Clock Pulse Modeling and Simulation of Push and Pull Processes in Logistics

Carlo Simon, Stefan Haag and Lara Zakfeld

Hochschule Worms
Erenburgerstr. 19, 67549 Worms, Germany
Email: {simon, haag, lara.zakfeld}@hs-worms.de

***Abstract*—This paper is about a new technique to find Petri net models for a clock pulse spotted simulation of processes in logistics and production. These models can be used to observe the raising and discharging of stocks in production in order to identify bottlenecks, to observe differences of push and pull strategies on the valued stocks, and to decide on strategic changes. For this, however, significant preliminary tasks had to be conducted first which are also the objective of this paper: a novel, Web-based Petri net modeling and simulation environment called Process-Simulation.Center (P-S.C) has been developed since existing tools are not at the least able to handle such sophisticated models. At the moment the tool worked properly, different approaches to model the described situation had to be compared. A teaching laboratory for logistics has been chosen as a sample application. The simulation now helps students to scale up their personal observations in the laboratory with respect to time, amount and value. This paper explains the situation in the laboratory, the new features of the P-S.C that enable the modeling of these processes, and the finding of the model itself. Finally, its development led to another, different approach to describe the teaching processes by a so called event triggered simulation that is shortly considered in contrast.**

Keywords–Conceptual models of timed dynamic systems; Simulation; Petri nets; Logistics; Teaching.

I. INTRODUCTION

Enterprise Resource Planning systems (ERP) can collect and integrate data using a common database, thereby representing a good basis for the overall accounting process [1]. This information can be used in all areas of a company such as sales, purchasing and finance, but also human resources or production. In many companies, they are used to plan and control manufacturing processes. These systems evaluate and name the next upcoming orders, record the progress of work by means of confirmation messages and determine production plans.

ERP systems, however, do not support their users in making decisions concerning a modified production strategy. For example, turning push processes in logistics and production into pull processes often comes along with a significant reduction of stock costs and, at the same time, an increased flexibility. Nonetheless, we observe that still many companies produce in accordance with the push principle because of over-reliance on the production strategy suggested by the ERP system, but also because of nescience of the different strategies and their effects. In order to avoid the latter, it is very important for students in the field of logistics to experience different strategies in their lectures.

However, practical experience is typically restricted concerning available time as well as amount of handled goods, machines and space. The so called *box game*, explained in Section III, for example, imparts this knowledge in a relatively simple setting. In order to extend this experience, we had the idea to develop a simulation model to overcome the mentioned limitations.

Since we work on Petri nets for decades, we decided to use Petri models to demonstrate the consequences of various processes in logistics, especially concerning the application of push and pull strategies. This, however, was a more challenging task than assumed at the beginning:

1) Especially models and simulations of pull processes rely on the possibility to distinguish between different customer orders. Hence, such processes cannot be modeled with simple Petri nets, but need Petri nets with individual tokens, so-called high-level Petri nets. Although such Petri net classes are known for many years (e.g., [2] [3]), no blueprint could be found how to develop appropriate models with them. Actually, most publications on this topic are of theoretical nature and not intended to make use of simulation results. We even assume that many of the shown models have never been tested in practice using tools. This, however, delivers new insights and new modeling techniques. Thus, one of the outcomes of this paper is a description of how the final clock pulse simulation model has been developed.
2) The above mentioned experience concerning modeling and simulation of Petri nets relies on the existence of an appropriate tool. At the present, this is a real problem. Almost all Petri net tools listed in [4] are either obsolete, do not support time aspects or Petri nets with individual tools, and none of them have modern user interfaces. They are useless for the finding of new modeling techniques for high-level Petri nets and new applications of simulation models in logistics and production. In order to overcome this limitation, we developed the Process-Simulation.Center (P-S.C), a novel, Web-based Petri net modeling and simulation environment that, among others, can be applied to the above mentioned logistics laboratory. Actually, this tool has been developed over years and its application to the box game is the last one in a series of use cases in the recent years.

In order to explain the research progress that could be achieved in the fields of the development of a tool for the conceptual modeling and simulation of processes with the aid of Petri nets, the finding of an appropriate model for the logistics problem, derived recommendations on how to find such models, and finally concerning the concrete laboratory process, this paper is organized as follows: The considered laboratory process is briefly described in Section III, while the related work is discussed in Section II. These two sections have a lot in common with the corresponding sections in [5] where a different modeling approach is considered for the same process. Section IV describes the development of the clock pulse model, explains its components, and discusses how the simulation results can be interpreted. Then, Section V shortly summarizes the specifics of the modeling approach of [5] and explains the fundamental differences between both modeling techniques. In Section VI, a conclusion both on the simulation results and the modeling approach is presented. The paper closes with an overview of planned and possible future work.

Contrary to other publications of ours, an explanation of the research agenda is omitted in this paper since this is outlined in other contemporaneous publications [5].

II. Related Work

Originally, Petri nets are defined as Place/Transition (P/T) nets with anonymous tokens indicating a system's state [6]. Diverse concepts for representing high level information in Petri nets exist, with the most widely known being:

Predicate/Transition nets omit anonymous tokens for ones carrying data that can be processed and altered by use of functions encoded on transitions. When firing, these functions accept data from tokens on the preset. Functions return their results by putting appropriate tokens on the postset. The places serve as predicates according to which transitions may fire. Thus, it is possible to model interactions of tokens according to real-world influences or with each other. [2]

Colored Petri Nets integrate colors into Petri nets such that tokens, places and transitions have an assigned identity, their color. When determining if a transition is enabled, the adjacent places and their tokens are examined by color separately. This allows for more compact net representations under certain circumstances. [3] [7]

Regarding modeling, more specifically that of technical systems or business processes, time is important. Without claiming completeness, several approaches combining Petri nets and time concepts have been presented: Time Petri Nets (TPN) [8] and Timed Petri Nets (TdPN) [9] associate time with transitions, Time Place & Transition Nets (TPTN) [10] assign time with places and their marking, and Arc Timed Petri Nets (ATPN) [11] [12] associate time with the arcs of Petri nets.

The aforementioned formalisms share one concept differentiating them inherently from original Petri nets: Their state relies not only on information as given by their respective markings, but also on some kind of timer clocks. Timestamps are means to encode time information in the marking only.

Timestamp Nets introduce tokens with timestamps designating the moment the corresponding token was placed. Transitions may fire in time windows as given by two non-negative values on the transitions' incoming arcs [13]. The permeability of arcs depends on these timestamps [14].

Extended Timestamp Nets integrate the concepts of Pr/T and Timestamp nets such that tokens carry timestamps and any further information [15].

Some of the approaches may be transformed into each other quite effortlessly [16] [17]. All of the presented Petri net formalisms, however, use artificial, abstract time units. To model and simulate real-world applications, real time values should be used. To this end, date and time data types seem beneficial to be included as possible information on tokens. For reference, there are other modeling methods that were developed to combine time and process structures.

Value Stream Diagrams (VSD) establish models of flows of information and material in order to evaluate value streams. To optimize the value streams, wait times - beside other factors - need to be minimized. The value stream method exposes such wait times. This concept became widely known due to Toyotas Production System from 1930 and its advancements by Japanese engineers Taiichi Ohno and Eiji Toyoda, but dates back as far as 1914 when graphical nets were used to examine routings and other flows to help *Installing Efficiency Methods* in a manufacturing company [18] [19].
Figure 2 depicts the exemplary process as a VSD. It gives a suitable high-level overview of the whole value stream from customer to supplier, however cannot be simulated.

Business Process Model and Notation (BPMN) is a notation and representation language for modeling business processes. It is extensively used due to the relative ease of both creating and understanding models. Using BPMN, it is possible to create both high-level models of companies and low-level models of single processes in a graphical approach similar to flowcharts [20] [21].
Although there are similarities between BPMN and Petri nets, the former lack the mathematical toolset that can be used to analyze Petri nets in form of linear algebra.

If conceptual models are developed for process simulation or execution, also tools are needed in addition to the formal mathematical base. The Process-Simulation.Center (P-S.C) is a Web-based modeling and simulation environment supporting the development of P/T and Pr/T nets [22].

In P-S.C, it is possible to assign data types to places and to use these places in analogy to tables in databases. Special types for time and date are important substructures for the simulation of processes in production and logistics and enhance the mentioned approaches to timed Petri nets.

In contrast to Relational Algebra and SQL where operations like select or projection are applied to the set of all affected tuples and result in a set again, in P-S.C the tuples are processed serially. This is since, in business and production, work items are also treated one after another. A decision on the concrete sequence is made locally by the transitions of the net which also has the ability to aggregate over the tuple tokens on a place, like this is known for database systems.

Also, the P-S.C can be used to combine the process view on a system with other views. Process maps can be used to collate different processes with each other and to express the strategic value of processes as primary, supportive or managing.

The organizational structure of an institution can be combined with the Petri net view on the processes by assigning its nodes to swim lanes for the responsible organizational units. Organizational charts complete the functions of the P-S.C.

Contrary to most other conceptual modeling tools, especially those that have been developed for Petri nets, for the P-S.C a specification language has been developed with which all types of models are scripted. Due to strong algorithms for automatic layout, modelers can concentrate purely on structural aspects of the domain to be expressed.

III. A Simulation Laboratory for Processes in Logistics

The so-called *box game* has been developed at the Worms University of Applied Sciences to teach students in logistics and is used as a sample application. Despite its simplicity, very different kinds of processes that also have a high impact for practice can be observed. It is, therefore, ideal for trying out different ways of conceptual modeling and simulation.

The concrete example is a simple construction process where students assemble small and large boxes, put the smaller into the larger ones, and check the quality at the end of the process. With this process, characteristics of push and pull systems are illustrated.

Training members are present during the game. Ideally, they take part in the game in order to gain first hand experience of motivation and work situations. Being engaged in the training helps the students to recognize different types of waste, such as *overproduction*, *waiting*, and *motion*, but also the transformation of waste types. Finally, discussing the shared experiences is a major part of the learning success. A complete simulation run of the *box game* lasts approximately two to three hours.

Despite the simplicity of the used material and the low level of technical requirements, the *box game* is easily transferable to assembly work stations in a more generalized form and has a highly practical impact. Mechanical production, however, where schedules, shift patterns, changeover times or multiple machine set-ups are of particular importance, is not an objective of this training.

Figure 1 shows the spatial organization of the *box game* in the learning laboratory: five work tables are arranged in a suitable location and standard positions like interim storages are marked with adhesive tape. As can be seen, the setting can also be build up in locations such as conference rooms, training rooms, or even canteens.

The following activities have to be conducted at the five stations or (transport-wise) between them:

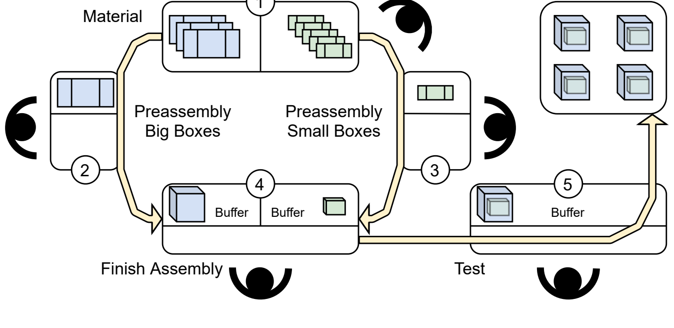

Figure 1. Layout design of the *box game*

1. **Storage (S)** Deliver boxes.
2. **Preassemble big boxes (PBB)** Fold box, close lid, and pass the box on.
3. **Preassemble small boxes (PSB)** Fold box, close lid, and pass the box on.
4. **Finish assembly (FA)** Open big box, insert small one, label small box with a post-it as ”package note”, close and tape big box lid up, and pass on.
5. **Test (QA)** Shake box for an acoustic quality check, apply a red dot to the upper left corner of outer box, and place the finished box in the storage area.

Beside the trainer, the following can partake in the game:

5 participants who will occupy the work stations
3 players who record the processing times
1 observer who records the inventory in the system
1 observer who records productivity levels
Possibly 2 employees who will disassemble the boxes

The initial stock of the *box game* is 75 big and 75 small boxes. However, it is not the aim to produce the entire demand in the shortest possible time, but to produce them according to the customer demand - one part every 15 seconds - without inventory and with as few employees as possible.

Figure 2 shows the value stream diagram of the *box game*. Although the processing time can be annotated in the diagram, it is hardly simulated due to a lacking mathematical foundation. Even software that replicates human intuition of value stream diagrams is missing. Nonetheless, the diagram helps to understand how the *box game* is played in detail.

Typically, the simulation is played in four rounds where each round lasts for a defined duration (e.g., 5 or 8 minutes). During the simulation, two types of principles with two batch sizes are played.

Batch size 3 - push principle: The products are passed on in batches of size 3. Each process step works functionally independent from the other and the participants are paid by the number of pieces they work on. Hence, it is the goal at each station to produce as much output as possible.

Batch size 3 - pull principle: Stations produce and pass on products in batches of size 3. Upstream stations have to hold their pieces and stop production until it is demanded by an internal or external customer. The capacity of a station and its buffer is limited to 3 items and items can only be replaced accordingly.

Batch size 1 - pull principle: The third round is played like the second one, but the batch size is reduced to one.

Improvement - pull principle: The last round is used to find improvements autonomously and to apply them as a team.

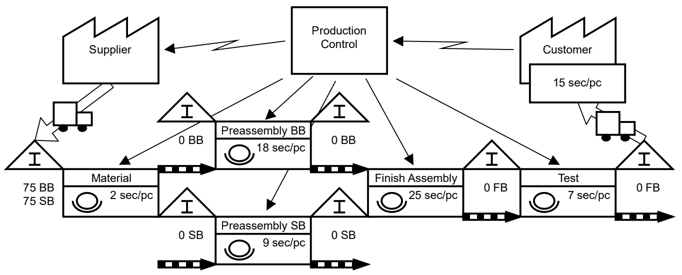

Figure 2. Value stream diagram of the *box game*

The advantage of this approach is that the participants gather personal experiences. This can hardly be replaced by a computer simulation. However, augmenting this hands-on experience by a computer simulation helps to scale up both complexity and range of the considered process.

IV. Clock Pulse Simulation Models

The aim of observing storage utilization over time necessitates a constant flow of time. To this end, the first step in modeling the *box game* is the implementation of a clock that ticks every second, as depicted in Figure 3 (upper left). A tiny Petri net consists of a time-typed place *clock*, a corresponding transition *pulse* and two arcs, one of which removes a time token from the *clock*, while the other one adds one second to the received value and puts the token back on the *clock*. Thus, each firing increments the elapsed total time by one second. At this point, it is worth mentioning that the ProcessSimulation.Center draws places such that both their labels and token quantity can be presented within the node.

The basic model of a working station - here, the folding of the big boxes, as shown in Figure 3 (lower) - consists of a place *inBB* for the attached upstream storage and a place *buildBB* for the workplace itself. An adjacent transition *deliverBB* provides the storage with feedstock coming from the *material* warehouse (and possibly information about the order volume).

The transition *startBB* gets the item with the minimal id according to the fifo principle and puts it on the workplace *buildBB*. Then, the transition *stopBB* waits for the item to be finished as indicated by the processing time (which is encoded on the item's token) and the time of the aforementioned *clock* elapsed. For this, in our models, all enabled transitions fire together in steps. This waiting is implemented as a condition on *stopBB*. Conditions and selection criteria can be displayed inside the model by clicking on the plus-symbol.

The transition *stopBB* is attached to an interim storage serving as buffer for the succeeding working station, as shown later. The place *idleBB* serves two purposes: as a semaphore that prohibits *startBB* from firing while the workplace is busy, and as a counter of the idle time of the workplace.

An observer net, as presented in Figure 3 (upper right), evaluates the costs associated with the storage. The value of place *inBBEval* is increased by the number of tokens on place *inBB* indicating the cost of the storage in that exact moment. In combination, these models allow for the observation of fluctuating storage levels, of bottlenecks, and of storage cost.

By expanding on the presented working space model and implementing the mentioned concepts, the first iteration of the *box game* - the push version as presented during a simulation run in Figure 4 - can be modeled.

As the main principle of the building net - exemplarily shown for the big boxes - can be used for the other workplaces, they can be structurally copied and then adapted with respect to the processing time.

Connected to the upstream storage places of the big and small box folding nets, there is the mentioned place for the main warehouse. The downstream storage places are linked to the interim storage serving as buffer for the finishing step.

The working station for tests is connected directly to the outgoing storage of the final assembly. As is the case for conditions, functions and data accesses on arcs can be shown or hidden, as needed.

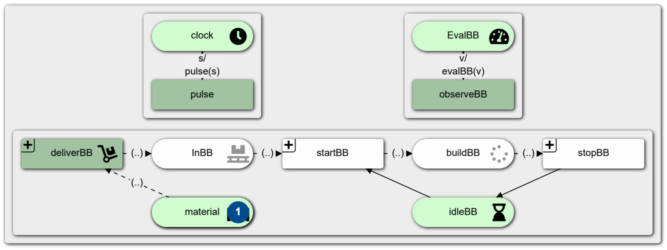

Figure 3. (upper left) Simple Petri net clock · (upper right) Observer net that evaluates the upstream storage in (lower) · (lower) Building big boxes as instance of a single working station

To further enhance the visual understanding of their respective functionality, nodes can be provided with symbols.

The pull version of the *box game* as depicted in Figure 5 is slightly more complicated, as supplementary elements are needed to implement pull requests. This is done by additional pull-transitions that keep track of the contents of upstream storage places for all three construction workplaces. As soon as these are empty, a pull request is issued, leading to a delivery.

Comparison of the two models' simulation runs show neither a change in total processing time nor one in idle times for the different workplaces, which is as expected. Differences on utilization of storage places become evident, though.

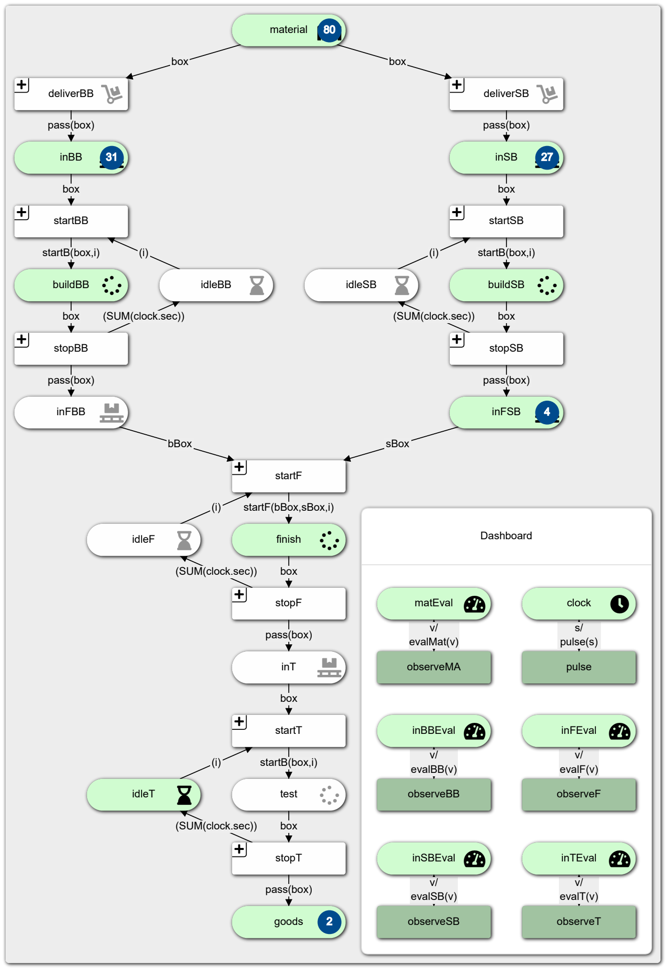

Figure 4. Pulse Clock Push Model

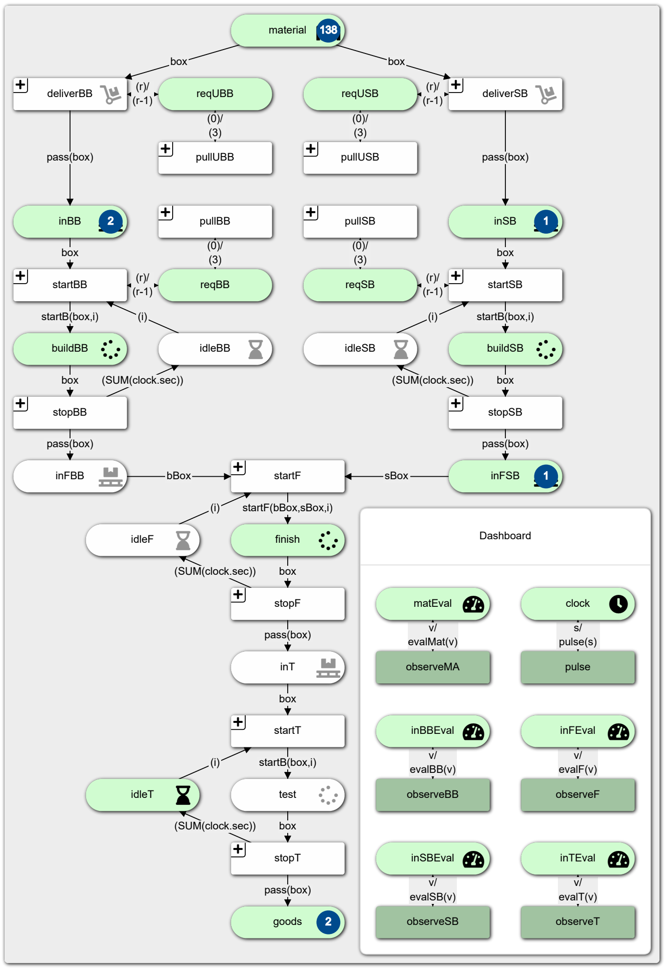

Figure 5. Pulse Clock Pull Model

The first model using push principles clearly shows the drawback of this approach: large interim storage and, as a result, high inventory costs. Figure 6 (upper) depicts the inventory on the material storage, the building buffer storages and the finished goods storage during the push simulation.

Figure 6 (lower) presents the pull simulation run's results with otherwise unaltered preconditions. The interim storage places are much less utilized as only those items are put into the assembly lines that are demanded by downstream stations.

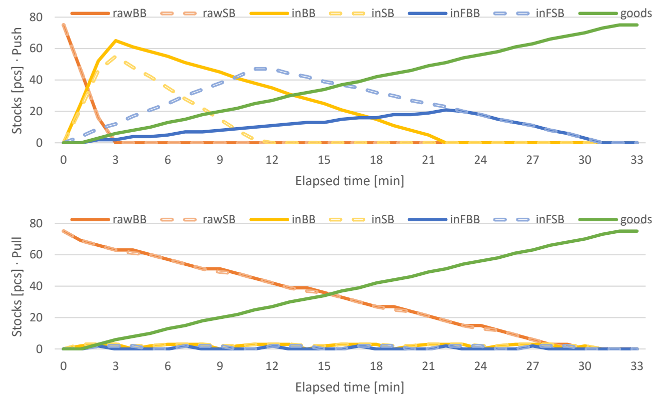

Figure 6. Stocks in the pulse clock model for Push (upper) and Pull (lower)

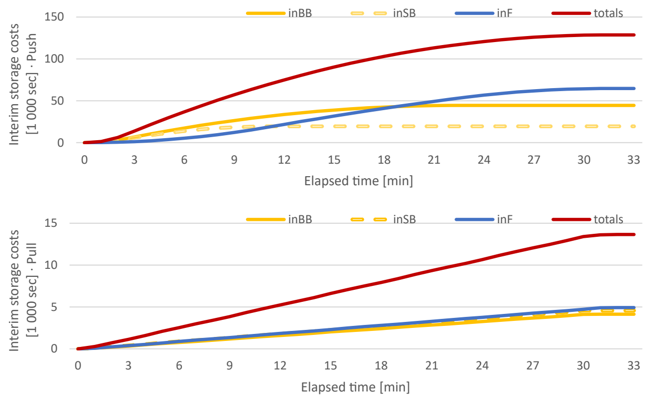

Figure 7. Inventory costs per interim storage and accumulated for Push (upper) and Pull (lower)

This is as expected, as it corresponds to the main goal behind just-in-time production schedules.

Figure 7 (upper) presents the costs of interim storage in the push model, while Figure 7 (lower) presents the same for the pull model. Also shown are the accumulated costs of all interim storage places. The finished goods storage is omitted as throughput is the same in both models, thus leading to the same costs on this storage. The material warehouse, otherwise, is put aside as the difference between the accumulated totals equals to the possible savings when externalizing the warehouse. By simply implementing pull principles, the evaluated stock cost on the interim storage places plummet. Thus, the advantages of a just-in-time production and a smaller main storage become obvious. The pull principle allows for exactly this.

What is not accounted for in these models are for example costs of transportation, as smaller batch sizes usually correspondent to higher transportation costs, leading to familiar knowledge: decreasing one muda typically increases other muda. Hence, batch size 1, which is optimal for the interim storage cost, is not necessarily the globally optimal solution.

V. An Alternative Modeling Approach

The advantage of the described way to model the box game is that the rising and discharging of the stocks can be observed throughout the simulation in a very illustrative way. The number of items in each stock is obvious for every second. Since in the P-S.C the color of and the symbol on places can change dependent on the number of tokens on the place, the advantages and disadvantages of push and pull can be demonstrated very well. As a consequence, for the students in the logistics laboratory the simulation is a demonstrative extension of their personal experience.

For the concrete example described here, 1902 steps that represent seconds (or almost 32 minutes) in the real world game are calculated for the full simulation, both for push and pull. This is a much longer period than the one that can be played by the students, because the concentration of the students decreases after about 5 minutes. But the calculation of all of these steps take time. On an iMac with 4 GHz Quad-Core Intel Core i7 processor and 16 GB RAM, the full simulation takes in a Chrome browser 8234 milliseconds. The duration for the simulation of a working day would increase linearly. This observation led to the question how to save simulation time. Actually, this can be reduced drastically if the visualization is less important compared to the concrete result.

In this case, it would be sufficient to calculate new states only in the moments of state changes. For the concrete example discussed in this paper, the simulation can be reduced from 1902 down to 79 steps for push and 228 steps for pull. On the same computer mentioned above, the simulation takes then 315 or 923 milliseconds, respectively. We call this second approach an event triggered simulation. Its benefits increase even more, if the demand interval in which changes occur vary strongly from one part of the model to the other since it takes into account local state changes.

The price that has to be paid for this is that a visualization of the simulation results must be generated in a separate working step. Also, we sensed the development of the event triggered model as more difficult compared to the clock pulse model. From this personal observation, we derive the following suggestion:

Use a clock pulse simulation if either a clock pulse visualization of the system's states is needed or if the computer is fast enough for the few simulations that must be run for the modeled system.

Use an event triggered simulation if simulation speed is necessary due the complexity of the modeled system, if fast answers are needed in production, or if a large number of variations of the production schedule or input data has to be compared. In general, the more often a specific model needs to be simulated, the more it is worth to develop an event triggered model instead of a clock pulse model.

VI. Conclusion and Future Work

The modeling exercises that had to be solved for this paper led to some new insights in the development of conceptual models for discrete timed systems. Finally, we found our personal good practice that consists of the following steps:

1. Define data types for the different stocks and other data objects, and initialize the corresponding places in accordance with the starting condition.
2. Augment the model by transitions for beginning and ending specific tasks like delivering raw materials, building or testing a box.
3. Identify the next item to be taken and the moment this will occur. This also allows for implementation of different prioritization strategies.
4. Start with modeling the simpler push principle and augment this model by pull principles.

Moreover, we now see the development of a clock pulse model as a preliminary for the development of an event triggered model. If an event triggered model is needed due to the above mentioned reasons, we suggest the following steps:

1. Develop the clock pulse model first.
2. Observe the reasons for state changes with the aid of the clock pulse model and derive the event triggered model from these observations.
3. Look for a proper visualization of the simulation results.

In order to ease the step from a clock pulse model to an event triggered model, in the future we will work on an extension of the P-S.C such that users are supported in finding the relevant moments of state changes. Probably it will be possible to automatically support the modelers in conducting this task.

What impressed us most is that what can be modeled and simulated with the aid of Petri nets is only restricted by the modelers imagination and the ability of the used tool. In opposite to other out of the box modeling environments for logistics, a user is free to lay the focus on any parameter they are interested in most.

However, two challenges exist: Modelers must be able to develop sophisticated, abstract models and must find a way to visualize the results. We hope that we have given one answer to the first challenge and see major future work concerning the second one.

References

[1] C. Caserio and S. Trucco, Enterprise Resource Planning and Business Intelligence Systems for Information Quality. Cham, Switzerland: Springer, 2018.

[2] H. J. Genrich and K. Lautenbach, “System Modelling with High-Level Petri Nets,” Theoretical Computer Science, vol. 13, 1981.

[3] K. Jensen, Coloured Petri-Nets, 1st ed. Berlin: Springer, 1992.

[4] “Petri Nets Tools Database Quick Overview,” https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html (last accessed 2020.09.20).

[5] S. Haag, L. Zakfeld, C. Simon, and C. Reuter, “Event Triggered Simulation of Push and Pull Processes,” in SIMUL 2020: The Twelfth International Conference on Advances in System Simulation. Porto, Portugal: IARIA XPS Press.

[6] W. Reisig, Understanding Petri Nets. Berlin: Springer, 2013.

[7] M. Montali and A. Rivkin, “From DB-nets to Coloured Petri Nets with Priorities (Extended Version),” CoRR, vol. abs/1904.00058, 2019. [Online]. Available: http://arxiv.org/abs/1904.00058

[8] P. Merlin, “The Time-Petri-Net and the Recoverability of Processes,” University California, Irvine, Tech. Rep., 1974.

[9] C. Ramchandani, “Analysis of Asynchronous Concurrent Systems by Timed Petri Nets,” MIT, Project MAC, Technical Report 120, 1974.

[10] J. Sifakis, “Use of petri nets for performance evalutation,” in Measuring, modelling and evalutating computer systems, ser. IFIP, H. Beilner and E. Gelenbe, Eds., North Holland Publ. Co., 1977, pp. 75–93.

[11] R. König and L. Quäck, Petri-Netze in der Steuerungs- und Digitaltechnik. München, Wien: Oldenbourg Verlag, 1988.

[12] H.-M. Hanisch, Petri-Netze in der Verfahrenstechnik. München: Oldenbourg, 1992.

[13] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzè, “A unified high-level petri net formalism for time-critical systems,” IEEE Transactions On Software Engineering, vol. 17, no. 2, 1991, pp. 160–172.

[14] H.-M. Hanisch, K. Lautenbach, C. Simon, and J. Thieme, “Timestamp Nets in Technical Applications,” in IEEE International Workshop on Discrete Event Systems, San Diego, CA, 1998.

[15] C. Simon, “Developing Software Controllers with Petri Nets and a Logic of Actions,” in IEEE International Conference on Robotics and Automation, ICRA 2001, Seoul, Korea, 2001.

[16] K. Jensen, “High-Level Petri Nets,” Informatik-Fachberichte, vol. 66, 1983, pp. 166–180.

[17] L. Popova-Zeugmann, Time and Petri Nets. Berlin: Springer, 2013.

[18] C. E. Knoeppel, Installing Efficiency Methods. The Engineering Magazine, 1915.

[19] T. Ohno, Toyota Production System. Milton Park, UK: Taylor & Francis, 1988.

[20] BPMI, “BPMN 1.0 - Business Process Model and Notation,” https://www.omg.org/spec/BPMN/ (letzter Zugriff 2020.09.20), 2004.

[21] OMG, “BPMN 2.0 - Business Process Model and Notation,” http://www.bpmn.org/ (last accessed 2020.09.20), 2011.

[22] C. Simon, “Web-Based Simulation Of Production Schedules With High-Level Petri Nets,” in 32rd International ECMS Conference on Modelling and Simulation (ECMS 2018), L. Nolle, A. Burger, C. Tholen, J. Werner, and J. Wellhausen, Eds. Wilhelmshaven, Germany: SCS Europe, 2018, pp. 275–281.