2019.
[3] R. Lutze, "Digital twins in ehealth–: prospects and challenges focussing on information management," in *2019 IEEE International Conference on Engineering, Technology and Innovation*. IEEE, 2019, pp. 1–9.
[4] M. Perno, L. Hvam, and A. Haug, "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers," *Computers in Industry*, vol. 134, p. 103558, 2022.
[5] A. Kung, C. Baudoin, and K. Tobich, "Report of twg digital twins: Landscape of digital twins," *EU Observatory for ICT Standardisation*, 2022.
[6] T. Seceleanu, N. Xiong, E. P. Enoiu, and C. Seceleanu, "Building a digital twin framework for dynamic and robust distributed systems," in *The 8th International Conference on Engineering of Computer-Based Systems, Västerås, Sweden, 16-18 October 2023*. LNCS, Springer Science and Business Media Deutschland GmbH, 2024, pp. 254–258.
[7] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.
[8] F. Tao, M. Zhang, J. Cheng, and Q. Qi, "Digital twin workshop: a new paradigm for future workshop," *Computer Integrated Manufacturing Systems*, vol. 23, no. 1, pp. 1–9, 2017.
[9] N. M. Josuttis, *SOA in practice: the art of distributed system design*. " O'Reilly Media, Inc.", 2007.
[10] M. Ciavotta, G. D. Maso, D. Rovere, R. Tsvetanov, and S. Menato, "Towards the digital factory: a microservices-based middleware for real-to-digital synchronization," *Microservices: Science and Engineering*, pp. 273–297, 2020.
[11] B. Pernici, P. Plebani, M. Mecella, F. Leotta, F. Mandreoli, R. Martoglia, G. Cabri *et al.*, "Agilechains: agile supply chains through smart digital twins," in *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference, Venice, Italy*, 2020, pp. 1–5.
[12] F. Longo, L. Nicoletti, and A. Padovano, "Ubiquitous knowledge empowers the smart factory: The impacts of a service-oriented digital twin on enterprises' performance," *Annual Reviews in Control*, vol. 47, pp. 221–236, 2019.
[13] K. Kruger, C. Human, and A. Basson, "Towards the integration of digital twins and service-oriented architectures," in *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2021, pp. 131–143.
[14] C. Human, A. Basson, and K. Kruger, "A design framework for a system of digital twins and services," *Computers in Industry*, vol. 144, p. 103796, 2023.
[15] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint:1708.06374*, 2017.
[16] P. Filipovikj, N. Mahmud, R. Marinescu, C. Seceleanu, O. Ljungkrantz, and H. Lönn, "Simulink to uppaal statistical model checker: Analyzing automotive industrial systems," in *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*. Springer, 2016, pp. 748–756.
[17] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *International journal on software tools for technology transfer*, vol. 1, pp. 134–152, 1997.
[18] R. Gu, T. Barbuceanu, N. Xiong, and T. Seceleanu, "Experiences in building a digital twin framework: Challenges and possible solutions," *The 48th IEEE International Conference on Computers, Software, and Applications*, To appear. 2024.