2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)
Munich, Germany, August 20-24, 2018

Digital Twin for Legacy Systems: Simulation Model Testing and Validation

Adnan Khan*, Martin Dahl*, Petter Falkman*, and Martin Fabian*

Abstract— **In this paper, an approach to incorporate a digital twin for legacy production systems is presented. Hardware-in-the-loop setups are routinely used by manufacturing companies to carry out virtual commissioning. However, manufacturing companies having online legacy production systems are still struggling to incorporate a digital twin due to the absence of verified and validated simulation models. Companies that use virtual commissioning as a part of their engineering tool chain, usually perform offline verification of the simulation model. This approach is typically based on visual inspection and is a tedious task as each aspect of the model has to be visually validated. For legacy systems, only assessing the behavior visually in the absence of updated documents can result in an incorrect simulation model, i.e. simulating incorrect behavior with respect to the specification. Due to this, such simulation models cannot be incorporated in the engineering tool chain, as the simulated results can lead to improper decisions and can even cause equipment damage. This paper presents a platform and an approach, based on model-based testing, that is a first step for manufacturing companies to incorporate a validated simulation model for existing online production systems that will serve as a digital twin.**

I. Introduction

In today's digital age, computers are doing most of the major tasks in manufacturing industries with more precision and efficiency providing products of better quality. Manufacturing industries in addition to the stress of producing quality products also struggle to meet physical commissioning deadlines of production systems. To make physical commissioning quicker, hardware-in-the-loop setups are being routinely used to carry out virtual commissioning. To implement a virtual commissioning setup, simulation and modeling engineers create a model of a real production system.

After modeling the production system, control logic is created and tested on the simulation model prior to implementing it on the real production system. The core advantage of carrying out virtual commissioning is reduced physical commissioning time. In addition, an accurate simulation model, a *digital twin*, which can be used throughout the real system's life span is created. This simulation model can then serve as a platform for testing future modifications. Having a digital twin in the engineering chain increases productivity, and assists systems and service applications during operation of the production systems [1].

This work has been carried out at the Wingquist Laboratory VINN Excellence Centre within the Production Area of Advance at Chalmers. It has been supported by ITEA3 Vinnova ENTOC (ref 2016-02716), and VR SyTeC (ref 2016-06204).

*Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden {kadnan, martin.dahl, petter.falkman, fabian}@chalmers.se

Currently, simulation engineers spend a lot of their time and effort on making detailed models of real production systems [2], [3]. Different approaches have been researched to ease the modeling task, e.g. automatic model generation [4], and using a single simulation model throughout the engineering cycle [5], [6].

However, in order to meet the challenges of the up-coming fourth industrial revolution, manufacturing companies having legacy production systems with functioning PLC code need a way to incorporate a validated digital twin in their existing tool chain. Unlike virtual commissioning of new production systems in which the simulation model is created with the help of complete documentation, legacy systems usually lack complete and updated engineering documents. Due to this, creating a robust simulation model that can serve as a digital twin for a legacy system is a difficult task.

As the documents related to legacy systems are either incomplete or rarely updated, the process of creating a simulation model for a legacy system must be supplemented by gathering data about the modifications made since the system was commissioned. Most of the data regarding modifications is typically collected manually by the engineers from the factory floor. Since the whole procedure of data collection from the shop floor is manual, human errors are inevitable. Also, due to to the use of natural language in the engineering documents auto-generation of the simulation model is typically intractable. Under such conditions, engineers have to rely on the manually collected data to create the simulation model, hence testing and validation of the created simulation model is a necessity.

Due to the problems associated with legacy systems, assessing behavioral equivalence by visual inspection is not enough, since minor but important errors can be overlooked by the engineers. The visual inspection procedure can be further strengthened by formally testing important properties. However, testing every property is impossible, and even testing important properties of a simulation model formally can be a tedious and time consuming task. Still, depending on the production system, testing certain important aspects in combination with visual inspection would result in a more robust simulation model.

Model-based testing [7] is an approach in which a model of an implementation undergoes a series of tests aiming to uncover errors. A specification provides the basis of the tests and the implementation either fails or passes the specified behavior. In the case of a legacy system, the simulation model will be the implementation which undergoes testing. Model-based testing can either be carried out in an offline or online

978-1-5386-3593-3/18/$31.00 ©2018 European Union

manner. In offline testing, issues of state-space explosion and handling non-determinism are intrinsic [8]. Due to these issues and having an already physically commissioned production system with existing PLC code makes online testing a perfect fit. But to test scenarios simultaneously on an online system and a simulation model is not always suitable, as production disruption or other issues can arise. Therefore, the approach based on offline model-based testing seems more suitable to validate a simulation model's behavior. The literature does not, to the best of the authors' knowledge, include any approach that incorporates offline model-based testing for validation of legacy system simulation model.

A. Contribution

This paper describes an offline model-based testing approach to test and validate a legacy system's simulation model, based on a setup as in Fig. 1. In this approach, the specification model, which is expressed using the software *Sequence Planner* [6], initiates different sequences of actions in the simulation by sending a signal *Start Sequence*. On the completion of each action in the simulation model, a feedback is sent back in terms of *Sensor Outputs* to the specification model. The received feedback signal updates the output values related to each operation in the specification model. Finally, the output values received are then compared with the specified values to verify the conformance of the outputs. If the received output values are not as expected then the conformance fails with respect to the specification. Non-conformance can occur either due to faulty specification or faulty implementation. The problem related to faulty specification is quite probable in the case of legacy systems due to outdated documents and needs to adjusted. For new systems, non-conformance due to faulty implementation is an issue and needs to be adjusted manually.

The benefits gained by using the proposed approach are as follows:

- Incorporation of a digital twin for existing production systems
- Validated simulation models for future modifications
- Overall reduction in commissioning time for future modifications

B. Outline

This paper is structured in the following way: In Section II, a brief overview regarding current concepts of virtual commissioning and the proposed approach is given. In Section III, the IOCO testing relation is introduced in the context of model-based testing. In Section IV, an overview of the proposed approach is detailed with an implementation example. Section V concludes the paper with future work direction.

II. Current Concepts and the Proposed Approach

A. Virtual Commissioning

Virtual commissioning [2], [3] is a hardware-in-the-loop configuration used before real (physical) commissioning of a production system to test and verify the control logic. In the setup, real PLCs with implemented control logic are used to control simulation models to find faults. The behavior of the simulation model is observed and errors found are corrected before the PLC program is put into place on the shop-floor controlling the real production system. Due to prior testing of the PLC code in the virtual environment, several programming and logical errors found in the PLC code have already been corrected, hence overall commissioning time of the real production system is reduced.

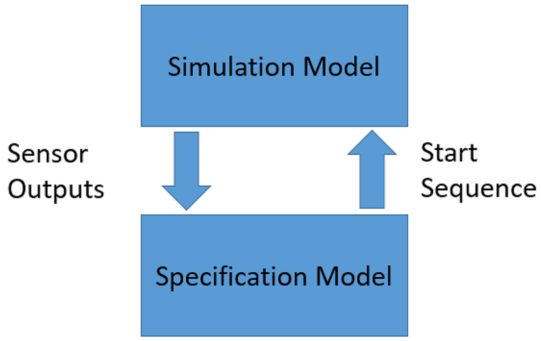

Fig. 1. Proposed Setup based on Offline model-based testing

B. Hybrid Commissioning

When a real controller is simultaneously connected to both a real production system and a simulation model, this setup is known as *hybrid commissioning* [9]. Unlike conventional virtual commissioning, hybrid commissioning consists of step-wise introduction of real components, resources, and or production lines of the production system replacing their virtual counterparts [10]. In hybrid commissioning, instead of testing the complete control logic of the whole production system at once, the control logic is tested gradually. Due to the gradual introduction of control, only the tested parts of production system are exposed to damage in the case of errors in the PLC code. Hence, this practice is considered more safe compared to conventional virtual commissioning.

C. Synchronous Simulation

Another concept where a real control system is connected to a simulation model and a real production system at the same time is known as *synchronous simulation* [11]. Unlike the hybrid commissioning, synchronous simulation is implemented after the real commissioning, i.e. when the production system is online already. The idea behind this concept is to compare the behavior of the simulation model with the real production system based on a production system description. Deviations found after the comparison of signals are then corrected. In [11] a few approaches are presented to implement synchronous simulation along with the requirements. Comparisons made using this approach are based on the production system description (usually expressed in terms of different drawings and natural language).

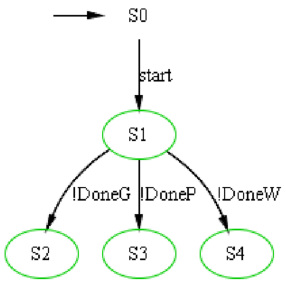

Fig. 2. Automaton as Operations

The synchronous simulation proposed in [11] can be classified as online model-based testing. The comparison made in the approach presented in [11] is based on the production system descriptions. But for legacy systems, the completeness and accuracy of such documents is questionable. In addition, if the synchronous simulation approach is applied to test the simulation model, the testing will be limited only to the operations run in real-time on the legacy system. Due to this, critical scenarios that can cause production disruption can only be tested if they occur naturally in real-time. Hence, this limitation on the scope of testing might pose questions on the reliability and robustness of the tested simulation model.

D. Proposed Approach

The setup shown in Fig 1 provides the basis of the proposed approach, which is based on offline model-based testing. Compared to synchronous simulation, which is based on visual inspection of the simulated behavior, this approach validates the correct input-output assignments based on conformance testing in addition to visual inspection. Due to conformance testing, modeling errors overlooked during visual inspection can be identified in a formal manner. Hence, resulting in a more robust simulation model.

Due to offline testing, the proposed approach does not suffer from the limitations posed by synchronous simulation. Therefore, many low probability and high risk scenarios that can cause production disruption or even damage to the equipment can be tested in a safe environment.

In this offline setup, different specified scenarios modeled in the specification model can be triggered in the simulation model. The signal to start any operation can be sent as an input to the simulation model and after the completion of each operation, a feedback is received as an output from the sensors in the simulation model. The received feedback from the simulation model is then compared with the specified outputs of the executed operation. The comparison reveals whether the test was either a success, in the case of positive conformance relation between the specification and the simulation model, or a failure.

III. Model-Based Testing

Manufacturing industries currently use a number of more or less informal tools, such as Piping and Instrumentation Diagrams, verbal descriptions of processes, and sketches [12], to specify production system behavior. These tools are then used to test and validate the behavior of real manufacturing production systems.

Due to the heterogeneity of the formal tools and the use of verbal descriptions to specify production systems, it is hard to generate robust and complete models of the production system automatically. Therefore, the simulation models are typically built manually.

Manufacturing production systems and their logical behavior can often be beneficially modeled as *discrete event systems* [13]. Discrete event systems evolve dynamically on the occurrence of events, while at each time instant occupying a specific state where certain conditions hold. Using discrete event formalisms [13], many formal approaches have been developed to implement and test the behavior of systems with respect to specifications.

One of those approaches is *model-based testing* [7], which is a formal approach to subject a model of an implementation to series of *tests* that try to falsify the specification according to which the implementation was created, in order to find faults in the implementation. To formalize this, the concept of *input-output conformance* (IOCO) was proposed by [14].

In the IOCO testing relation, the specification provides the basis for the behavior of the implementation in that it dynamically defines the outputs that the implementation is allowed to emit. If other than the specified outputs are emitted by the implementation, it is not IOCO with respect to the specification and some modifications need to be made. When the implementation is found to be non-IOCO, this can mean two things. Either the specification is correct and the implementation needs to be modified. Or, because of having a legacy system, the specification was designed from outdated documentation and it is in fact the implementation that is correct and the specification is faulty. This of course needs to be evaluated for each case separately, but if the specification is found to be faulty, then it has to be changed.

In IOCO the inputs and outputs are viewed as **events** and **outs**(i) and **outs**(s) represent the respective possible outputs of the implementation when in state i, and the specification when in state s, respectively. From the formal definition of the IOCO testing relation presented in [15], the key concepts relevant to IOCO are **traces**, **after** and **outs**. A trace t is a sequence of (input and output) **events**, which when executed establishes a path between two *states* in a system. A state defines the status of a system and can be reached if the trace leading to that particular state is enabled by the **events**. The **outs** in the IOCO definition are basically output **events**, that are computed from the state reached **after** the execution of a particular trace. The **traces** of the specification are used as a limit to check the implementation. Formally, this can be defined as:

$$\forall t \in \textbf{traces}(s) : \textbf{outs}(i \textbf{ after } t) \subseteq \textbf{outs}(s \textbf{ after } t) \qquad (1)$$

The formal definition (1) from [15] describes the IOCO relation such that, an implementation conforms to a specification, if and only if for all the **traces** in the specification the output events possible from the state i, reached by the

implementation after the trace, form a subset of the possible output events form the state s, reached by the specification after the same trace. Whenever this subset relation between the respective sets of output events exist, the implementation is said to be IOCO with respect to the specification, for that particular trace. If the implementation is IOCO with respect to the specification for all the traces defined by the specification, then the implementation is said to be IOCO with respect to the whole specification.

To further elaborate the concepts of **traces**, **events**, and **after**, an example shown in Fig 2 is considered. In the automaton, *start*, *!DoneG*, *!DoneP*, and *!DoneW* are input and output **events**, with the output events prefixed by an exclamation mark. The possible **traces** in the shown automaton are the empty trace of length zero, the *start* trace of length one, and the *start.!DoneG*, *start.!DoneP*, and *start.!DoneW* traces of length two. The **outs after** the trace *start*, which establishes a path between the initial state *S0* and state *S1*, are *!DoneG*, *!DoneP*, and *!DoneW*. For all other traces the possible **outs** in the given example is the empty set.

The above mentioned concepts **traces**, **outs**, and **after** related to the IOCO testing relation need to be appropriately applied to their respective conceptual equivalent in the simulation model. The concept of **traces** is related to *sequences* that occur dynamically depending on the inputs and outputs in an implementation. Similarly, **after** will be the post actions scenario in the implementation and **outs** will be the outputs from the sensors. These sensor outputs will then be compared to the expected outputs to check if the implementation is IOCO or not.

In practice, there are no events shared between the simulation model and the specification, but only Boolean signal values. The correlation between events and signal values will in this work be made as *False* is a subset of both *False* and *True*, while *True* is considered to be a subset of only *True*. Hence, if the specification output is *False* and the implementation output is *True* then the IOCO relation fails as the implementation output is not a subset of the specification output.

IV. IOCO for testing Simulation Model

To create a simulation model, different parts, components, and equipment are put together manually in a manner that it can perform the required *sequences of actions*. These sequence of actions once completed will give a feedback to the specification model in terms of *Sensors Outputs*.

In the proposed methodology, instead of checking each input-output combination for conformance, the specification model describes sequences of operations [16]. Creating operations eases the testing procedure as multiple input-output combinations that are part of an operation can be tested simultaneously. In the specification model, each sequence will have an input *Start Sequence* associated with it. This start event, once triggered will initiate a sequence of actions in the simulation model.

The initiated sequence of actions in the simulation model will activate different parts, equipment and sensors associated with the operation triggered from the specification model. After the completion of the actions related to the initiated operation, the status of the sensor outputs in the simulation model will always be set, either "true" or "false" (assuming binary sensors). The values of these sensors from the simulation model will then be fed back to the specification model.

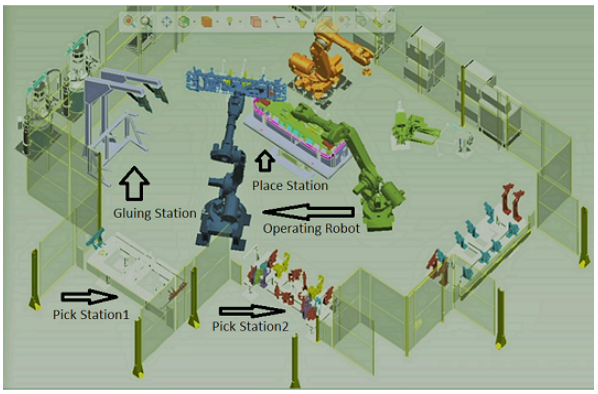

Fig. 3. Simulation Model

After receiving the feedback from the simulation model, i.e. updated values of the sensors, the specification model will update its own variables associated with the sensor outputs. For the purpose of identification, the output **events** in the specification model are labelled with a tag number or a name of the operation. In Fig. 2, *G*, *P*, and *W* are the tags of the associated operation. The updated values of executed operations are constantly monitored by the specification model and in case of non-conformance the executed sequence stops at the operation which is non-IOCO.

A. Use Case

1) Simulation Model: An application of the model-based testing has been carried out on a simulation model of a real cell, which exists in a production facility at one of Sweden's leading vehicle manufacturers. The cell is presented in Fig. 3. The equipment and work stations used for testing the input-output conformance involves only the two pick stations, one robot (labelled as Operating Robot), the gluing station, and the place station. These are pointed out in Fig. 3. From the perspective of IOCO, this simulation model is the implementation. In this implementation, the following operations have been modeled and are tested using the input-output conformance relation:

- Loading of the pick stations
- Picking a part from Pick Station 1
- Picking a part from Pick Station 2
- Gluing
- Pressing
- Placing the parts on Place Station

The above operations are also specified in the specification model for input-output conformance testing, which will be detailed later. When an operation is finished, it provides feedback to the specification model, the feedback provided

in this case will be for the signals *loading done*, *pick done 1*, *pick done 2*, *gluing done*, *press done*, and *place done*.

The series of actions in the cell begins with the simultaneous loading of parts into both pick stations. The loaded parts are then detected by the sensors installed on the pick stations, the part detected will trigger the Operating Robot to move and pick the part from Pick Station 2, which it then moves to the Gluing Station where glue is applied on the exposed area of the part.

Once the glue is applied, the part is taken to Pick Station 1 by the robot. At Pick Station 1, the robot places the part on top of the part that is already lying on Pick Station 1. Both parts are now pressed against each other for the glue to bind them together. After the pressing operation, the robot picks the pressed part and places it on the Place Station.

2) Sequence Planner for Writing Specifications: Sequence Planner (SP) [17] is a tool for modeling and analyzing automation systems. In SP, automation systems are modeled using operations and variables. It includes supporting algorithms for a variety of use cases relating to modeling, for instance synthesizing control logic, and visualizing complex operation sequences in different projections. By also providing online monitoring and control, it can be used in early phases to integrate model-based control design (based on operations) with simulation-based validation based on virtual commissioning.

In this example, SP is used to create the specification model, shown in Fig. 4, which monitors and tests the implementation (simulation model). The sequence starts with the simultaneous loading of parts on both the pick stations. Two operations in parallel i.e. *If2LoadPart* and *If1LoadPart0* in the specification model are related to the loading of parts.

Once the parts are loaded in the simulation model, the output values of the sensors at the pick stations get updated, these updated sensor values from the pick stations represent the **outs** of the implementation. The specification model updates the **outs** related to the loading operations after getting the feedback from the simulation model. Now, the values of **outs** related to the loading operations in the simulation model and the specification model are the same.

Now, to test the IOCO testing relation for the loading operation, we recall Definition 1, which states that for all **traces** in the specification, the **outs** of the implementation (simulation model) should be a subset of the **outs** of the specification. In this case, the loading operation is the trace executed and the **outs** of the simulation model and the specification model are the same, so the implementation is IOCO with respect to the specification, for the trace related to loading.

Applying the same concept to all operations, the IOCO relation was found to be valid for all operations except the pick operation from Pick Station 1. After the gluing operation, the robot takes the glued part at Pick Station 1, where the pressing operation is carried out successfully. But the implementation (simulation model) fails the IOCO testing relation in the next pick operation. This pick operation consists of segments *AR31PickLF2Seg20*, *AR31PickLFSeg30*,

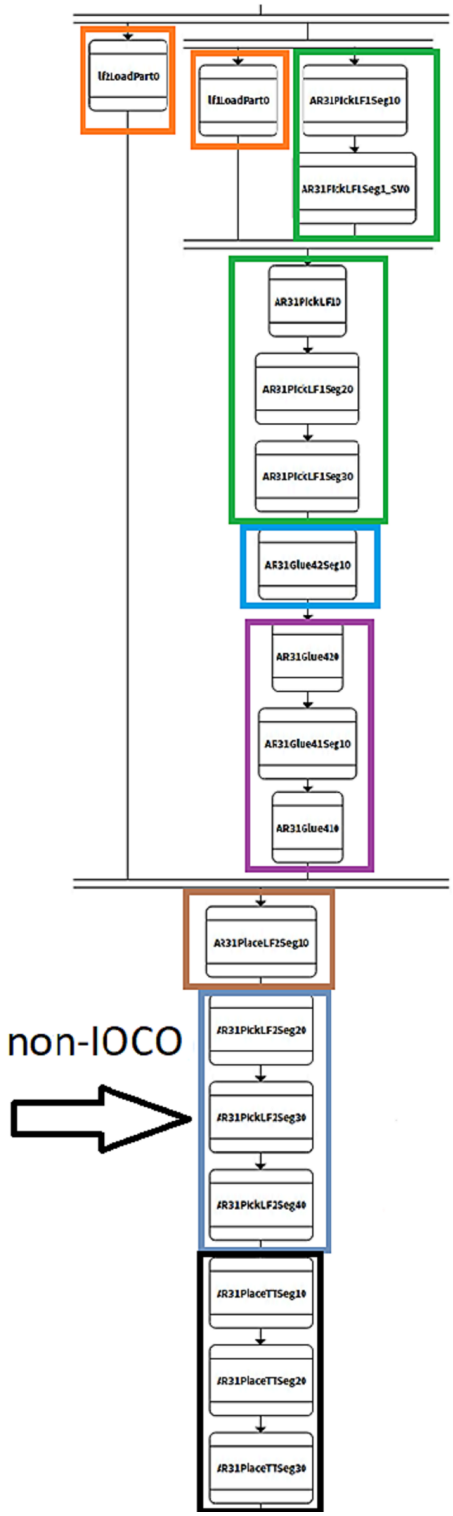

Fig. 4. Specification Model in Sequence Planner

and *AR31PickLFSeg40*, and is pointed out in Fig.4.

In this operation, the robot picks only the top part, leaving the bottom part at Pick Station 1. Due to this, the value of the sensor installed at Pick Station 1 remains *True*. However, the value of the related **outs** in the specification model for this operation is *False* as the specification expects that the two parts now glued together are simultaneously removed from the station. Hence, according to the definition of IOCO the stated operation is not IOCO, as the **outs** in the implementation is *True* and is not a subset of the **outs** in the specification, which is *False*. This non-IOCO operation causes the sequence to stop at its second to last step,which needs to be adjusted to make the implementation IOCO with respect to the specification. The adjustment is carried out manually by adding the missing actions in the pressing operation.

V. Conclusion

In this paper, a methodology of incorporating a digital twin for legacy systems based on model-based testing is outlined. The proposed methodology is applied on a simulation model to demonstrate the input-output conformance relation. For validation, any software using operations and variables can be used; in the described case study the tool Sequence Planner was used. Also, depending on the completeness of design documents, this approach can be extended to complex legacy systems. This work is an initial step to incorporate model-based testing in the field of virtual commissioning. Furthermore, this work has laid the foundation to use a model-based testing approach to test and validate the safety-PLC code in a virtual environment, which will be carried out in the future.

Another direction for future work is related to the automatic adjustment of a non-conforming implementation. If the implementation fails to conform to the specification, there are two possible scenarios. One scenario is related to faulty specifications. The IOCO relation assumes that the specification is correct, and in the case of non-conformance the implementation has to be changed rather than specification. But in the case of legacy systems this might not always be true, as the specification can be faulty due to outdated documents. To counter this, an algorithmic way to adjust the specification using *synthesis* [13], [18] could be used similar to the approach by [19]. Then, a non-conforming implementation due to faulty specification can be fully automatically adjusted. Depending on the requirement and the application, synthesis using the *supremal controllable sub-language* [20] or the *infimial controllable super-language* [21] can be carried out.

The second scenario is related to faulty implementation and at the moment the implementation has to be adjusted manually. In the future, we will examine different approaches to automatically adjust a faulty implementation by using synthesis or similar techniques.

References

[1] S. Boschert and R. Rosen, "Digital twin—the simulation aspect," in *Mechatronic Futures*. Springer, 2016, pp. 59–74.

[2] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," *Journal of Computational Design and Engineering*, vol. 1, no. 3, pp. 213–222, 2014.

[3] P. Hoffmann, R. Schumann, T. M. Maksoud, and G. C. Premier, "Virtual commissioning of manufacturing systems a review and new approaches for simplification." in *24th European Conference on Modelling and Simulation (ECMS 2010)*, 2010, pp. 175–181.

[4] O. Mathias, W. Gerrit, D. Oliver, L. Benjamin, S. Markus, and U. Leon, "Automatic model generation for virtual commissioning based on plant engineering data," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 635–11 640, 2014.

[5] S. Seidel, U. Donath, and J. Haufe, "Towards an integrated simulation and virtual commissioning environment for controls of material handling systems," in *Proceedings of the winter simulation conference*. Winter Simulation Conference, 2012, p. 252.

[6] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, and P. Falkman, "Integrated virtual preparation and commissioning: supporting formal methods during automation systems development," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1939–1944, 2016.

[7] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[8] M. Mikucionis, K. G. Larsen, and B. Nielsen, "T-UPPAAL: Online model-based testing of real-time systems," in *Proceedings of the 19th IEEE international conference on Automated software engineering*. IEEE Computer Society, 2004, pp. 396–397.

[9] Z. Liu, C. Diedrich, and N. Suchold, *Virtual Commissioning of Automated Systems*. INTECH Open Access Publisher, 2012.

[10] S. Dominka, F. Schiller, and S. Kain, "Hybrid commissioning–speeding-up commissioning of field bus driven production plants," in *Mechatronics, ICM2007 4th IEEE International Conference on*. IEEE, 2007, pp. 1–6.

[11] S. Kain, S. Dominka, M. Merz, and F. Schiller, "Reuse of HIL simulation models in the operation phase of production plants," in *Industrial Technology, 2009. ICIT 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[12] G. Frey and L. Litz, "Formal methods in PLC programming," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2431–2436.

[13] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, ser. SpringerLink Engineering. Springer US, 2009.

[14] G. Tretmans, "Test generation with inputs, outputs and repetitive quiescence, 1996," *URL http://doc. utwente. nl/65463*, vol. 46, 1996.

[15] C. Gregorio-Rodríguez, L. Llana, and R. Martínez-Torres, "Input-output conformance simulation (iocos) for model based testing," in *Formal Techniques for Distributed Systems*. Springer, 2013, pp. 114–129.

[16] K. Bengtsson, *Flexible design of operation behavior using modeling and visualization*. Chalmers University of Technology,, 2012.

[17] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, and P. Falkman, "Sequence planner: Supporting integrated virtual preparation and commissioning," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5818 – 5823, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317309047

[18] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[19] H. Marchand, J. Dubreil, and T. Jéron, "Automatic testing of access control for security properties," in *Testing of Software and Communication Systems*. Springer, 2009, pp. 113–128.

[20] R. Malik, K. Åkesson, H. Flordal, and M. Fabian, "Supremica-–an efficient tool for large-scale discrete event systems," in *IFAC World Congress, Tolouse, France*, 2017.

[21] L. Ricker, S. Lafortune, and S. Genc, "DESUMA: A tool integrating GIDDES and UMDES," in *2006 8th International Workshop on Discrete Event Systems*, July 2006, pp. 392–393.