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.

PRESERVING DEPENDENCIES IN PARTITIONED DIGITAL TWIN MODELS FOR ENABLING MODULAR VALIDATION

Ashkan Zare[1] and Sanja Lazarova-Molnar[2,1]

[1] Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, DENMARK
[2] Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, GERMANY

ABSTRACT

Leveraging Digital Twins, as near real-time replicas of physical systems, can help identify inefficiencies and optimize production in manufacturing systems. Digital Twins' effectiveness, however, relies on continuous validation of the underlying models to ensure accuracy and reliability, which is particularly challenging for complex, multi-component systems where different components evolve at varying rates. Modular validation mitigates this challenge by decomposing models into smaller sub-models, allowing for tailored validation strategies. A key difficulty in this approach is preserving the interactions and dependencies among the sub-models while validating them individually; isolated validation may yield individually valid sub-models while failing to ensure overall model consistency. To address this, we build on our previously proposed modular validation framework and introduce an approach that enables sub-model validation while maintaining interdependencies. By ensuring that the validation process reflects these dependencies, our method enhances the effectiveness of Digital Twins in dynamic manufacturing environments.

1 INTRODUCTION

Digital Twins (DTs), as near real-time evolving virtual replicas of physical systems (Grieves 2014), are a great asset in today's complex manufacturing industry. To utilize the full potential of DTs, they need to continuously reflect the real-life systems accurately. However, as manufacturing systems grow increasingly complex, corresponding DT models become more cumbersome, making the validation process an enabler of DT and a significant challenge. Therefore, traditional one-time validation approaches cannot be applied to DT models as the evolving nature of DTs requires continuous validation.

Furthermore, as complex manufacturing systems consist of different components that evolve at varying rates, validating systems as single units may not be adequate. Therefore, in our previous research (Zare and Lazarova-Molnar 2024a), we introduced a modular validation approach, partitioning models into sub-models based on activity rates and setting validation frequencies accordingly. Our approach then determined whether sub-models needed to be recalibrated, re-extracted, or kept intact. The key challenge in partitioning is defining meaningful sub-models while preserving the underlying dependencies and interactions among the components. Even though in our initial approach we relied on rate of occurrence of activities in the system and data recorded in event logs for model partitioning and capturing of dependencies, the need for more robust approaches to partitioning persists. Here, we extend our modular validation approach to enhance sub-model validation while maintaining interdependencies.

In manufacturing systems, not all components behave or change with the same rate or in the same way, and their impact on the system differs. Thus, model partitioning must consider multiple criteria to accurately reflect these variations. It is also equally important to not treat the partitioned sub-models as complete independent entities as components and processes are interconnected and changes in one may impact the others. For example, as depicted in Figure 1, consider a sequential manufacturing line where the first robotic arm must finish its operation before the second one can begin its process. If these interdependent manufacturing processes are partitioned in different sub-models and each sub-model is validated in complete isolation (indicated by red rectangles), the validation process ignores the critical interactions and dependencies between them. As a result, we may still deem each sub-model valid while validity of the complete model remains questionable. Therefore, modular validation is useful if the underlying partitioning approach effectively partitions the model and preserves the dependencies (indicated by a blue rectangle) among the resulting sub-models.

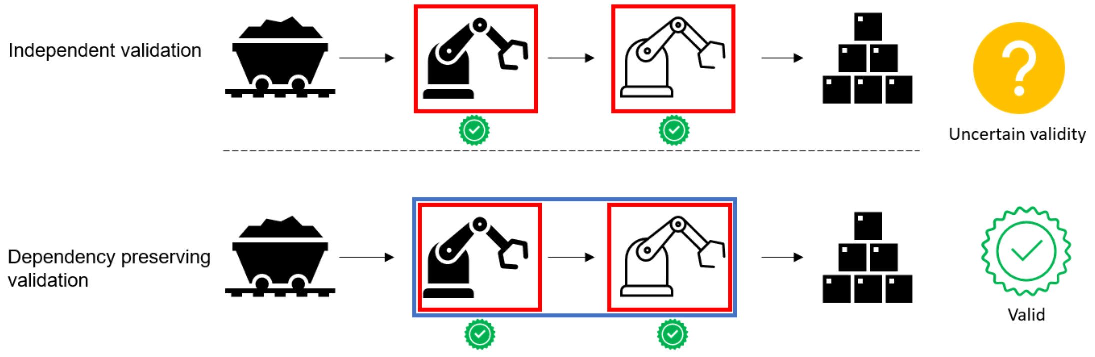

Figure 1: Preservation of dependency when partitioning.

To address the challenges of partitioning models for validation while preserving interactions and dependencies among components, we propose a data-driven approach that maintains these dependencies among sub-models during the partitioning of the complete DT model. In our proposed approach, we adopt stochastic Petri nets as modeling formalism, leveraging their abilities to represent and preserve interactions among components. Stochastic Petri nets (SPNs) are a well-suited formalism for capturing and describing processes in discrete-event systems such as manufacturing systems. In SPNs, activities are modeled as transitions that fire to destroy and create tokens between places, creating the dynamics in Petri nets. We utilize the flow of tokens within Petri nets to preserve dependencies after model partitioning. Additionally, we formalize our SPN-based approach using the Petri Net Markup Language (PNML) (Billington et al. 2003), enabling structured representation and interoperability.

The rest of the paper is organized as follows. In Section 2, we review the background and related work on DT validation and model partitioning. Section 3 presents our methodology for partitioning models with dependency preservation. In Section 4, we further evaluate our approach through a manufacturing case study. Finally, we conclude the paper and discuss future research directions in Section 5.

2 BACKGROUND AND RELATED WORK

In the following, we review the current literature on validation of DTs and highlight the need and the existing gap in modular approaches, provide an overview of model partitioning, and formally describe stochastic Petri nets.

2.1 Validation of Digital Twins

DTs, by definition, are required to continuously reflect the current state of their real-world counterparts. To ensure this synchronicity, model validation must be an integral component of DTs, serving to confirm the accuracy of the DT's model within its intended domain of application (Schlesinger 1979). Validation of simulation models is a well-established research area, with numerous validation techniques developed over the decades (Sargent 2013). These conventional approaches, however, are not readily applicable to DTs, as static, one-time model validation is insufficient for models that need to evolve over time (Zare and Lazarova-Molnar 2024b).

The challenge and necessity of ongoing validation of DT models has gained increasing attraction in recent years, prompting the development of new simulation model validation approaches that enable continuous validation of DT models. For instance, Hua et al. (2022) proposed a hybrid validation approach that combines human expert input with data collected via IoT devices. Utilizing time series analysis, an online approach was proposed by Lugaresi et al. (2023) based on data collected from IoT devices and focusing on operational phase of DTs. Mertens and Denil (2024) explored anomaly detection and reuse of existing model validation techniques through the monitoring of validation metrics for continuous assurance. A signal processing approach was employed by Morgan and Barton (2022), who utilized Fourier transform properties to detected discrepancies using the Fourier coefficient magnitudes, which was tested for validation of dynamic behavior of a DT.

Other noteworthy contributions include the use of machine learning and statistical process control methods for continuous validation of initially validated DTs, as demonstrated by dos Santos et al. (2023). Separating initial validation from ongoing validation at run-time, Friederich and Lazarova-Molnar (2023a) proposed a two-phase validation approach for validation of data-driven discrete-event simulation models. An automated periodic validation and update method based on comparing key performance indicators (KPIs) between the physical system and its DT representation was proposed by Overbeck et al. (2023). Furthermore, aggregating all epochs' data and multi-variate KPIs, He et al. (2024) evaluated validity of DT models.

Recent studies highlight the importance of validation in DTs. However, existing approaches treat models as a whole and focus solely on final outputs. This may not be effective for complex systems, where localized inaccuracies may propagate and compromise overall model reliability. This limitation underscores the necessity for modular model validation processes and partitioning of DT models, which allow for more granular, component-level validation strategies, critical for managing the complexity and dynamics inherent in modern DT applications.

2.2 Partitioning Simulation Models

The transition from traditional to advanced manufacturing, enabled by the advancement in technologies such as cyber-physical systems and IoT (Xiang et al. 2023), has significantly increased the complexity of manufacturing systems. Consequently, the corresponding simulation models have also grown in complexity. To address the challenges of running complex simulations models, both in terms of resources and validation, *model partitioning*, which involves decomposing a large model into multiple smaller, more manageable sub-models, offers a promising solution. Although partitioning can aid in computational load-balancing, distributing workload among processes, and run time reduction (Boukerche and Das 1997), our aim is to utilize partitioning for modular validation of DTs.

Model partitioning as a means of optimizing computation efficiency and reducing simulation time, has been extensively studied. Within this context, GloMoSim library for parallel simulation was developed Zeng et al. (1998). It enables computational load distribution by partitioning and decomposing of simulation models. Furthermore, Peschlow et al. (2007) proposed a dynamic partitioning algorithm for partitioning simulation to optimize communication and computation workload. Costa and Gomes (2009) proposed a Petri net partitioning through net splitting operation for concurrent execution within embedded systems. Utilizing principles of dynamic decoupling, Papadopoulos and Leva (2015) proposed an approach to partition a model based on relevant time scales. Myers et al. (2016) proposed an online in situ partitioning method to reduce data transfer and storage requirements during simulation. In addition, self-clustering techniques were used by D'Angelo (2017) to balance computational loads and reduce communication overhead. Bogdanovic et al. (2022) developed a partitioning method for distributed simulations by analyzing time delays, ensuring simulation stability after decoupling. Based on fuzzy c-means clustering algorithm, He et al. (2025) proposed a simulation zone partitioning algorithm that improves computational efficiency for agricultural industry.

From the related literature, partitioning is primarily employed as a technique for load balancing and optimizing resource consumption. However, since our goal is to validate simulation models by partitioning and through intermediate, sub-system validation (Netter et al. 2013), we identify a gap in existing research. This highlights the need for further investigation into partitioning strategies specifically designed to support model validation.

2.3 Stochastic Petri Nets

As noted in the introduction, we adopt stochastic Petri nets (SPNs) as our modeling formalism. Petri nets (Petri 1962) are a modeling formalism best used in describing discrete-event systems. Since its introduction in 1962, there have been many extensions to the Petri nets. Stochastic Petri nets are an extension of Petri nets that allows for transitions in Petri nets, activities in systems, to be assigned probability distribution functions determining their firing delays. We adopt SPNs as the timed transitions can mimic processes in manufacturing systems. SPNs are formally described as the following (Lazarova-Molnar 2005):

$$SPN = (P, T, A, G, m_0)$$

Where:

- P is the set of places in the Petri net, drawn as circles.
- T is the set of transitions, both timed and immediate, drawn as bars. Immediate transitions have a constant value assigned to them while timed transitions have corresponding distribution functions that compute the firing probability.
- A is the set of arcs, input, output, or inhibitor arcs, connecting places and transitions. Depending on connecting a place to a transition or a transition to a place, arcs are identified as *input arc* or *output arc* respectively. *Inhibitor arcs*, connecting places to transitions, block the transitions if the number of tokens in a place is greater than or equal to the multiplicity of the arc.
- G is the set of guard functions and their associated transitions.
- m_0 is the initial state of the Petri net, referred to as initial marking of the Petri net.

Furthermore, in our approach, we take advantage of the Petri net markup language (PNML), a universal XML-based format to describe Petri nets. PNML's flexibility and compatibility ensures any kind of Petri net and its additional information can be converted to PNML, and for new Petri nets to be interpreted by any tools without previous knowledge of their type (Billington et al. 2003). By incorporating PNML in our approach, we ensure the partitioning, and the overall modular validation, is not limited to a certain type of Petri net. In Figure 2, we show a graphical example of a stochastic Petri net model of a simple manufacturing process and a snippet of its corresponding PNML file.

3 DEPENDENCY PRESERVATION IN VALIDATION

In this section, we present our approach to preserving interactions and dependencies in partitioned models by introducing *interface places* and *token generators*. We first provide an overview of our previously proposed modular validation framework (Zare and Lazarova-Molnar 2024a) and then highlight our new approach for partitioning and preserving dependencies. As noted earlier, we adopt stochastic Petri nets as modeling formalism and, as such, we tackle the challenge of maintaining dependencies by utilizing *interface places* and recording the token flow within models. This ensures that interactions between sub-models are preserved, even after partitioning.

3.1 Modular Validation

To ensure a robust validation process, we previously proposed a framework for modular validation of Digital Twins' models that enables targeted validation of underlying sub-models (Zare and Lazarova-Molnar 2024a). As shown in Figure 3, the framework consists of two phases. The first phase focuses on validating the initial model, described using Petri Net Markup Language, which is done through standard

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pnml xmlns="http://www.pnml.org/version-2009/grammar/pnml">
  <net id="SimpleWarehouse" type="http://www.pnml.org/version-2009/grammar/ptnet">
    <name>
      <text>Simple Warehouse Process</text>
    </name>
    <page id="page1">
      <!-- Places -->
      <place id="P1">
        <name><text>warehouse</text></name>
        <initialMarking><text>0</text></initialMarking>
      </place>
      <!-- Transitions -->
      <transition id="T1" type="T">
        <name><text>arrival of task</text></name>
        <distribution>
          <type>exponential</type>
          <parameters>
            <a>0.5</a>
          </parameters>
        </distribution>
      </transition>
```

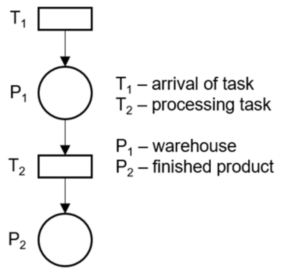

Figure 2: Stochastic Petri net model of a simple process with a snippet of its corresponding PNML file.

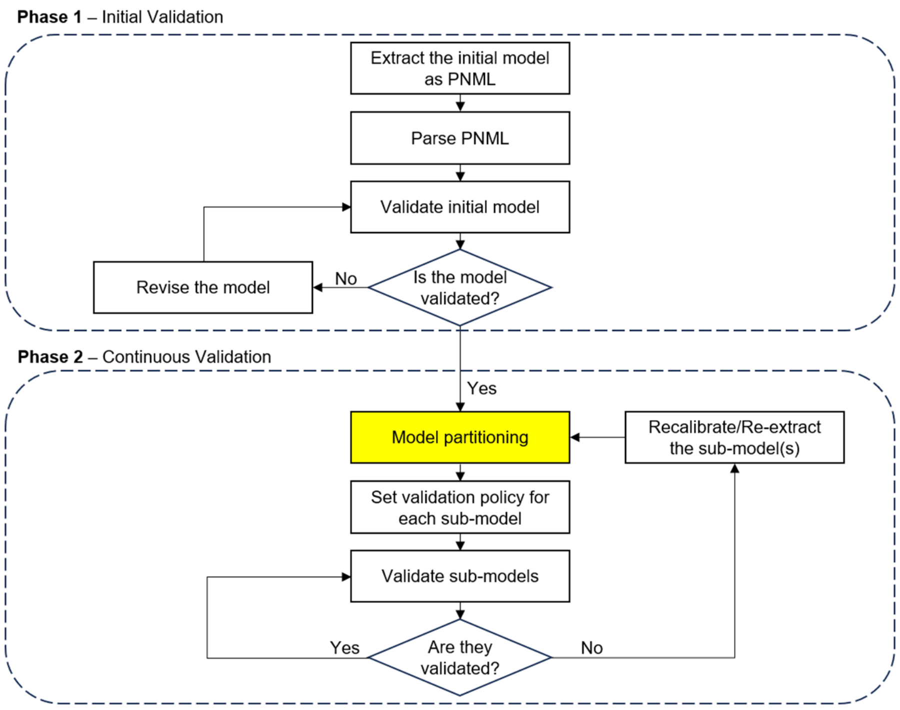
Phase 1 – Initial Validation
Extract the initial model as PNML
Parse PNML
Validate initial model
Is the model validated?
No
Revise the model
Phase 2 – Continuous Validation
Yes
Model partitioning
Recalibrate/Re-extract the sub-model(s)
Set validation policy for each sub-model
Validate sub-models
Are they validated?
Yes
No

Figure 3: Modular validation framework (Zare and Lazarova-Molnar 2024a).

validation techniques. The second phase is the continuous modular validation process in which the model is partitioned into sub-models, and each sub-model is validated based on the sub-model's validation policy. Finally, if any sub-model is deemed invalid, it undergoes recalibration or, if necessary, complete re-extraction.

In this paper, we focus exclusively on the model partitioning module, with particular emphasis on preserving interactions and dependencies among sub-models after partitioning. Identifying partitioning points is a non-trivial task that requires careful consideration. While in our earlier work (Zare and Lazarova-Molnar 2024a) we partitioned SPN models based on transitions and their firing rates, human expert knowledge, and structural analysis (Aybar and Iftar 2002) are other promising approaches in identifying logical partitioning points. Furthermore, in our previous work, we relied on event logs for detecting the underlying interactions and dependencies among the sub-models. This approach was effective as the SPN models were completely extracted from recorded event logs. This bottom-up approach, however, is not feasible when the event logs are not comprehensive for capturing dependencies or when the model is not extracted from event logs, such as manually developed or adapted models. Therefore, we explore the idea of using places as interfaces connecting one sub-model to another and record and recreate the token creation/destruction pattern in the sub-models to ensure dependencies are reflected regardless of how the initial model is developed. Preserving such dependencies during model partitioning is critical for modular validation as it not only ensures fidelity of a model as the model is broken up to multiple sub-models, but it also enables intermediate validation. Intermediate validation points aid in identifying underlying discrepancies even if variations do not affect the final system output.

3.2 Places as Interfaces

To maintain dependencies among sub-models, we introduce *interface places* to connect the sub-models. In SPNs, every transition must connect to a place; we utilize this characteristic to partition the model and identify the interface places. Specifically, when partitioning, we focus on places that connect transitions across different sub-models and are critical to the overall behavior of the system to use as interfaces. For instance, consider a simple example from manufacturing, as shown in Figure 4. When partitioning the model into two sub-models, the place between the two processes is identified as an interface place as it connects the transition from the first sub-model to the transition from the second one. This interface place, shown in Figure 4 as a blue circle, is duplicated in both sub-models to preserve the dependencies between the sub-models are maintained.

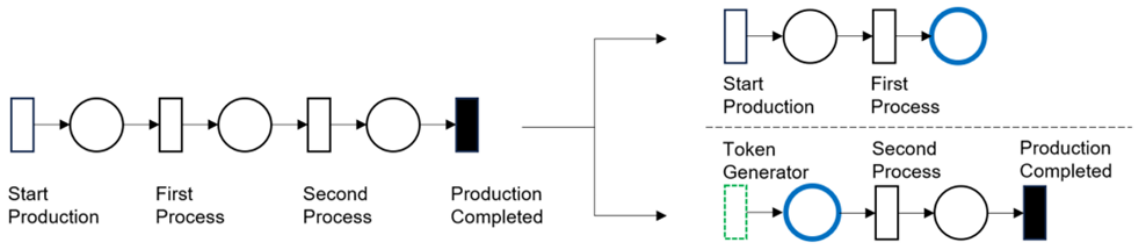

Figure 4: An example of using places as interfaces.

Furthermore, the interface places are duplicated based on the behavior of the system. In the given example, since the processes are in sequential order, the duplication is done from the first sub-model to the second one. This means the interface place gets updated if there is a change in the first sub-model and not the other way around. Having places as interfaces allows for partitioning models while maintaining interactions and dependencies among the resulting sub-models.

3.3 Token Flow Recording

To ensure independent modular validation process while preserving dependencies, we need a mechanism to replicate the dynamics of the system during independent validation of sub-models. As noted earlier, firing of transitions corresponds to occurrence of events in a discrete-event system. The creation/destruction of tokens, resulting from transitions firing, creates the dynamics in SPNs. When a model is partitioned, the dynamics are interrupted as creation/destruction of tokens may not match the original model. Therefore, we introduce a mechanism that records and recreates token flow at interface places across sub-models to preserve the dynamics of the complete model through creation of a special transition, called *token generator*, visualized as a dotted green rectangle in Figure 4.

First, during the initial validation of the complete model, we capture time-series data of token creation/destruction at each interface place. Second, we analyze the captured data to estimate the underlying probability distribution that best fits the token creation/destruction pattern. Third, we create a *token generator* for each sub-model that receives tokens from preceding sub-models in the complete system. For example, in Figure 4, the token generator is created for the interface place in the second sub-model, retaining dependencies to the first sub-model. Finally, the identified probability distribution functions are assigned to the corresponding token generators to recreate the complete model's dynamics. In case of changes to the real system, we implement a periodic update mechanism to reflect the current dynamics and dependencies of the real system in the sub-models. We capture new data of token creation/destruction, at defined intervals or manually, from the preceding sub-models or, if necessary, the complete model. The newly captured data is then analyzed to find probability distribution functions for interface places and if needed, the token generators are updated with the new probability distribution functions to reflect current system behavior. The periodic updates keep the token flow and the dynamic of our partitioned model consistent with the behavior of the complete system. The token generator ensures proper synchronization among sub-models and allows for modular and independent validation of sub-models while maintaining dependencies and behavior of the original model leading to complete model validation. Algorithm 1 presents our partitioning approach within our modular framework for continuous validation of Digital Twins along with the modifications needed at PNML level. The algorithm receives stochastic Petri net model, parsed from PNML code, as input and outputs PNML code excerpts of the sub-models with relevant properties for preservation of dependencies, namely interface places and token generators.

Algorithm 1: Algorithm for SPN partitioning as part of the modular validation framework.
Input: PNML file of a stochastic Petri net model, captured token data from initial validation.
Output: Partitioned sub-models with interface places and token generators as PNML code excerpts.

```
Step 1 Partition the SPN model based on expert knowledge or other relevant approaches. PNML
       elements associated with each sub-model are identified.

Step 2 Identify places that connect transitions across sub-models as interface places. Interface places
       are determined by comparing the partitioned models to the original model.

Step 3 Create duplicates of the interface places in the relevant sub-models. PNML elements (place
       and arc elements) for interface places and their associated arcs are created.

Step 4 For each interface place, estimate a best-fit probability distribution function based on the
       captured token data.
```

Step 5 Create token generators as output transitions with arcs leading to the interface places, and assign the corresponding probability distribution functions to these transitions. The transitions along with their associated distributions and arcs, are created as PNML elements.

Step 6 Return sub-models with preserved dependencies, namely token generators and interface places as PNML code excerpts for further model validation.

Step 7 Based on preset time intervals or manual input, capture new data of token creation/destruction and repeat **Steps 4-6** to reflect the current system dynamics.

4 ILLUSTRATIVE CASE STUDY

To demonstrate our approach for preserving dependencies while partitioning SPN models, we conducted an experiment using a manufacturing production line as case study. The line consists of five sequential stages: two assembly operations, a quality control station, a packaging operation, and a shipping operation. This configuration is ideal for testing our proposed approach as the sequential stages ensure clear dependencies among processes. We utilized the PySPN library (Friederich and Lazarova-Molnar 2023b) for simulation and validation. To facilitate processing of PNML files, we extended the library with a custom parser capable of translating PNML files into executable SPN models.

In our previous work, we applied a data-driven approach (Friederich and Lazarova-Molnar 2022) to extract the model and the dependencies from event data. Here, however, to examine our proposed approach we opted for a manual model development approach. Through manual development of the model, we ensure dependencies are preserved through introduction of interface places and token generator transitions, enabling a more focused evaluation of our proposed approach.

First, we manually developed the complete model as a Stochastic Petri net model formalized in PNML. We then simulated the complete model to capture token data and validated the model by comparing its production throughput to that of the system. Next, based on expert knowledge, we partitioned the model into two sub-models, an assembly stage and a post-assembly stage. In addition, we identified the quality control place to utilize as the interface place between the two sub-models as it connects transitions across the sub-models.

To preserve the dynamics of the system, based on the previously captured token creation/destruction data of the complete model, we determined a probability distribution function that best fitted the token arrival patterns of the interface place and created a token generator (timed transition), to replicate the behavior of the system in the post-assembly stage sub-model.

Finally, we generated separate PNML code excerpts for each sub-model for independent simulation and validation, including the relevant place, transition, and arc elements needed for the interface place and the token generator. Figure 5 shows the graphical representation of the resulting SPNs, derived by parsing the generated PNML code excerpts in PySPN. The interface place is highlighted in blue.

To evaluate the effectiveness of our model validation approach in preserving interdependencies between the two sub-models, replicating the dynamics of the original model, and ensuring accurate model validation, we conducted a comparative analysis across three simulation scenarios. We conducted 100 terminating simulation replications for each scenario: simulating the complete model (manually developed initial model), simulating the partitioned model with interface places and token generators to preserve dependencies (dependent sub-models), and simulating partitioned model without dependency preservation (independent sub-models). These scenarios were designed to evaluate the effectiveness of our dependency-preserving modular validation approach by comparing different KPIs from each scenario against the system (ground-truth). Comparing the throughputs of the three scenarios to the actual throughput of the system, as shown in Figure 6, we found that in the scenario where sub-models are independently validated, the dependencies were lost and the sub-models were validated as stand-alone models, resulting in an inconsistent outcome compared to the system and the initial model. However, our dependency-preserving validation approach replicated the dependencies and dynamics of the complete model and maintained fidelity to both the system and the complete model.

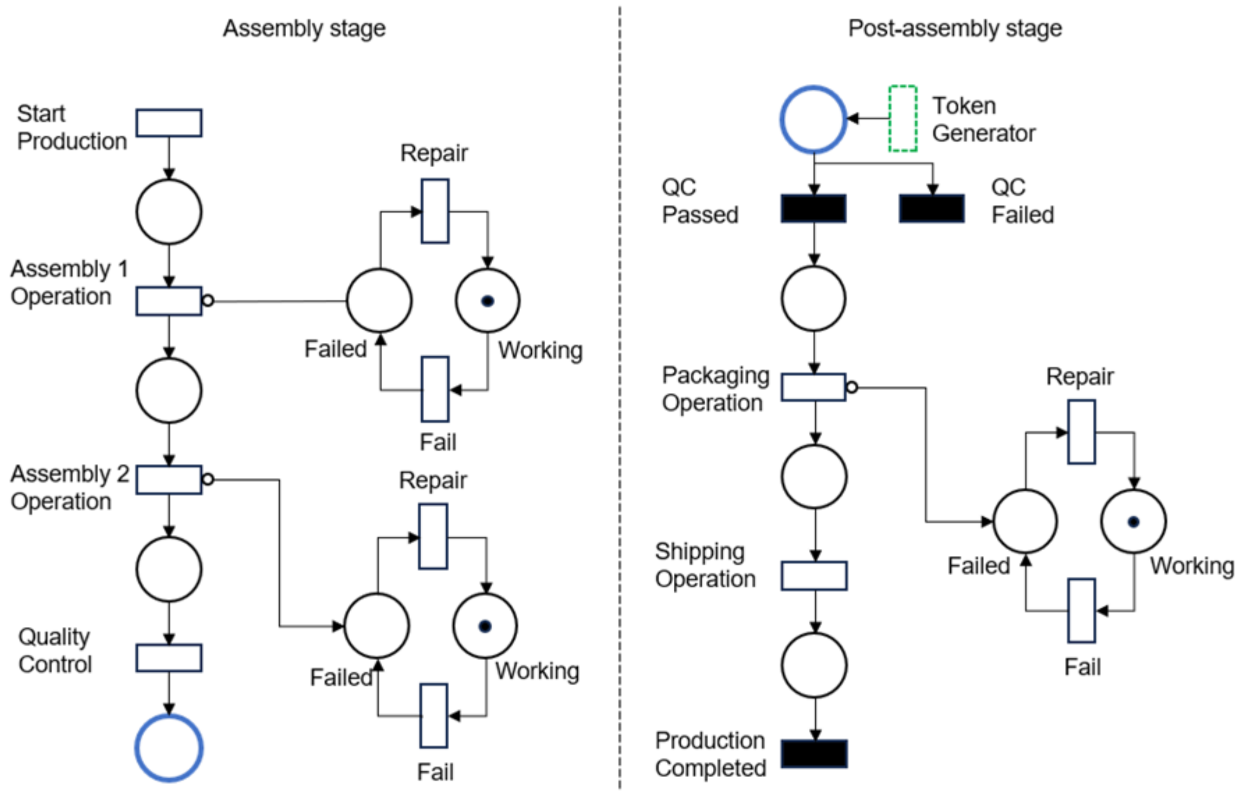

Figure 5: Resulting sub-models of our partitioning approach.

In addition, we experimented with an intermediate KPI to further ensure our approach preserves the dynamics of the original model and can be utilized for intermediate validation. The selected metric was *time to reach quality control*. This metric serves as a critical point in the case study as it is the interface connecting the two partitioned sub-models and can also be utilized for validation of the assembly stage.

To perform the comparison, we conducted 100 independent simulation replications on the complete model and on the partitioned assembly-stage sub-model (graphically represented on the left side of Figure 5) with inclusion of the interface place. The comparison of the intermediate metric (time to reach quality control), illustrated in Figure 7, showed our partitioning closely matches the behavior of the underlying processes and can be utilized for intermediate validation.

The KPI comparisons of the three scenarios in our case study demonstrated that our approach for partitioning DT models while preserving dependencies achieved its intended purpose. Through the introduction of interface places and token generators, we were able to successfully partition the model, preserve dependencies, and maintain dynamics of the system, enabling both modular validation as well as intermediate validation.

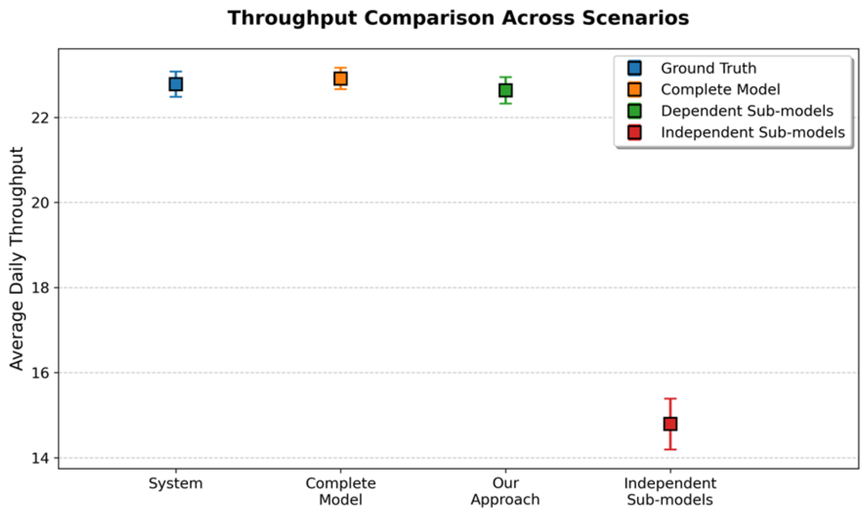

Figure 6: Comparison of simulation results (error bars representing 95% confidence interval for 100 independent replications).

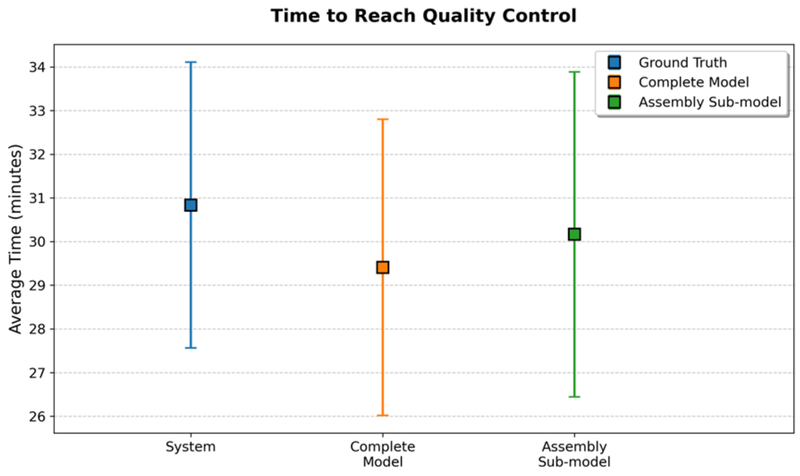

Figure 7: Comparison of result of intermediate validation points (error bars representing 95% confidence interval for 100 independent replications).

As the focus of this paper is on enabling modular validation of DT models through the preservation of dependencies during model partitioning, we designed the experiments to showcase the importance of preservation of dependencies and to demonstrate the effectiveness our approach. The dependency preservation approach builds upon our previously developed framework for modular validation of DT models, which incorporates elements such as validation frequency and model recalibration/re-extraction (Zare and Lazarova-Molnar 2024a).

5 CONCLUSION

Utilization of Digital Twins is a growing topic in manufacturing, enabled by the advancement in technologies. However, ensuring continuously valid Digital Twin models remains a challenge. The complex and dynamic manufacturing systems require robust model validation processes capable of detecting and rectifying discrepancies. To this end, as part of our earlier work, we proposed a modular validation framework, aiming to mitigate the challenges of validating underlying model of a Digital Twin by partitioning the model and validating the partitioned sub-models based on their individual characteristics.

The challenge arising from partitioning a complex model and independently validating the partitioned sub-models is to ensure that the intricate interactions and dependencies are preserved in the sub-models and ultimately reflected in the validation process. As these dependencies can influence the dynamics of a model, validating sub-models independently does not necessarily guarantee the complete model remains valid. To address this, we build upon our modular validation framework by proposing an approach that ensures preservation of dependencies during both the partitioning and validation processes of the complete model.

In our approach, we use stochastic Petri nets to describe underlying models, formalized through the Petri Net Markup Language, and utilized their structural characteristics, namely the place/transition structure and token creation/destruction, to preserve dependencies within partitioned models. In a stochastic Petri net, dynamics of a model is represented through the creation and destruction of tokens in places, triggered by firings of transitions. Therefore, we introduced interface places and token generators to capture and reflect the dynamics in sub-models. After partitioning of the complete model, we identified places that connect to transitions in different sub-models and designated them as interface places, duplicating them in the sub-models. To recreate the behavior of the system, we estimated best-fit probability distribution functions to drive token generation at the interface places within the sub-models, ensuring synchronization among the sub-models. Through a case study, we evaluated our approach and further demonstrated that utilizing interface places and token generators effectively preserves dependencies among sub-models and supports intermediate validation.

While this work addresses the preservation of dependencies in partitioning Digital Twin models for enabling modular validation, several challenges and limitations remain. Notably, the partitioning points in this study are determined through expert knowledge. As part of our future work, we aim to develop algorithms for automatic identification of optimal partitioning points and to further evaluate the proposed approach on more complex systems.

ACKNOWLEDGMENTS

This research is part of the ONE4ALL project, funded by the European Commission, Horizon Europe Programme under the Grant Agreement No 101091877.

REFERENCES

Aybar, Aydin, and Altug Iftar. 2002. 'Overlapping decompositions and expansions of Petri nets', *IEEE Transactions on Automatic Control*, 47: 511-15.

Billington, Jonathan, Søren Christensen, Kees Van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno, and Michael Weber. 2003. "The petri net markup language: Concepts, technology, and tools." In *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003 Eindhoven, The Netherlands, June 23–27, 2003 Proceedings 24*, 483-505. Springer.

Bogdanovic, Milica, Marija Stevic, and Antonello Monti. 2022. 'Non-Intrusive Delay-Based Model Partitioning for Distributed Real-Time Simulation', *Energies*, 15: 767.

Boukerche, Azzedine, and Sajal K Das. 1997. "Dynamic load balancing strategies for conservative parallel simulations." In *Proceedings of the eleventh workshop on Parallel and distributed simulation*, 20-28.

Costa, Anikó, and Luis Gomes. 2009. "Petri net partitioning using net splitting operation." In *2009 7th IEEE International Conference on Industrial Informatics*, 204-09. IEEE.

D'Angelo, Gabriele. 2017. 'The simulation model partitioning problem: an adaptive solution based on self-clustering', *Simulation Modelling Practice and Theory*, 70: 1-20.

dos Santos, Carlos Henrique, Afonso Teberga Campos, José Arnaldo Barra Montevechi, Rafael de Carvalho Miranda, and Antonio Fernando Branco Costa. 2023. 'Digital Twin simulation models: a validation method based on machine learning and control charts', *International Journal of Production Research*: 1-17.

Friederich, Jonas, and Sanja Lazarova-Molnar. 2022. "Data-driven reliability modeling of smart manufacturing systems using process mining." In *2022 Winter Simulation Conference (WSC)*, 2534-45. IEEE.

Friederich, Jonas, and Sanja Lazarova-Molnar. 2023a. "A Framework for Validating Data-Driven Discrete-Event Simulation Models of Cyber-Physical Production Systems." In *2023 Winter Simulation Conference (WSC)*.

Friederich, Jonas, and Sanja Lazarova-Molnar. 2023b. "PySPN: An Extendable Python Library for Modeling & Simulation of Stochastic Petri Nets." In *International Conference on Simulation Tools and Techniques*, 70-78. Springer.

Grieves, Michael. 2014. 'Digital twin: manufacturing excellence through virtual factory replication', *White paper*, 1: 1-7.

He, Jiaoyang, Yanxi Zhao, Ping He, Minglei Yu, Yan Zhu, Weixing Cao, Xiaohu Zhang, and Yongchao Tian. 2025. 'Rice Yield Prediction Based on Simulation Zone Partitioning and Dual-Variable Hierarchical Assimilation', *Remote Sensing*, 17: 386.

He, Linyun, Luke Rhodes-Leader, and Eunhye Song. 2024. "Digital Twin Validation with Multi-Epoch, Multi-Variate Output Data." In *2024 Winter Simulation Conference (WSC)*, 347-58. IEEE.

Hua, Edward Y, Sanja Lazarova-Molnar, and Deena P Francis. 2022. "Validation of Digital Twins: Challenges and Opportunities." In *2022 Winter Simulation Conference (WSC)*, 2900-11. IEEE.

Lazarova-Molnar, Sanja. 2005. 'The proxel-based method: Formalisation, analysis and applications', Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek.

Lugaresi, Giovanni, Sofia Gangemi, Giulia Gazzoni, and Andrea Matta. 2023. 'Online validation of digital twins for manufacturing systems', *Computers in Industry*, 150: 103942.

Mertens, Joost, and Joachim Denil. 2024. 'Reusing model validation methods for the continuous validation of digital twins of cyber-physical systems', *Software and Systems Modeling*: 1-23.

Morgan, Lucy E, and Russell R Barton. 2022. 'Fourier trajectory analysis for system discrimination', *European Journal of Operational Research*, 296: 203-17.

Myers, Kary, Earl Lawrence, Michael Fugate, Claire McKay Bowen, Lawrence Ticknor, Jon Woodring, Joanne Wendelberger, and Jim Ahrens. 2016. 'Partitioning a large simulation as it runs', *Technometrics*, 58: 329-40.

Netter, Florian, Frank Gauterin, and Björn Butterer. 2013. "Real-data validation of simulation models in a function-based modular framework." In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 41-47. IEEE.

Overbeck, Leonard, Stephen C Graves, and Gisela Lanza. 2023. 'Development and analysis of digital twins of production systems', *International Journal of Production Research*: 1-15.

Papadopoulos, Alessandro Vittorio, and Alberto Leva. 2015. 'A model partitioning method based on dynamic decoupling for the efficient simulation of multibody systems', *Multibody System Dynamics*, 34: 163-90.

Peschlow, Patrick, Tobias Honecker, and Peter Martini. 2007. "A flexible dynamic partitioning algorithm for optimistic distributed simulation." In *21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07)*, 219-28. IEEE.

Petri, Carl Adam. 1962. 'Kommunikation mit automaten'.

Sargent, Robert G. 2013. 'Verification and validation of simulation models', *Journal of simulation*, 7: 12-24.

Schlesinger, Stewart. 1979. 'Terminology for model credibility', *Simulation*, 32: 103-04.

Xiang, Wei, Kan Yu, Fengling Han, Le Fang, Dehua He, and Qing-Long Han. 2023. 'Advanced manufacturing in industry 5.0: A survey of key enabling technologies and future trends', *IEEE Transactions on industrial informatics*, 20: 1055-68.

Zare, Ashkan, and Sanja Lazarova-Molnar. 2024a. "Modular Validation within Digital Twins: A Case Study in Reliability Analysis of Manufacturing Systems." In *2024 Winter Simulation Conference (WSC)*, 2903-14. IEEE.

Zare, Ashkan, and Sanja Lazarova-Molnar. 2024b. "Validation of Digital Twins in Labor-Intensive Manufacturing: Significance and Challenges." In *Procedia Computer Science*, 623-30.

Zeng, Xiang, Rajive Bagrodia, and Mario Gerla. 1998. "GloMoSim: a library for parallel simulation of large-scale wireless networks." In *Proceedings of the twelfth workshop on Parallel and distributed simulation*, 154-61.

AUTHOR BIOGRAPHIES

ASHKAN ZARE is a PhD student at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark. His research interests concern Modeling and Simulation, Data Analytics, and Machine Learning. His PhD project focuses on continuous validation of Digital Twins in manufacturing systems. His email address is zare@mmmi.sdu.dk.

SANJA LAZAROVA-MOLNAR is a Professor at both the Karlsruhe Institute of Technology and the University of Southern Denmark. Her research focuses on data-driven simulation, Digital Twins, and cyber-physical systems modeling, with an emphasis on reliability and energy efficiency. She develops advanced methodologies to optimize complex systems and leads several European and national projects in these areas. Prof. Lazarova-Molnar holds leadership roles in IEEE and The Society for Modeling & Simulation International (SCS), where she currently serves as SCS Representative to the Winter Simulation Conference (WSC) Board of Directors. She was Proceedings Editor for WSC in 2019 and 2020 and serves as Associate Editor for *SIMULATION: Transactions of The Society for Modeling and Simulation International*. Her email address is lazarova-molnar@kit.edu.