Available online at www.sciencedirect.com
ScienceDirect
Procedia Manufacturing 00 (2017) 1–8
Procedia MANUFACTURING
www.elsevier.com/locate/procedia

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

Automatic modeling and simulation of robot program behavior in integrated virtual preparation and commissioning

M. Dahl[a,], K. Bengtsson[a], M. Fabian[a], P. Falkman[a]

[a]*Chalmers University of Technology, Department of Signals and Systems, 412 96 Göteborg, Sweden.*
{martin.dahl, kristofer.bengtsson, fabian, petter.falkman}@chalmers.se

Abstract

This paper presents a method where the behavior of a robot cell is automatically modeled based on existing robot programs and a simulation model of the cell. Robot programs from the shop floor are uploaded into a virtual manufacturing tool, and a formal model is then generated from the robot programs. Then, control logic is automatically calculated, and the fastest possible execution order is found by using the generated model to formulate an optimization problem. The result is continuously analyzed and validated by simulation in the virtual manufacturing tool.

Keywords: virtual commissioning, virtual manufacturing, robot programming, automation development

1. Introduction

Production systems evolve during their life cycle. On-line changes of the control code, fine tuning, and system maintenance, gradually change the system from what was originally prepared off-line. With the ongoing heavy investment in virtual manufacturing [1], it is important that the virtual manufacturing models (VMMs) evolve together with the system.

Being able to simulate existing production systems using existing virtual manufacturing models is crucial for example when introducing new product types into an existing system. Updating a VMM to accommodate *physical* changes is already being done with laser scanning into point clouds [2], but proper tools are also needed to work with the system *logic*.

Off-line robot programming has since long been used in industry, and most of today's virtual manufacturing tools support both downloading programs to robots and uploading them back into a virtual model. However, the robot programs usually contain handshaking (e.g. interlocking) with an external control system (e.g. a PLC), so before the introduction of virtual commissioning (VC) — which enables a virtual model of the system to be controlled by an external control system [3] (see Section 2) — simulating this handshaking of the uploaded robot programs required manual intervention.

Most VMMs in industry today were created before the advent of VC. In plants with many hundreds if not thousands of robots (e.g. in the automotive industry) manual intervention needs to be kept at a minimum. For production cells that are mainly robot based, this work proposes a method that instead of performing a costly migration of existing VMMs to full VCs, a formal model of the system including the handshaking with the control system can be generated automatically. This formal model contains all possible combinations of executions, and can be used to visualize, study and simulate the system. A formal model also enables verification and optimization of the system behavior. For stations that contain mostly robots, this approach requires very little human intervention. Models for other devices (e.g. clamps, turn tables) can then be added manually, depending on how detailed the preparation work needs to be. With the addition of VC support in virtual manufacturing tools in recent years, the formal model can be used to control the simulation, in effect acting as a high-level control system. This enables far greater opportunities to simulate things like disruptions, running maintenance programs, and executing programs for different product types in sequence compared to the traditional way of creating sequences for different cases manually. Figure 1 shows the software Sequence Planner [4], which hosts the generated models, controlling the virtual manufacturing software Process Simulate[1]. The depicted production cell with four robots is used as an example throughout Sections sections 4 to 6.

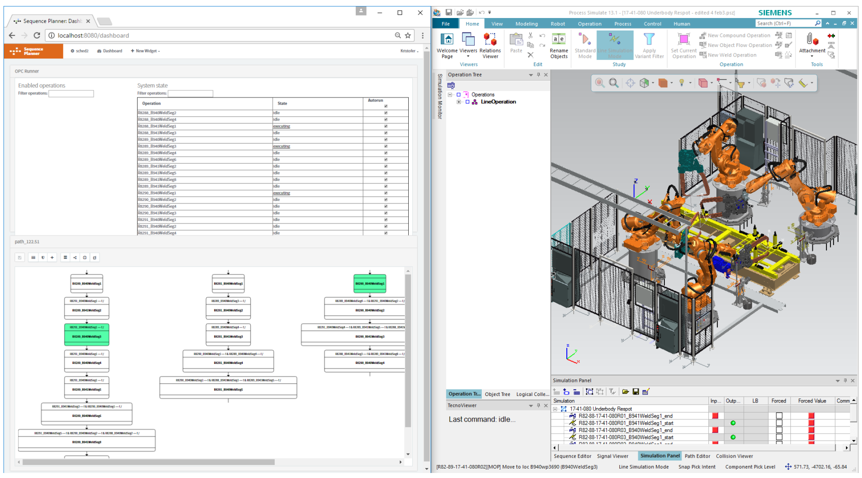


Fig. 1: Graphical user interface of the software Sequence Planner to the left shown controlling a simulation of four robots in a virtual manufacturing software to the right. Currently executing operations are shown in green color.

This paper is a small step in realizing an integrated virtual preparation and commissioning framework (see Section 2.1), where modern virtual manufacturing technology is integrated with model based preparation in the earlier phases. Being able to import logic from existing systems is key to transition into an integrated virtual preparation and commissioning framework.

The organization of this paper is as follows: A brief introduction to virtual manufacturing and commissioning is given in Section 2. In Section 3, the formal modeling language used is briefly described. Section 4 describes how a formal model is generated from robot programs, and how control logic in synthesized. The synthesized control logic is used for control of a simulation in Section 5, where it is also shown how optimization can be used to fulfill some interesting criterion, like minimum makespan. Section 6 contains a brief example of how a change might be handled within the framework.

[1]Siemens PLM, http://www.plm.automation.siemens.com/, 2017

2. Virtual Manufacturing and Commissioning

Today's virtual manufacturing tools can upload, and simulate, robot programs from the shop floor. They also allow the simulated devices to be controlled using a real control system. This setup is commonly referred to as *virtual commissioning* (VC) [3]. The main motivation to use VC as defined in the previous paragraph is to reduce testing and integration time during development. A study performed in Denmark [5] estimates that VC reduces the number of on-site man hours during installation by 50%. This is achieved by being able to test and integrate the control system before the physical production system is completely installed. Using a simulation model of the production system, undesirable behavior can be detected well ahead of physical installation. In addition, VC enables tests that would be prohibitively expensive or even impossible to run on a physical system (e.g. unlimited virtual parts). The simulation model also makes it possible to test changes to the production system while it is running and allows last minute changes with less concern about their impact on the physical system [6, 7, 3]. Even though software and vendor support for VC is growing, there are a number of issues that hinder its broader adoption. Because VC uses the real control system, the simulation models need to be specified at the level of sensors and actuators [3]. Creating the simulation models at such a detailed level requires both an extensive modeling effort and multi-disciplinary (robot, control systems, and simulation) expertise which may not be readily available in-house. Although research is being conducted on reducing the modeling effort required by generating physical/kinematic parts of the models (e.g. [8, 9]) and the logical parts (e.g. [10, 11]), this is not yet general enough to be applied in all cases.

2.1. Integrated Virtual Preparation and Commissioning

Current methodology applies VC as the last step in the automation engineering phase, to reduce the physical commissioning time of the control system. However, VC can provide value during the entire automation engineering process, as shown by [12], where a framework called *Integrated Virtual Preparation and Commissioning* (IVPC) is introduced. A formal model of the control logic, combined with having the same virtual models of the production system shared between the preparation, control system implementation, and VC phases, enables both early validation of high level control as well as a possibility to perform early optimization based sequencing. The framework is based on rapid continuous iterations that can be divided into four categories:

VMM updates. Whenever a new device is added or changed in the VMM, the formal model needs to be updated with new operations and specifications, either automatically or manually. Ideally devices should contain a formal representation just as they contain geometries and kinematics.

VMM generation. Just as the VMM can be used to generate parts of the formal model, parts of the VMM can be generated (and by extension, re-generated when changes occur) based on the formal model of the system. An example of this is generating robot programs using path planning software.

Verification - "Did we build the thing right?". By having a formal model underpinning the automation system, it is a natural part of the work flow to continuously *verify* that specifications are fulfilled. This is done by proving that properties about the system hold, e.g. correct resource allocation or guaranteeing cycle time limits.

Validation - "Did we build the right thing?". Because of the reliance on synthesis of the high-level control logic, whenever all specifications are satisfied, development is at the stage where *validation* can be performed by simulating the system. By using the VC facilities provided by modern virtual manufacturing tools, the synthesized control logic can be used to control the VMM. This gives the ability to *see* how the system behaves, giving a feel for whether the specifications given earlier make sense. Especially when optimization is involved in the design, it can be hard to predict the behavior of the final system.

This work adds the orange box to the left in Figure 2: extracting logic from an existing VMM that contains robots and other devices, including robot programs uploaded from the shop floor. In the following sections the other areas of the IVPC framework are briefly described in the context of visualizing current system behavior.

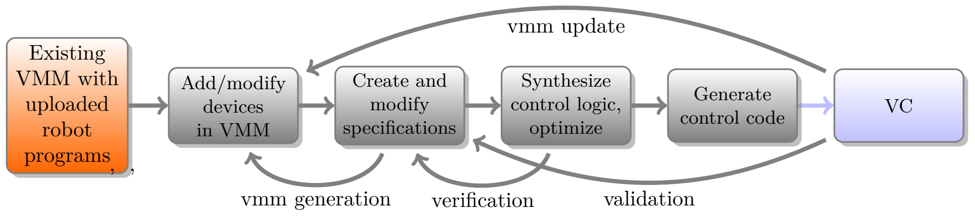
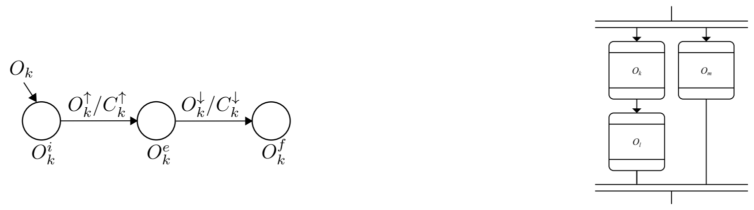

vmm update
Existing VMM with uploaded robot programs
Add/modify devices in VMM
Create and modify specifications
Synthesize control logic, optimize
Generate control code
VC
vmm generation
verification
validation

Fig. 2: Workflow cycles within IVPC

3. Sequences of Operations (SOPs)

A central abstraction in IVPC is the *operation*. Informally, an operation defines an action that can be performed by a resource. This informal definition of an operation fits well with most virtual manufacturing tools, where different types of operations (e.g. processes, movement, product flows) are created, which enables a clear mapping between the formal definition presented below and what is specified in VMMs.

The operation model can be represented formally by *extended finite automata* (EFA) [13], a generalization of automata including guards and actions associated with the transitions. A transition in an EFA is enabled if and only if its corresponding guard formula evaluates to true; when the transition is taken, a set of variables may subsequently be updated. Formal tools for EFAs are available (see [14]), making both formal verification and synthesis applicable directly to the suggested EFA models.


O_k
$O_k^{\uparrow}/C_k^{\uparrow}$
$O_k^{\downarrow}/C_k^{\downarrow}$
O_k^i
O_k^e
O_k^f
O_k
O_m
O_l

(a) EFA for operation O_k. (b) SOP containing operations O_k, O_l, and O_m.

Fig. 3: An operation and a SOP.

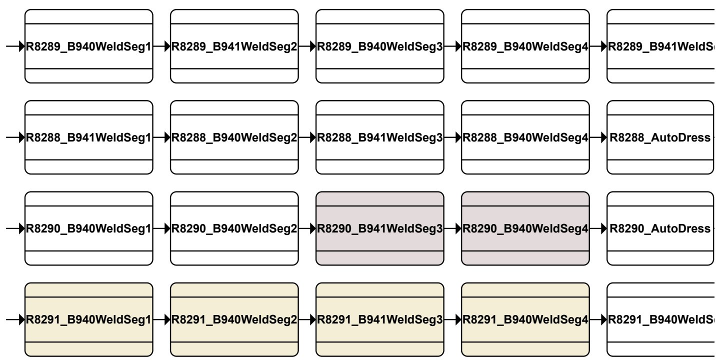
The transition for the operation O_k (depicted in Figure 3a) from the initial location O_k^i to the execution location O_k^e is enabled when the *precondition* $C_k^{\uparrow}$ is satisfied, after which the transition can be fired and the start event $O_k^{\uparrow}$ occurs. In the same way, the completion event $O_k^{\downarrow}$ can only occur when the *postcondition* $C_k^{\downarrow}$ is fulfilled. For operations that need to be repeated, the operation model in Figure 3a is modified. A reset transition is then introduced, from O_k^e to O_k^i, such that the operation O_k can be repeated from its initial location, O_k^i, when a reset event, $O_k^{\leftarrow}$, has been fired. This can happen when the *reset condition* $C_k^{\leftarrow}$ is satisfied.

Temporal relations between operations are visualized in the formal and graphical SOP language. The SOP language has operators for combining operations in hierarchies, sequences, parallel execution, arbitrary order, and alternatives. Figure 3b shows an example of a SOP where the operation

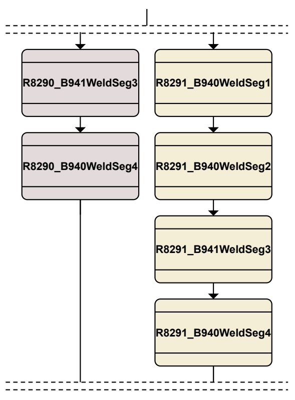
O_k precedes O_l and O_m is parallel to (independent of) of both O_k and O_l. An in-depth description of the formal operation model and the SOP language is given in [15, 16].

4. Generation of operations, specifications, and control logic

At the studied facility, robot programs are organized in way where a main loop determines the control flow for each robot. Based on handshaking with the control system, different subprograms are called. For example, a zone booking handshake might look like `WaitSignal AllocateZone1`, followed by some welding commands, later followed by `WaitSignal ReleaseZone1`. The robot programs, uploaded from the shop floor, can be imported into a virtual manufacturing tool, which generates a tree of operations (e.g. welding, point-to-point). Note that not all commands are recognized by the virtual manufacturing tool, for instance the commands for zone allocation described above will not be handled. However, the original robot code is kept in the VMM which makes it possible to further parse them for handshaking with the control system.



(a) Part of a straight sequence for four different robots.



(b) A shared zone.

Fig. 4: Two specifications extracted from a set of robot programs. The color coding indicates which robot operations in the main sequence corresponds to which operations in the shared zone.

The operations generated from the robot programs in the VMM are mapped to corresponding EFA operations. From the operations and the extracted handshaking commands, specifications are created as SOPs. Figure 4 show two generated specifications from the production cell depicted in Figure 1. To the left: the main sequence for each robot, to the right: an example of a mutual exclusion specification (a shared zone). The specification for the shared zone is modeled in the SOP language as “arbitrary order”, meaning that either the left sequence gets to run first, allowing robot R8290 to access the shared zone, or the right sequence in its entirety, allowing robot R8291 access. The figures are color coded to indicate which robot operations in the main sequence correspond to which operations in the shared zone.

In this stage, it may be necessary to add some details manually to the model, details that cannot be automatically extracted from the robot programs. For instance, it can be the case that the robot program waits for a certain signal given by the PLC. If this signal is triggered by something not found in any of the robot programs (e.g. waiting for an external device), this needs to be manually specified, since the PLC-programs are not part of the input data. The natural way of handling this within IVPC is either by adding an additional specification like the ones generated above, or if more details are necessary, adding the missing devices to the VMM with their own formal model.

Not only is it convenient to obtain correct control logic as the solution to a synthesis problem, it is also faster and less error-prone than doing it manually. The operations and the specification SOPs generated in the previous section are used to formulate a supervisor synthesis problem. Using the method described in [17], the solution to this synthesis problem can be obtained as additional conditions on the preconditions of the EFA operations. The synthesis is performed by the software *Supremica* [14]. Figure 5 shows the result for a single operation, which can, after synthesis, no longer start whenever *R8289_B941WeldSeg8* is in its executing location.

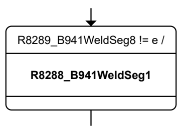


Fig. 5: An operation with an extra condition added by a synthesis algorithm.

5. Iteratively simulating the system and improving it using optimization

The formal model is not only used for verification and optimization, it can also be used for control. The EFA operations with additional synthesized guards implicitly contain all possible executions of the system. By exploiting the possibility to connect an external control system to the VMM, just as one would have done when performing VC, the generated model can be used to control the system. In Figure 1 the software Sequence Planner [4], which hosts the generated EFA model, controls the virtual manufacturing software Process Simulate. The communication is handled via an OPC UA server and uses signals which are automatically generated within the VMM. This means that there is no manual step involved in setting up the simulation other than starting an OPC UA server and loading the correct signal names. Operations can start whenever their precondition is satisfied, which is done by setting the operations' corresponding "start signal" to true and moving to its executing location. The simulation of the VMM sets the operations' "finished signal" to true when the operation has finished, which allows the operation to move to its finished location (and be reset).

The simulation can provide analysis of the generated system. However, starting operations on a first-ready, first-serve basis may not be optimal. By finding an optimal coordination it is possible to get an understanding about whether the system can be improved in terms of various objectives, like makespan or energy efficiency. Optimization of the system is a natural form iteration in IVPC and as such the result of an optimization should be possible to simulate immediately. If the VMM operations also contain their respective duration (ignoring any simulation accuracy details for now), optimization w.r.t. makespan can produce a number of "good" orderings of the zone allocations.

Since the model of the supervised system typically contains millions of states, an optimization problem is formulated and solved, to find the shortest total time (λ in (1)). While the system described in this paper is very simple, a more complex system of SOPs have been previously been optimized in [18]. In (1) the start times ($s_i, \forall i \in 1..N$) of N operations are the decision variables. To simplify expressing the constraints, the end times ($e_i = s_i + d_i, \forall i \in 1..N$) of the operations are also included, where d_i refers to the duration of each operation. Sequence constraints are expressed using pairs of indexes for the start and end times in P. Similarly, mutual exclusion w.r.t. zone allocation is expressed using the pairs in M. Using pairs of indexes in F, operations that are performed in sequence within the same arbitrary order specification are forced to start right after each other. Finally, the two last constraints in (1) describes that operations can only start executing when another operation finishes (except at time $t = 0$), to model the event based nature of the system, and that start times need to be ≥ 0. When solving, some additional constraints are added to guide the search; these are not included in (1).

$$
\begin{aligned}
\underset{s_i,\ \forall i \in 1..N}{\text{minimize}} \quad & \lambda \\
\text{subject to} \quad & e_i = s_i + d_i, && \forall i \in 1..N \\
& e_i < \lambda, && \forall i \in 1..N \\
& e_{p_i} \le s_{p_j}, && \forall \langle p_i, p_j \rangle \in P \\
& e_{m_i} \le s_{m_j} \lor e_{m_j} \le s_{m_i}, && \forall \langle m_i, m_j \rangle \in M \\
& e_{f_i} = s_{f_j}, && \forall \langle f_i, f_j \rangle \in F \\
& s_i = 0 \lor \exists e_j = s_i, && \forall i, j \in 1..N, i \neq j \\
& s_i \ge 0, && \forall i \in 1..N
\end{aligned}
\tag{1}
$$

The solution to the optimization problem is a vector of operation start times. Using the relation identification algorithms described in [19], this vector can be used to identify any additional preconditions required to ensure this particular order of execution. Figure 6 shows the same main sequences for the four robots as in Figure 4, with the additional preconditions added to them. For example, the top right operation *R8288_B940WeldSeg2* (top row, second operation from the left) now has the additional precondition *R8291_B941WeldSeg3 == f* & *R8289_B941WeldSeg5 == f*, meaning that for *R8288_B940WeldSeg2* to start execution, both *R8291_B941WeldSeg3* and *R8289_B941WeldSeg5* must be in their *finished* location, i.e. completed. Again, the system can be simulated as described in Section 5, to study the effects of the optimization.

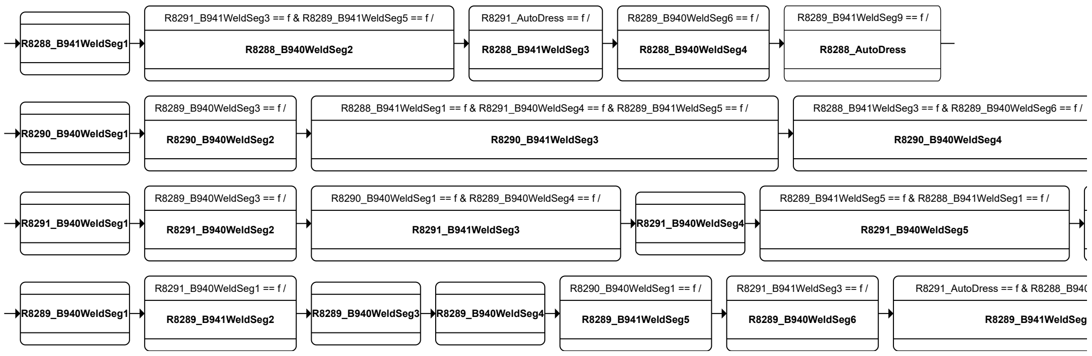

R8288_B941WeldSeg1
R8291_B941WeldSeg3 == f & R8289_B941WeldSeg5 == f /
R8288_B940WeldSeg2
R8291_AutoDress == f /
R8288_B941WeldSeg3
R8289_B940WeldSeg6 == f /
R8288_B940WeldSeg4
R8289_B941WeldSeg9 == f /
R8288_AutoDress
R8290_B940WeldSeg1
R8289_B940WeldSeg3 == f /
R8290_B940WeldSeg2
R8288_B941WeldSeg1 == f & R8291_B940WeldSeg4 == f & R8289_B941WeldSeg5 == f /
R8290_B941WeldSeg3
R8288_B941WeldSeg3 == f & R8289_B940WeldSeg6 == f /
R8290_B940WeldSeg4
R8291_B940WeldSeg1
R8289_B940WeldSeg3 == f /
R8291_B940WeldSeg2
R8290_B940WeldSeg1 == f & R8289_B940WeldSeg4 == f /
R8291_B941WeldSeg3
R8291_B940WeldSeg4
R8289_B941WeldSeg5 == f & R8288_B941WeldSeg1 == f /
R8291_B940WeldSeg5
R8289_B940WeldSeg1
R8291_B940WeldSeg1 == f /
R8289_B941WeldSeg2
R8289_B940WeldSeg3
R8289_B940WeldSeg4
R8290_B940WeldSeg1 == f /
R8289_B941WeldSeg5
R8291_B941WeldSeg3 == f /
R8289_B940WeldSeg6
R8291_AutoDress == f & R8288_B94
R8289_B941WeldSeg

Fig. 6: Part of an optimization result, displayed as extra preconditions per robot sequence.

6. Introducing changes

Perhaps the main advantage of using the presented framework with formal models is when changes are to be introduced. Consider the following example: If needed, tip-dressing is done at the end of each cycle for the robots in the cell (the *AutoDress* operations). Assume that the optimization shown above indicates that one of the robots is always waiting for a few seconds, enough to be able to perform the tip-dressing operation during waiting. By simply removing the tip-dress operation from the generated specification in Figure 4a it is possible to let the optimizer decide when it should be executed to minimize the total makespan. When specifications exist that ensures correct behavior, complexity can be handled by calculation rather than manual work. But getting the specifications right is not always an easy task, which is why the ability to continuously validate the results through simulation is needed.

7. Conclusion

The method presented in this paper has been applied to several existing real world production cells from the automotive industry. We have demonstrated the ability to use uploaded robot programs to generate a formal model of the control flow of the robots in production cell. This model can be used to control a real time simulation of the production system, which gives a simulation engineer the ability to simulate robot programs using their actual behavior without having to migrate the VMM to full VC. This provides greater ability to simulate things like disruptions and executing multiple cycles compared to how this has traditionally been performed.

Future work involves extracting more logic, both from the current control system (i.e. most likely a PLC) but also from the virtual manufacturing models (e.g. creating specifications based on intersections of sweep volumes to ensure that collisions cannot happen).

Acknowlegments

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 Vinnova FFI VIRTCOM (ref 2014-01408), and ITEA3 Vinnova ENTOC (ref 2016-02716).

References

[1] E. Filos, Four years of ‘factories of the future’in europe: achievements and outlook, International Journal of Computer Integrated Manufacturing 30 (1) (2017) 15–22.

[2] E. Shellshear, R. Berlin, J. S. Carlson, Maximizing smart factory systems by incrementally updating point clouds, IEEE computer graphics and applications 35 (2) (2015) 62–69.

[3] C. G. Lee, S. C. Park, Survey on the virtual commissioning of manufacturing systems, J. Comput. Des. Eng. 1 (3) (2014) 213–222.

[4] Sequence Planner team, Sequence Planner, available from `https://www.github.com/kristoferB/SP` (2017).

[5] N. Shahim, C. Möller, Economic justification of virtual commissioning in automation industry, in: Winter Simulation Conference (WSC), 2016, IEEE, 2016, pp. 2430–2441.

[6] P. Hoffmann, T. Maksoud, R. Schumann, G. Premier, Virtual Commissioning of Manufacturing Systems a Review and New Approaches for Simplification, Proc. 24th Eur. Conf. Model. Simul. 2 (Cd).

[7] S. C. Park, M. Chang, Hardware-in-the-loop simulation for a production system, Int. J. Prod. Res. 50 (8) (2012) 2321–2330.

[8] M. Chang, M. Ko, S. C. Park, Fixture modelling for an automotive assembly line, Int. J. Prod. Res. 49 (15) (2011) 4593–4604.

[9] M. Barth, A. Fay, Automated generation of simulation models for control code tests, Control Eng. Pract. 21 (2) (2013) 218–230.

[10] H.-T. Park, J.-G. Kwak, G.-N. Wang, S. C. Park, Plant model generation for PLC simulation, Int. J. Prod. Res. 48 (5) (2010) 1517–1529.

[11] S. C. Park, M. Ko, M. Chang, A reverse engineering approach to generate a virtual plant model for PLC simulation, Int. J. Adv. Manuf. Technol. 69 (9-12) (2013) 2459–2469.

[12] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, P. Falkman, Integrated virtual preparation and commissioning: supporting formal methods during automation systems development, IFAC-PapersOnLine 49 (12) (2016) 1939–1944.

[13] M. Skoldstam, K. Akesson, M. Fabian, Modeling of discrete event systems using finite automata with variables, in: Decision and Control, 2007 46th IEEE Conference on, IEEE, 2007, pp. 3387–3392.

[14] K. Åkesson, M. Fabian, H. Flordal, R. Malik, Supremica-an integrated environment for verification, synthesis and simulation of discrete event systems, in: Discret. Event Syst. 2006 8th Int. Work., IEEE, 2006, pp. 384–385.

[15] B. Lennartson, K. Bengtsson, C. Y. Yuan, K. Andersson, M. Fabian, P. Falkman, K. Åkesson, Sequence planning for integrated product, process and automation design, IEEE Transactions on Automation Science and Engineering 7 (2010) 791–802.

[16] K. Bengtsson, Flexible design of operation behavior using modeling and visualization, PhD Thesis at Chalmers University of Technology, 2012, 231.

[17] S. Miremadi, B. Lennartson, K. Åkesson, A BDD-based approach for modeling plant and supervisor by extended finite automata, Control Syst. Technol. IEEE Trans. 20 (6) (2012) 1421–1435.

[18] N. Sundström, O. Wigström, P. Falkman, B. Lennartson, Optimization of operation sequences using constraint programming, IFAC Proceedings Volumes 45 (6) (2012) 1580–1585.

[19] K. Bengtsson, P. Bergagard, C. Thorstensson, B. Lennartson, K. Akesson, C. Yuan, S. Miremadi, P. Falkman, Sequence planning using multiple and coordinated sequences of operations, IEEE Transactions on Automation Science and Engineering 9 (2) (2012) 308–319.