STRATIFICATION OF TIMED PETRI NETS AT THE EXAMPLE OF A PRODUCTION PROCESS

Carlo Simon and Stefan Haag and Lara Zakfeld
Fachbereich Informatik
Hochschule Worms
Erenburgerstr. 19, 67549 Worms, Germany
E-Mail: {simon,haag,zakfeld}@hs-worms.de

KEYWORDS

Stratified Modeling and Simulation, Petri Net Folding, Timed Dynamic Systems, Process Management

ABSTRACT

Timed dynamic systems can be modeled and simulated with Petri nets using very different approaches. In Clock Pulse Models (CPM), one marking represents exactly one moment in time and all enabled transitions fire simultaneously according to a global clock. This allows for real-time observation of the modeled system during a simulation run. Since simulation time is proportional to observed real-time, such an approach barely scales concerning the observed time horizon. If not the observable behavior over time is of interest but the final simulation result, Event Triggered Models (ETM) can overcome this limitation as has been shown in former publications. A time-lapse model accelerates simulation runs by focusing on the moments state chances occur.

CPM and ETM, however, share one disadvantage: the models' sizes tend to be proportional to the number of events that may occur simultaneously in the real world. Hence, they scale barely with the number of modeled entities. This unsolved scaling problem is addressed in this paper and pursues the idea of folding similar subnets in temporal layers called strata. The result is a Stratified Simulation Model (SSM), a small, flexible model that scales well concerning the number of modeled entities. SSM are derivates of CPM, since they still allow for nearly-real-time observation of the real world. Moreover, these models ease the visualization of the simulation in dashboards. The approach is explained at the example of a small production process.

INTRODUCTION

It is the nature of simulation that it is performed for systems that are too complex for a human mind to understand them entirely. Computational complexity considers the two dimensions time and space (Arona and Barak 2009). Time complexity classifies the time needed to execute an algorithm on an input for a given parameter such as the number of elements in an array. Space complexity classifies the memory amount needed.

Batty and Torrens (2001) apply this idea to the term modeling complexity which links it to simulation. In simulation research, we must handle both kinds of complexity to make simulation methods applicable.

The two parameters that influence the complexity of a simulation run are the complexity of the modeled system and the observation time. Popovics and Monostori (2016) distinguish between structural and software complexity measures. The structural complexity is derived from the number of elements a modeled system consists of. Both structural and software complexity influence the memory space complexity of a simulation problem.

Also, the model complexity must be considered which consists of the model's size and the connections within the model. This complexity is influenced by the problem but also by the modeling approach used. If, for example, modelers try to build simulation models of production lines with the aid of basic Petri net concepts, they will fail soon because of the model complexity. But even Clock Pulse Models (CPM) and Event Triggered Models (ETM), high-level Petri nets which have been proven to solve real-world modeling and simulation problems in production and logistics, must handle the burden of complexity. This paper discusses Stratified Simulation Models (SSM), a derivate of CPM, which is intended to solve this problem partially.

Figures 1 and 2 illustrate time complexity with respect to simulated time horizon and model complexity with respect to number of simulated components that CPM, ETM, and the newly introduced SSM tend to. Since CPM and, as a derivate, SSM are used to observe systems' behavior over time, they scale barely with the considered time horizon. And since SSM may stratify a simulated moment over several subsequent Petri net states, SSM may even perform inferior to CPM. In ETM, on the other side, only state changes and not the progress in time is simulated which enables a time-lapse simulation.

In CPM and ETM, real world components are modeled and simulated with subnets, and their composition models and simulates the entire system. In SSM, these subnets are folded into one if they are equally structured. The behavior of the formerly different subnets is then expressed with the aid of token attributes: an SSM can be interpreted as a simulation model of its CPM.

The folding technique that leads to SSM has two major advantages: First, SSM is superior regarding model complexity compared with CPM and ETM as shown in figure 2. Second, the overall system state is no longer distributed over a larger number of places and their marking but is concentrated in small number of places. This makes it much easier to observe the system and visualize the system state in a dashboard.

Communications of the ECMS, Volume 36, Issue 1,
Proceedings, ©ECMS Ibrahim A. Hameed, Agus Hasan,
Saleh Abdel-Afou Alaliyat (Editors) 2022
ISBN: 978-3-937436-77-7/978-3-937436-76-0(CD) ISSN 2522-2414

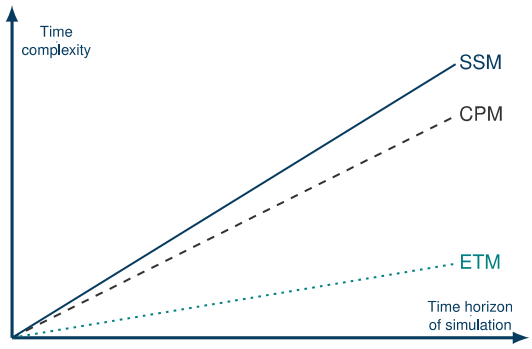

Figure 1: Time Complexity Trends for CPM, ETM, and SSM

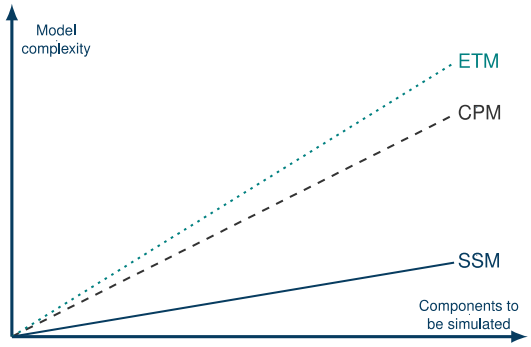

Figure 2: Model Complexity Trends for CPM, ETM, and SSM

The presented research is experimental, as the authors do not have an algorithm that takes a CPM and generates an SSM automatically. Experiments need a laboratory and ingredients. The authors' laboratory is the Process-Simulation.Center (P-S.C), a modeling and simulation environment for high-level Petri nets which is free to use for academic purposes. The ingredient is a simple production process that may happen all over the world.

The remainder of the paper is structured as follows: after this introduction, related work on Petri net folding techniques is presented which leads to the SSM approach. Next, the example process is explained. Using a pattern from literature, a CPM is developed first. In a non-algorithmic but structured way, an SSM is derived from this step-by-step afterwards. Moreover, a dashboard is designed to present the simulation results. For a discussion of the results and an outlook to the future work, the role of ETM is explained in more detail.

RELATED WORK

Reisig (2013) provides a fine introduction to Petri nets which are well suited to model, simulate, and analyze dynamic systems. Even large systems can be examined by expanding and connecting local components; different Petri net concepts allow for apposite modeling (Recalde et al. 2004). Their mathematical power is a bless and a challenge at the same time since the same problem can be solved with the aid of Petri nets in very different ways. It is difficult to teach the required creativity and for some people it is hard to follow this approach, especially if they do not have access to a suitable Petri net tool.

CPM, ETM, and the SSM approach introduced with this paper deliver modeling and simulation patterns for processes in production and logistic. They make use of high-level nets in the form of Predicate/Transition nets (Genrich and Lautenbach 1981). Tokens carry individual information, and a firing rule decides on which tokens to use for the next transition.

Since time concepts are highly relevant in practice, many different extensions have been proposed to include time information to otherwise timeless basic Petri nets. In the modeling approaches discussed here, time data is included beside any other system relevant information in the high-level net structure.

However, Petri nets per se exhibit an issue most graphical presentation methods ail from: models of real systems become very large quickly, i.e., the model complexity of Petri nets raises fast. Beside the already mentioned high-level Petri net concepts, other methods have been proposed to address this problem and squeeze large Petri net models:

Modularization is a decomposition approach. For example, Righini (1993) modularizes larger nets using Petri subnets as an implementation of hierarchical nets (Fehling 1993). Christensen and Petrucci (2000) modularize even without explicit subnets.

Clustering and Folding morphisms are further approaches (Keller 2002). These two methods make it possible to establish smaller models, however clusters are no Petri net concepts which causes semantical problems.

A special form of clustering and folding results from avoiding **redundant transitions** and **implicit places** (Garcia-Valles and Colom 1999; Simon 2008).

As a shortcoming of these methods, the scaling problems remain partly unsolved or are only relocated. For example, subnets or modules externalize portions of one model into other models that need to be integrated. This possibly leads to increased system complexity as interfaces need to be established and a necessity for redundant handling of data may arise.

A practical solution would be a small Petri net model that retains all information from a larger one while minimizing the footprint. To this end, folding seems to be a fitting technique. This paper examines the folding of a CPM and the challenges this poses to modelers.

A CPM FOR THE EXAMPLE

To present the SSM approach, a small real-world production process is introduced. It originates from the region the authors' university is located: viniculture in Rhenish Hesse in Rhineland-Palatinate, Germany. The example is used to examine timed dynamic systems in logistics and production. The following model is derived from a model pattern introduced in Simon et al. (2021b).

The process illustrated in figure 3 is the handling of personalized wine gifts in a winery. *Unlabeled bottles* of wine are taken from the storage and delivered to the input inventory of a workbench where they receive a personalized label according to customer wishes. In parallel, *cartons* are dispatched in form of *sheets* to another workplace to be assembled there.

Both items are taken to the packing station where bottle and newly created supporting material like greeting cards are put into the carton. Afterwards, the *packed gift box* (for short: *box*) is taken to another workplace where the shipping label is attached, and the box is sealed. The *completed box* is deposited in the outgoing goods area.

Figure 3: Layout Illustration of the Sample Process

Despite its relative simplicity, the process enables the examination of many different problems encountered in larger scale production processes. The concept can be modified as needed: different production strategies can be simulated, bottlenecks or constraints may be added, or process steps can be inserted, omitted, or temporally altered. Thus, it serves both as an easy-to-understand example and as testing grounds for bigger systems. The main point of this sample process is its transferability and scalability to general work and assembly workplaces.

In the following, customer orders of 75 gift boxes are examined. Thus, there are 75 unlabeled bottles and 75 unfolded cartons in the material storage. Implemented as a push process, items are put into the assembly-line as fast as possible. The process times are estimated: building a carton takes roughly 9 seconds while labeling a bottle requires 18 seconds. This time difference means that the downstream inventory of the carton production fills up faster than the other one. The most time-consuming activity with 25 seconds is the packaging step. Thus, both already mentioned inventories build up over time. The completing step accounts for 7 seconds.

Figure 4 shows the CPM. In short, the upper part depicts the bottles' labeling on the left side, while the right side presents the folding of the associated cartons. The subnet starting with the transition *startP* is associated with the merging of the two lines where bottle and additional items get packed into the carton, creating the gift box. Afterwards, the package is finalized and cleared.

Four types of places must be distinguished:

- Places modeling inventories like *inM* for the raw material or *inU* for unlabeled bottles.
- Places modeling activities to express that a specific task is currently conducted like *labeling* or *packing*.
- *feeding*∗-places that provide the first inner inventories with their raw materials.
- *idle*∗-places serving as semaphores to prevent more than one item to be processed at once.

The *idle*∗-places may additionally track the workplaces' idle times to measure utilization rates. *start*∗- and *stop*∗-transitions frame the activity-places.

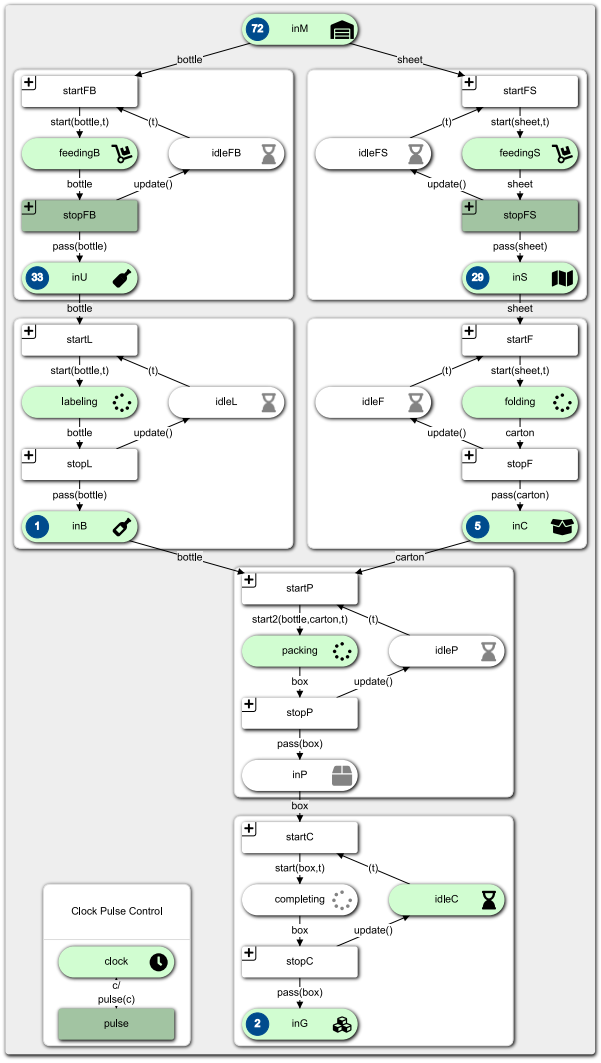

Figure 4: CPM of the Personalized Wine Gift's Production after 77 Clock Pulses

A further place type is *clock*, marked with the current simulation time. Firing the transition *pulse* increases this value by one time unit, here, a second. All other enabled transitions fire in unison, establishing a new system state. The *P-S.C* allows for exporting all system states. Using this data, the grounding system can be analyzed deeply.

This model can be simulated using different input data on place *inM*. The net's current state is represented as follows: blue dots show the number of tokens on high-level places that can be marked with records comparable to a table in a database. This is omitted for the *idle*∗-places and the *clock* which carry at most one token.

Simon et al. (2021b) have demonstrated how to integrate a simple dashboard into the given net which counts the number of items of each inventory in every moment in time. The SSM developed next is not only more compact than this CPM but also establishes new possibilities to design a simulation dashboard.

TRANSFORMATION FROM CPM TO SSM

Frames in figure 4 show repeating structures. *start*∗- and *stop*∗-transitions frame their activity places. The *start*∗-transitions take "material" from incoming inventory and, in addition, are controlled by an *idle*∗-place. The processed items are put on an outgoing inventory that also serves as incoming inventory for the following structure. A deviation can be observed for *startP* which has two incoming arcs and, correspondingly, two incoming inventories.

Since the corresponding places are type equivalent and the arcs and transitions are compatible, differing only concerning the included time information, they can be folded to the structure shown in figure 5: transition *start* takes an object from inventory *in* and puts it on *perform* indicating that an activity takes place. At the same time, the availability marker is removed from *idle* which avoids that a second object is "performed". When the performance ends, transition *stop* takes the object from *perform* and puts it on inventory *in* again. Marking place *idle* unblocks the *start-stop* subnet.

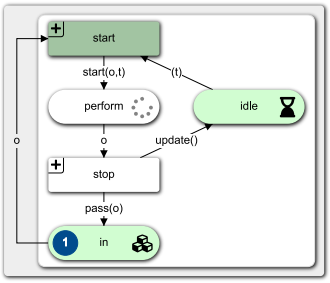

Figure 5: Folded subnet for repetitions in the CPM of figure 4

In the folded net information gets lost during the folding process: the particular place an item is located on. This information must be integrated into the item itself. To understand this important step, the high-level structure of the CPM in figure 4 must be considered.

Table 1 (left) shows a part of the initial marking of the place *inM* for the CPM. All other inventory places of the CPM are of the same structure. In a first step, this can be changed to the right structure where each token is augmented by the name of the place it lays on. Available simplifications are explained later.

Table 1: Initial Sample Allocation for Items in (left) the CPM and (right) the SSM

id	type	stamp
1	bottle	0:00
⋮	⋮	⋮
76	carton	0:00
⋮	⋮	⋮

id	type	stamp	place
1	bottle	0:00	inM
⋮	⋮	⋮	⋮
76	carton	0:00	inM
⋮	⋮	⋮	⋮

The time-typed *idle*∗-places of the original CPM must be enriched, too, as depicted in table 2: an attribute *place* refers to the former place and attribute *type* to the kind of item that lies on the place. The Boolean attribute *idle* symbolizes the availability of the workplace.

Table 2: Initial Sample Allocation for Workplace Availability in (left) the CPM and (right) the SSM

stamp
0:00
⋮
0:00
⋮

type	place	idle	stamp
bottle	inM	true	0:00
⋮	⋮	⋮	⋮
carton	inM	true	0:00
⋮	⋮	⋮	⋮

Transitions select tokens they may use for firing. The existence of selection rules is indicated by the plus symbol in the upper left corner in the depicted model. Outgoing arcs from transitions allow for computations to be made. Functions on these arcs are parametrized with variables taken from the incoming arcs.

The newly introduced additional information and the time information must be updated while they circle in the folded net. For the time information, the *clock/pulse* subnet remains in the SSM. The *pulse* transition fires whenever no other transition is enabled. This means that this stratum – the concept of all activities to be performed in a given time period – has been simulated, completely.

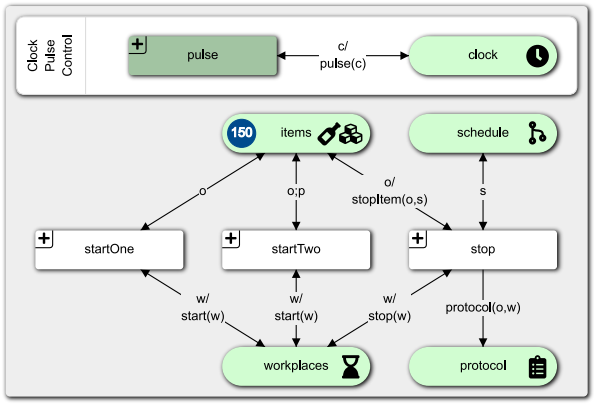

Figure 6: SSM of the Personalized Wine Gift's Production

The SSM in figure 6 takes the considerations made above into account and is a derivate of the CPM in figure 4. The savings that can be achieved concerning model complexity are tremendous as the number of places for the actual process is reduced from 19 to 4, and the number of transitions is reduced from 12 to 3. Actually, the savings concerning the places is even better since place *protocol* has only been introduced to ease the observation of the simulation.

Two aspects of the model in figure 6 have not been explained in advance: the necessity of two start transitions and the role of the place *schedule*.

Since in Petri nets the number of tokens that may be transferred over an arc simultaneously is fixed, two transitions are needed: One to model taking a single item, and a second to model taking two items at once. The selection criteria on these places guarantee that *startOne* cannot "steal" tokens that *startTwo* needs for firing.

During the process of folding the CPM manually, a characteristic of the P-S.C that seemed to be limiting at first glance led to the introduction of the place *schedule.*

This place supplies information about the processing times of the workstations and the state changes that occur for these items. While performing a specific work step on an item is time consuming, accessing an item out of an inventory is regarded as timeless. Figure 7 shows the chain of states the items traverse while they are processed by the Petri net, and which is implemented as the marking of place *schedule*.

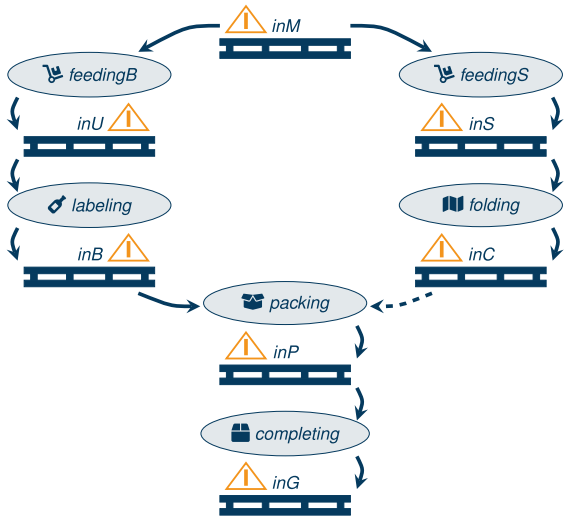

Figure 7: States of the items in the CPM and possible state changes

These considerations led to the final simplification: it is not necessary to present the current time for items on the place *items*. It is sufficient to code this information on *workplaces* as its data combined with that from *items* establish new record sets for *protocol* at those moments an inventory receives an item, or an activity starts.

As has been pointed out before, all folding steps have been conducted in a structured way but without an algorithm. The authors, however, see a chance and the potential of further automation of these steps.

A DASHBOARD FOR SIMULATION RESULTS

The CPM of the exemplary process allows to observe the raising and sinking of the inventory stocks while the Petri net simulation runs. Since these stock quantities and the utilization of the working places are folded into few places, this is not possible for SSM anymore.

As an alternative, the place *protocol* was introduced which summarizes the simulation and is results over time. For each item, all reached states are stored in attributes *id* (to identify the item), *type* for the kind of item and *place* with the information where the item is currently located. An attribute *busy* indicates the moment the state was reached and *idle* the moment a following item is allowed to reach the same state.

Every information about the model, its dynamics and its results can be extracted from this protocol easily. Table 3 shows an excerpt of the protocol relevant to produce the first gift box.

Table 3: Protocol Excerpt for the First Giftbox

id	type	place	busy	idle
1	bottle	feedingB	0:00	0:02
76	carton	feedingS	0:00	0:02
1	bottle	inU	0:02	0:02
76	carton	inS	0:02	0:02
76	carton	folding	0:02	0:11
76	carton	inC	0:11	0:11
1	bottle	labeling	0:02	0:20
1	bottle	inB	0:20	0:20
1	bottle	packing	0:20	0:45
76	carton	packing	0:20	0:45
1	box	inP	0:45	0:45
1	box	completing	0:45	0:52
1	box	inG	0:52	0:52

The advantage of SSM concerning model complexity is paid for with a higher time complexity compared to CPM. This can also be seen in the marking of place *protocol*. It is possible that no action at all takes place in one second, i.e., one stratum. On the other side, table 4 shows all state changes that can be observed for second 20. And since each of these changes are written by transition *stop*, the strata span in the simulation for this moment in real time becomes obvious. Moreover, also *start*-transitions may fire for one second.

Table 4: Protocol Excerpt for second 20

id	type	place	busy	idle
1	bottle	labeling	0:02	0:20
10	bottle	feedingB	0:18	0:20
1	bottle	inB	0:20	0:20
77	carton	building	0:11	0:20
10	bottle	inU	0:20	0:20
85	carton	feedingS	0:18	0:20
77	carton	inC	0:20	0:20
85	carton	inS	0:20	0:20
11	bottle	feedingB	0:20	0:22
86	carton	feedingS	0:20	0:22
78	carton	building	0:20	0:29
2	bottle	labeling	0:20	0:38
1	bottle	packing	0:20	0:45
76	carton	packing	0:20	0:45

After a stratum has been simulated completely, the clock increments by one and the next stratum can be simulated. This special form of simulating a timed system led to the name for this modeling technique: *Stratified Simulation Model*.

The final step for the discussed approach is how to represent the simulation results in a dashboard, a major problem identified by Simon et al. (2021a).

Dashboards serve as a communicator to the user, the business community, or to attract the attention of other researchers. On the other hand, they serve as a checkpoint to ensure that the results in the newly developed concepts are still correct.

For the generation of a dashboard out of the simulation results of a Petri net, the data must be prepared first. In business intelligence, this is conducted in the ETL-phases Extract, Transform, and Load (van der Lans 2012).

For CPM, the information concerning reachable system states must be collected from the different places all over the net. This is different for the presented SSM and its place *protocol* which focusses the observation of the entire system on a single place. This makes the ETL process much easier. This become obvious when the amount of information generated during the simulation is considered: for the CPM model, a CSV file is generated with a size of 3.1 Mbyte, while the SSM model only generates a CSV file of 38 Kbyte, i.e., only 1% of the original size. However, this file still contains all relevant information concerning stock utilization and idle times of working places.

Figure 8 demonstrates how to prepare the data for further use. Assume that the Winery in Worms wants to analyze the productivity of the individual areas for the purpose of overhead cost accounting. Based on the time stamps of place *protocol* it is possible to examine exactly which box was where and when, and when it was processed. Interactive filters in dropdown menus exemplarily help to find the shortest and longest waiting time per inventory and many other information.

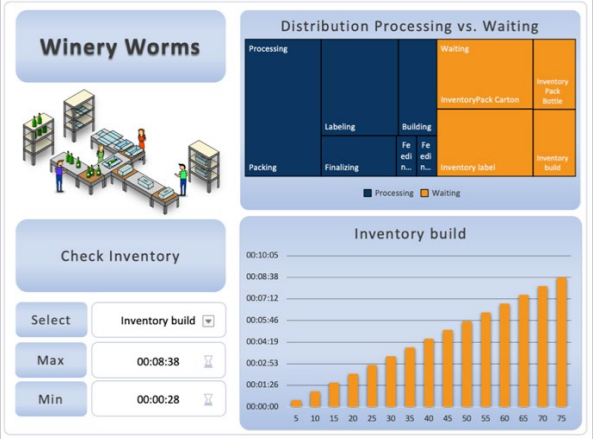

Figure 8: SSM Dashboard

The upper graph in the dashboard of figure 8 shows the distribution of waiting and processing times at the Winery in Worms. If a dashboard user wants to follow the origin of the large, orange tile of the inventory build label, this can happen elegantly over the selection menu beside *Select*. The graphics and times adapt accordingly.

CONCLUSION AND OUTLOOK

The current paper demonstrates an approach how to solve a problem commonly known from modeling and simulation, especially if Petri nets or other graphical concepts are used: The models tend to have a high model complexity which makes them hard to prepare, difficult to read, and error-prone.

Nonetheless, Petri nets and especially CPM can be applied successfully to problems in production and logistics as has been shown by Simon et al. (2021b), but there still is the necessity to overcome the model complexity problem.

The SSM approach that has been introduced here would be one possible solution if SSM could be derived from CPM automatically. The development of CPM patterns to model technical components of such systems and to provide SSM transformations for these patterns could possibly solve this task. One problem, however, would persist: the time complexity.

A first answer to this problem is ETM, as has also been demonstrated by Simon et al. (2021b). ETM are high-level Petri nets, too. The marking of a place represents the system state of a specific component at a specific moment in time, but the distinct moments considered for the different places of the net may vary from place to place. Necessarily, time information must be included in each token which indicates the moment a specific state was reached. From this stage, the next possible (local) state change can be calculated including the moment this change occurs. Especially if state changes occur rarely in the time scale chosen for the simulation, ETM tend to a lower time complexity when compared to CPM and SSM. If observing the system during a simulation run is not important, this is a practical solution.

Future work on new Petri net modeling techniques for timed systems will be the development of folding techniques for ETM. These Folded Event Systems (FES) are currently in their conceptual phase and are anticipated to have time and model complexities compared to CPM, ETM, and SSM as depicted in figures 9 and 10.

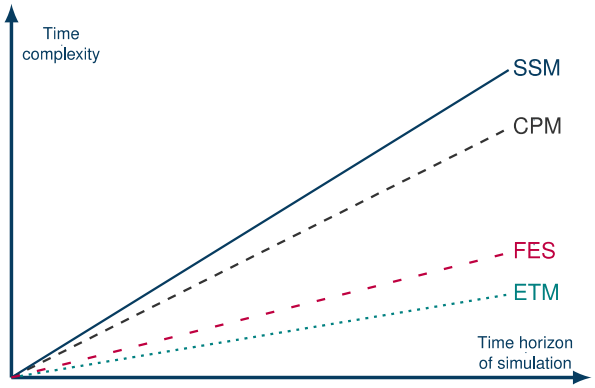

Figure 9: Anticipated Time Complexity Trend for FES

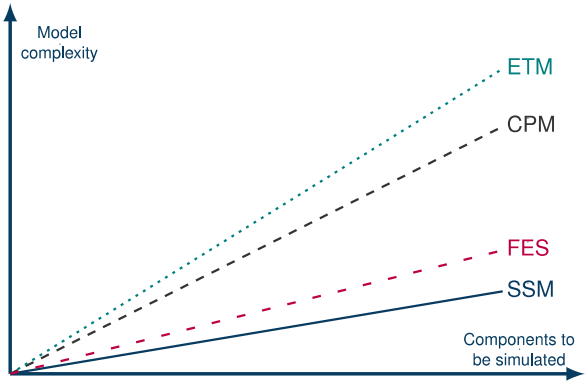

Figure 10: Anticipated Model Complexity Trend for FES

The previous consideration on the development of dashboards as based on the place *protocol* gives a hint on how the future research is conducted. With the aid of this place, it is possible to identify all moments in time when exactly one single clock pulse occurs in the SSM. If the clock pulses can be substituted by event pulses, the simulation speed can be increased dramatically.

REFERENCES

Arona, S. and B. Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press, UK.

Batty, M. and P. M. Torrens. 2001. "Modeling complexity: The limits to Prediction", *UCL Working Papers Series* 36.

Christensen, S. and L. Petrucci. 2000. "Modular Analysis of Petri nets". *The Computer Journal* 43, 3, 224-242.

Fehling, R. 1992. *Hierarchische Petrinetze* (German, transl. Hierarchical Petri nets). Kovac, Hamburg, Germany.

Garcia-Valles, F. and J. M. Colom, 1991. "Implicit places in net systems". *Proceedings of the 8th International Workshop on Petri Nets and Performance Models*, 104-113.

Genrich, H. J. and K. Lautenbach. 1981. "System Modeling with High-Level Petri Nets". *Theoretical Computer Science* 13, 1, 109-135.

Keller, W. 2002. "Clustering for Petri nets". *Theoretical Computer Science* 308, 145-197.

van der Lans, R. 2012. *Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses*. Morgan Kaufmann Series on Business Intelligence, Elsevier, Amsterdam, the Netherlands.

Popovics, G. and L. Monostori. 2016. "An approach to determine simulation model complexity". *Procedia CIRP* 52, 257-261.

Recalde, L.; M. Silva; J. Ezpeleta; and E. Teruel. 2004. "Petri nets and manufacturing systems: An examples-driven tour". *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, 742-788. Springer, Berlin, Germany.

Reisig, W. 2013. *Understanding Petri Nets*. Springer, Wiesbaden, Germany.

Righinni, G. 1993. "Modular Petri nets for simulation of flexible production systems". *International Journal of Production Research* 31, 10, 2463-2477.

Simon, C. 2008. *Negotiation Processes - The Semantic Process Language and Applications*. Shaker, Düren, Germany.

Simon, C.; S. Haag; and L. Zakfeld. 2021a. "Research Agenda for Process Simulation Dashboards". In: *ECMS 2021: 35th International ECMS Conference on Modeling and Simulation*, 243-249.

Simon, C.; S. Haag; and L. Zakfeld. 2021b. "Simulation of Push- and Pull-Processes in Logistics: Usage, Limitations, and Result Presentation of Clock Pulse and Event Triggered Models". *International Journal on Advances in Software* 14, 1&2, 88-106.

AUTHOR BIOGRAPHIES

CARLO SIMON studied Informatics and Information Systems at the University of Koblenz-Landau. For his PhD, he applied process thinking to automation technology in the chemical industry. For his state doctorate, he considered electronic negotiations from a process perspective. Since 2007, he is a Professor for Information Systems, first at the Provadis School of Technology and Management Frankfurt and since 2015 at the Hochschule Worms. His e-mail address is: `simon@hs-worms.de`.

STEFAN HAAG holds degrees in Business Administration and Engineering as well as Economics with his main interests being related to modeling and simulation in graphs. After working at the Fraunhofer Institute for Systems and Innovation Research ISI Karlsruhe for several years, he is now a Research Fellow at the Hochschule Worms. His e-mail address is: `haag@hs-worms.de`.

LARA ZAKFELD graduated in International Logistics Management (B.A.) after completing an apprenticeship as a management assistant in freight forwarding and logistics services. She is currently pursuing a master's degree in Entrepreneurship and works as a Research Assistant at the Hochschule Worms. Her e-mail address is: `zakfeld@hs-worms.de`.