Automated Digital Twins Generation for Manufacturing Systems: a Case Study

Giovanni Lugaresi * **Andrea Matta** *

* *Department of Mechanical Engineering, Politecnico di Milano*
(e-mail: giovanni.lugaresi@polimi.it)

Abstract:
The recent industrial scenario was defined by the emergence of digital twins and cyber physical systems as key elements for manufacturers leadership. Digital models can perform good in terms of production planning and control decisions if they are correctly representing their physical counterparts at anytime. Discrete event simulation can be considered as established digital models of manufacturing system, thanks to the proven capabilities of correctly estimating the system performances. Automated simulation model generation techniques can significantly reduce model development phases and allow for using simulation models for short term decisions in production. Application studies and test cases are scarce in the literature. In this paper, we present the application of a digital model generation method. The test case is done exploiting a lab-scale model of a manufacturing system composed by six stations. We investigate how the model generation works online, during the transient phase of a manufacturing system. Results confirm the real-time applicability of the approach provided that sufficient data points are available from the production event logs.

Keywords: Industry 4.0, Simulation, Digital Twins, Process Mining

1. INTRODUCTION

Digital twins are considered among the key components that determined the success of Industry 4.0 [Liu et al. (2020)]. Besides, the latest developments in industry moved towards new technologies for data acquisition and storing. New possibilities for exploiting data made it possible to improve manufacturing systems performances with new management policies and tools [Tao et al. (2019)]. The exploitation of real-time data streams in manufacturing means the shop floor status can be instantly available anytime [Chen et al. (2016)], which allows for taking online data-driven decisions and reaching quick-response capabilities. Within production planning and control applications, discrete event simulation can be effectively used as digital twin of a manufacturing system. Real-time Simulation is a concept that involves using simulation as digital model of a system with the goal to take accurate decisions based on the current system state [Lugaresi and Matta (2018)]. However, the ability to take online decisions is strongly based on the assumption that models are properly aligned with the real system. As a consequence, practical implementations of digital twins remain scarce due to the challenges of real time alignment.

This paper presents the application of a data-driven model generation method [Lugaresi and Matta (2021)]. Provided the availability of real-time data, such approach can also be applied online. The real-time application allows for adapting simulation models to the real system counterpart, potentially at any time a modification occurs. In this work, the method is applied for the online development of the simulation model for a lab-scale model of a manufacturing system. A set of features of the real system is observed during the model generation phase. The scope of the work is to prove the applicability of the model generation procedure in a real-time setting, and to observe how the the observation length influences the discovery of model parameters.

The rest of the paper is organized as follows: section 2 summarizes significant contributions for automated model development and digital twin deployment in manufacturing; section 3 describes the model generation method used in this work; section 4 outlines the test case that has been selected; section 5 summarizes the numerical results, and section 6 presents our final remarks.

2. RELATED WORKS

The automated generation of a simulation model can be defined by the following main steps. First, data are collected from the real system and organized in event logs [Perera and Liyanage (2000)]. Then, starting from the data available, the material flows are identified. Given the sequence of activities performed by each part in the system, the structure of the production process can be inferred (e.g., system layout, operations precedences) [Van Der Aalst (2016)]. Parameters such as processing and waiting times can be retrieved from the data, together with the statistical distributions describing the production dynamics. Also the rules followed by resources and operators in the system (e.g., routing, part-mix, dispatching rules) can be retrieved from data [Ferreira and Vasilyev (2015)]. Finally, the generated simulation models are validated before being used to take decisions [Sargent (2013)].

Recent approaches exploited Process Mining (PM) for the automated generation of simulation models. PM is a discipline aiming to discover and exploit valuable information from event logs available in information systems [Van Der Aalst (2016)]. Given the data-based nature and the fact that it is application agnostic, PM has been applied in numerous fields.

2.1 Process Mining-based Model Generation

Rozinat et al. (2009) can be considered as the first complete assessment on how PM can support the generation of simulation models. Bergmann et al. (2015) introduced a methodology for recognizing the production policies applied on a manufacturing system. Milde and Reinhart (2019) developed an approach for discovering the material flow, estimating parameters, and identifying the control policies from manufacturing systems event logs. Martin et al. (2015) improved interarrival times modeling by including parts queuing at the entrance of the system in a mining algorithm. The authors used the proportion of entities queuing at arrival identified from the event log to approximately estimate the parameters of the interarrival times distribution. Ferreira and Vasilyev (2015) combined PM with logical decision trees to understand the causes of process delays. Martin et al. (2016a) used PM to retrieve daily availability records from an event log, by considering resource availability with both a temporal dimension and the possibility of intermediate interruptions. Martin et al. (2017) designed an algorithm to mine how operational activities are batched within a production environment. Denno et al. (2018) developed a methodology to mine the production system structure and used genetic programming to link colored Petri Net states and exceptional system states. Halaška and Šperka (2019) benchmarked five recent PM algorithms for process discovery in their ability to discover a digital model of a job shop. The authors highlighted how discovery algorithms perform better overall with more extensive event logs when discovering complex models: the more information is available in the event log, the better process models are produced by discovery algorithms. Lugaresi and Matta (2020) developed a method for generating simulation models with a proper level of detail.

2.2 Field-connected Digital Twins

Digital twins are among the most important constituents of the latest Industry 4.0 revolution. In their basic form, digital twins are representations of physical systems, able to replicate their behavior and correctly estimate the real system performances [Negri et al. (2017)]. Cyber physical Systems (CPSs) are defined in conjunction with digital twins. Differently, for CPS it is highlighted their capability to be used as tools for decision making. For instance, Monostori et al. (2016) have defined the main modes for simulation models used within CPSs: (1) offline simulation is used for sensitivity analysis and robustness evaluation of production rules before their implementation, (2) proactive simulation is used online with the aim of defining corrective actions after the recognition of potential deviations from the optimal plan, and (3) reactive simulation is used online for the evaluation of alternatives following disruptions such as machine failures or unplanned maintenance interventions. A central capability of digital twins and CPSs is to mirror the real system state at any time. The concept of real time is strongly related to the decision to be made. For instance, a scheduling problem might be updated weekly, with daily system state refresh, while a dispatching or routing problem has to be solved within minutes, if not seconds. Negri et al. (2020) have addressed the capabilities of a field-synchronized digital twin with a flow shop system state estimation from field measurements. The data samples are used to constantly update a real system status estimate, which is used to synchronize a digital model of the system to correctly estimate the future performance. In this work, we exploit a real-time stream of shop-floor data for the online development of a manufacturing system simulation model. Model generation is done exploiting a PM-based approach adapted for manufacturing applications.

3. DIGITAL TWIN GENERATION

In this section, we describe the input data which is assumed available for the online model generation. Then, we outline the relation between data and simulation models for manufacturing systems. Finally, we describe the digital twin generation approach that we have developed, which is tested in section 4.

3.1 Event Logs

Event logs are files containing information about parts flowing in the system (e.g., serial codes associated to the parts, activity time stamps). Let us assume that all data from the manufacturing system sensors are aggregated and collected in an event log file. In general, the event log may contain several types of information, such as parts flows, resources identifiers, and quality check outcomes. In this work, we will concentrate on an event log which contains three information types: (1) the activity identifier $a \in \mathbb{A}$, (2) the work-piece identifier $i \in \mathbb{I}$, and (3) the timestamps $t_S(a, i)$ and $t_F(a, i)$ indicating the moment in which the a-th activity has started and finished on the i-th work-piece, respectively. Table 2 shows an example of event log.

3.2 Simulation Model Components

In the following, we describe the simulation model components as linked with the available data in an event log. We refer to the schema proposed by Martin et al. (2016b), with a focus on discrete part manufacturing processes.

Entities represent the objects that move in the system, and **activities** can be performed on them by a set of **resources**. Entities be identified through a one-to-one correspondence with the part identifiers $i \in \mathbb{I}$ in the log. Similarly, activities may correspond to the unique set of $a \in \mathbb{A}$. However, more complex assignments can be performed. For instance, if a codification for identifiers is known, grouping could be allowed based on parsing IDs. Another option is to used entity types (e.g., small parts, large parts) as guidelines for grouping. Entities can also represent clusters of work-pieces, perhaps obtained as a result of trace clustering [Song et al. (2008)].

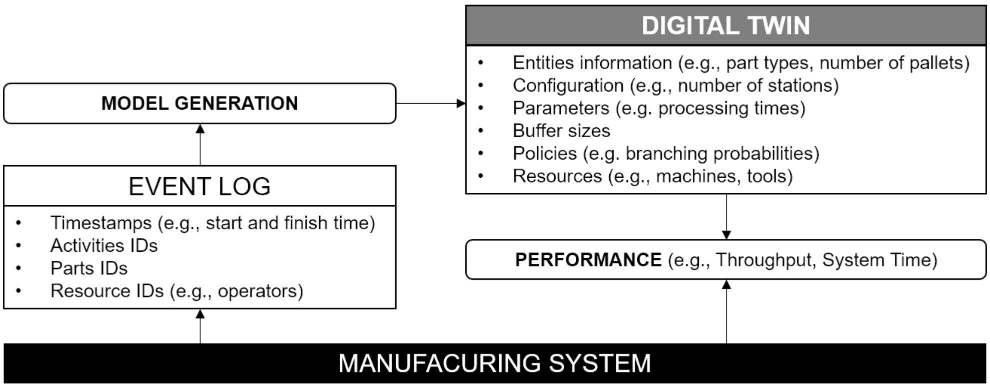

DIGITAL TWIN
Entities information (e.g., part types, number of pallets)
Configuration (e.g., number of stations)
Parameters (e.g. processing times)
Buffer sizes
Policies (e.g. branching probabilities)
Resources (e.g., machines, tools)
MODEL GENERATION
EVENT LOG
Timestamps (e.g., start and finish time)
Activities IDs
Parts IDs
Resource IDs (e.g., operators)
PERFORMANCE (e.g., Throughput, System Time)
MANUFACURING SYSTEM

Fig. 1. Overview of the automated digital twin generation.

Buffers are components that can store entities if the required resources cannot be allocated. In manufacturing environments, the locations where work-in-progress can accumulate are not always known. Therefore, in general, each activity could possess an input buffer. In the proposed model generation method, we have developed a queue-size estimation algorithm which relies on the maximum number of parts observed in accumulation between two activities. Namely, a part i has queued in front of activity a if $t_F(a-1,i) \leq t_S(a,i-1)$. The queue size estimation depends on the observation length, hence it tends to underestimate the buffer size. Martin et al. (2015) have highlighted how it is important to also consider queue disciplines (e.g., first-in-first-out, last-in-first-out) for an accurate simulation model building.

Policies are generally associated with gateways and routing rules. For instance, the priority rules governing a material flow split among two alternative activities may be inferred from the event log. Among other techniques, machine learning can be applied to understand the resource association or priority rules [Milde and Reinhart (2019)]. Also the policy identification strictly depends on the availability of data in the event log and even simple routing decisions can be biased if the input data is scarce.

Parameters such as processing times may be estimated with several techniques, such as statistical distribution fitting, kernel density estimation, or empirical distributions. Since event logs store timestamps of the recorded events, it is directly possible to estimate the statistical distributions that better fit the data. As a consequence, the fitting capabilities strictly depend on the availability of enough data points. Notice that for this purpose, activity type tags *start* and *finish* are essential to estimate the duration of activities [van der Aalst (2015)]. Such additional tags in the event log specify if the corresponding event refers to an arrival, the starting or finishing time, or the exit of a part. This information directly influences the obtainable performance of the mined simulation model. For instance, if arrival tags are not specified, arrival rates are in fact unknown and have to be estimated [Martin et al. (2015)].

3.3 Online Model Generation

Model generation is the procedure which links the system data contained in the event log with a simulation model. A model is generated by representing all the relational properties (i.e. precedences) in a directed graph in which nodes correspond to the activities and arcs express the material flow relationships between activities. The complete model generation phase has been defined by Lugaresi and Matta (2021). Hereby, we describe the main steps. Firstly, a unique set of activities $\mathbb{A}$ is created. The next step is the identification of the *traces*. A trace is the specific route that each part followed in the system. It can be expressed as a sequence of activity identifiers. Each i-th part has a corresponding trace $\theta_i = \{a^{(1)}, a^{(2)}, \dots, a^{(e)}, \dots, a^{(N_i)}\}$ where N_i is the number of the activities performed by the i-th part and e indicates the sequential position of the a-th activity as observed in the log for part i. For example, with reference to the event log portion of Table 2, $\theta_1 = \{1, 2\}$. The traces are used to retrieve precedence relationships between activities. Hence, a node exists in the model if a certain activity has been performed by at least one part in the system, and an arc indicates that a production step has followed another in at least one trace. Specifically, arc (a, b) exists if $\exists\, i \in \mathbb{I} | t_F(a,i) < t_S(b,i)$. The following step is populating the graph model with the properties of nodes and arcs. Specifically, with the goal of representing the simulation model elements described in section 3.2, the graph model can then be enriched with properties of nodes and arcs. For instance, the finite capacity of a conveyor between two activities is be expressed as a property of the corresponding arc. Finally, a graph model can be converted into Petri Nets or Event Relationship Graphs.

4. TEST CASE: LAB-SCALE PRODUCTION SYSTEM

As a test case for the application of the model generation procedure, we have exploited lab-scale models of manufacturing systems. Such models have been developed with the scope of testing Real-time Simulation applications in a controlled environment [Lugaresi et al. (2020)]. In general, the system is composed by stations representing manufacturing activities, and conveyors which bring material from a station to another. Parts are represented by wooden discs of 35 mm-diameter tagged with colored plates. Stations are controlled by LEGO®-EV3 intelligent bricks which are programmed using customized Python scripts (EV3DEV OS). Each station has its own configuration such that different distributions of the processing and repair times can be assigned to different stations. Machine conditions of working (i.e., a part is loaded in the station), starvation (i.e., the upstream buffer is empty), and blocking (i.e.,

Table 1. Test Case: parameters of the lab-scale model depicted in Figure 2.

Station s	Upstream Buffer Capacity b_s	Processing Time p_s $[s]$	Failure Probability f_s	Repair Time r_s $[s]$
1	4	1	0.15	UNIF(5,60)
2	3	1.5	0.1	UNIF(5,60)
3	6	1.1	0.35	EXPO(1)
4	6	1	0.34	Max(0.5, NORM(4,2))
5	2	Max(2, NORM(2, 10))	0	0
6	4	2.5	0	0

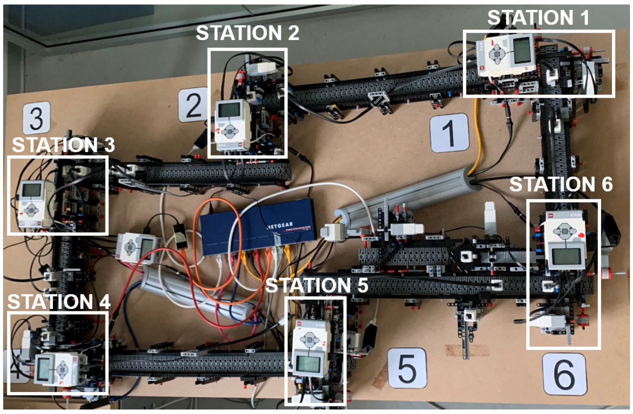
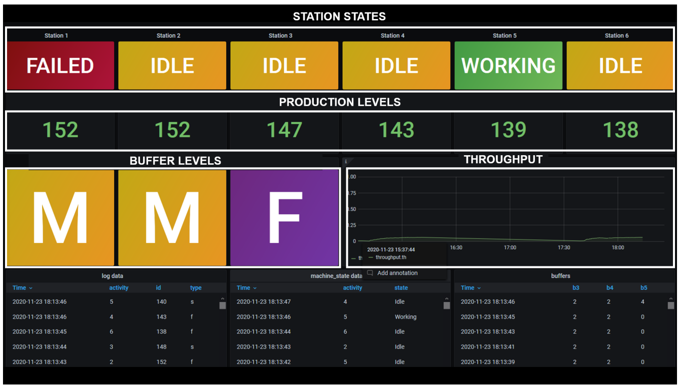


Fig. 2. Test Case: 6-station flow line used for the numerical analysis of this work.



log data

Time	activity	id	type
2020-11-23 18:13:46	5	140	s
2020-11-23 18:13:46	4	143	f
2020-11-23 18:13:45	6	138	f
2020-11-23 18:13:44	3	148	s
2020-11-23 18:13:43	2	152	f

machine_state data

Time	activity	state
2020-11-23 18:13:47	4	Idle
2020-11-23 18:13:46	5	Working
2020-11-23 18:13:44	6	Idle
2020-11-23 18:13:43	2	Idle
2020-11-23 18:13:42	5	Idle

buffers

Time	b3	b4	b5
2020-11-23 18:13:46	2	2	4
2020-11-23 18:13:45	2	2	0
2020-11-23 18:13:43	2	2	0
2020-11-23 18:13:41	2	2	0
2020-11-23 18:13:39	2	2	0

Fig. 3. Test Case: real-time dashboard showing the state of the system of Figure 2.

Table 2. Test Case: portion of the event log generated by the lab-scale model of Figure 2.

Time-stamp	Part-ID	Activity-ID	Type
2020-11-23 16:37:40	1	1	start
2020-11-23 16:37:44	1	1	finish
2020-11-23 16:37:47	2	1	start
2020-11-23 16:37:51	2	1	finish
2020-11-23 16:37:52	1	2	start
2020-11-23 16:37:54	3	1	start
2020-11-23 16:37:57	1	2	finish

the downstream buffer is full) are supervised through specific sensors. All system controllers are connected through a local network and a PC through Secure Shell Host connection protocol. Data are collected in a time-series database that constitutes the event log. Further, a real-time dashboard (Figure 3) aggregates the event log data and shows an overview of the system status at anytime (e.g., parts produced by each station). Additional details about the physical model are available in related works [Lugaresi et al. (2020)].

In this work, we use data generated by a 6-station flow line lab-scale model. A schematic representation of the line is shown in Figure 2. The physical system is a closed-loop production line composed by six stations with intermediate conveyors that operate also as buffers. We denote with b_s the buffer capacity before station s. The blocking after service rule is applied.

A fixed number of pallets ($n = 20$) circulates into the system. It is assumed that station $s = 1$ is the load/unload station and a large number of unprocessed parts are waiting in front of the first station, and that a finished part can immediately leave the system. Each station can process one part at the same time. Production activities are represented by the time p_s that a station holds a part before releasing it to the downstream conveyor. Stations $s \in [1, 4]$ are unreliable and may fail with probability f_s. If a failure occurs, the part is held by the station for an additional amount of time r_s, which accounts for the station repair. All stochastic quantities are sampled each time a part enters a station. The quantity $\phi_s(i)$ represents the whole operation time that the i-th part spends in a station, and its realization is as follows:

$$\phi_s(i) = \tilde{p}_s + \tilde{I}_s \tilde{r}_s \tag{1}$$

where $\tilde{I}_s$ is an indicator function which is 1 if $u < f_s$, 0 otherwise. u is a random number in the interval $[0, 1]$ and it is sampled each time a part enters a station. Conveyors move at a constant speed. Table 1 reports the parameters of the lab-scale model[1] .

5. EXPERIMENTS

In this section we present the experiments and the numerical results.

5.1 *Experiments description*

The system described in section 4 has been used to produce parts for around 1 hour. This production experience has been repeated 9 times among different days, hence 9 independent event logs are available. We have applied the model generation procedure with an online setting. For practical convenience, experiments have been done in a separate time. Namely, we have selected the portion of the log such that $t_F(a, i) \leq \tau \, \forall i \in \mathbb{I}, a \in \mathbb{A}$, where $\tau \in \{25, 35, 45, 60, 100\}$, corresponding to the time to

[1] Although unrealistic, high failure probabilities have been set to increase the amount of failures that can be observed within a production session.

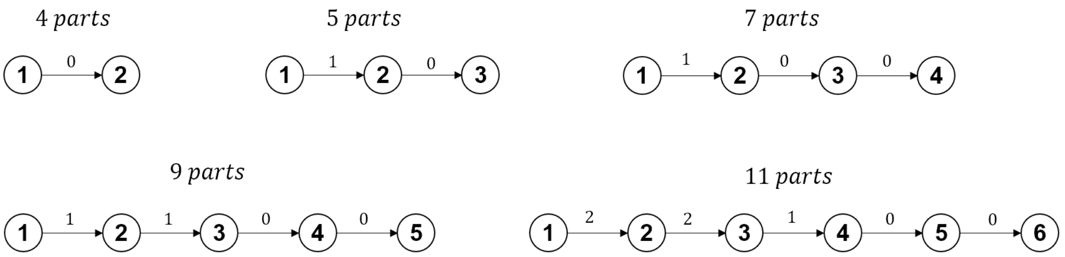


Fig. 4. Test Case: discovered models at different times (first log). Arcs are tagged with the respective buffer sizes.

produce 4,5,7,9, and 11 parts, respectively. The goal is to observe the automated model building behavior during the initial transient phase. Further, we have recorded the models built for τ up to 2000 seconds, corresponding to the production of 142 parts. The goal is to assess if the parameters of the system are correctly estimated given a certain amount of data. Finally, we have used the nine independent logs to check how parameters fitting techniques behave with different sample sizes. Namely, we have used (1) Kernel Density Estimation and (2) Empirical Cumulative Distribution Function to estimate the distribution of operation time ϕ_5.

5.2 Results

Figure 4 shows the models developed from the first event log, depending on different values of τ. It can be noticed how the initial transient determines the capability of discovering the correct model of the system. Indeed, a time of $\tau = 100\,s$ is needed for obtaining the correct 6-station model. From Figure 4 we can also notice that the discovered buffer sizes have not reached the real values even after $100\,s$. Figure 5 shows the buffer capacities estimated by the model development method for $\tau \in [0, 2000]$. The convergence of all buffer capacities is reached after $\tau = 600\,s$. However, the convergence is not sufficient to obtain a correct estimation of buffer capacities. Indeed, the size of b_3 is biased by one slot even after $\tau > 2000\,s$. Figure 6 shows the estimation of the processing time ϕ_5 among different days. We may observe that one day is not enough for an estimation of the distribution, two days can be sufficient for a rough estimation of the distribution parameters (e.g., first moment), while the correct estimation of the distribution is reached at 9 days, despite a biased estimation of the first moment which is mostly due to noise of the real data. From the results we may also infer that KDE and ECDF produce comparable results for the estimation of operation times.

6. FINAL REMARKS

The recent industrial scenario and the Industry 4.0 revolution allowed for the introduction of technologies that can support the generation and alignment of digital twins of manufacturing systems. In this work, we have investigated the behavior of a model generation method in terms of system topology identification and parameter estimation capabilities during a transient phase. The buffer capacity estimation is still biased for some of the buffers if not enough observations are available in the event log. In the future, more tests shall be done on digital instances derived from more complex systems and realistic sized databases. Next developments should also assess the capability of correctly estimating other elements such as production policies and resource utilization. Last but not least, the value of additional information which may be available in the event logs shall be assessed.

REFERENCES

Bergmann, S., Feldkamp, N., and Strassburger, S. (2015). Approximation of dispatching rules for manufacturing simulation using data mining methods. In *2015 Winter Simulation Conference (WSC)*, 2329–2340. IEEE.

Chen, C.C., Chen, C.L., Ciou, C.Y., and Liu, J.X. (2016). Communication scheduling scheme based on big-data regression analysis and genetic algorithm for cyber-physical factory automation. In *Proceedings of the 2016 IEEE SMC Conference*, 2603–2608. IEEE.

Denno, P., Dickerson, C., and Harding, J.A. (2018). Dynamic production system identification for smart manufacturing systems. *Journal of manufacturing systems*, 48, 192–203.

Ferreira, D.R. and Vasilyev, E. (2015). Using logical decision trees to discover the cause of process delays from event logs. *Computers in Industry*, 70, 194–207.

Halaška, M. and Šperka, R. (2019). Performance of an automated process model discovery–the logistics process of a manufacturing company. *Engineering Management in Production and Services*, 11(2), 106–118.

Liu, M., Fang, S., Dong, H., and Xu, C. (2020). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*.

Lugaresi, G., Alba, V., and Matta, A. (2020). Lab-scale models of manufacturing systems for testing real-time simulation and production control technologies. *Journal of Manufacturing Systems*.

Lugaresi, G. and Matta, A. (2018). Real-time simulation in manufacturing systems: Challenges and research directions. In *2018 Winter Simulation Conference (WSC)*, 3319–3330. IEEE.

Lugaresi, G. and Matta, A. (2020). Generation and tuning of discrete event simulation models for manufacturing applications. In *Proceedings of the 2020 Winter Simulation Conference*. IEEE.

Lugaresi, G. and Matta, A. (2021). Automated manufacturing system discovery and digital twin generation. *Journal of Manufacturing Systems*, 59, 51–66.

Martin, N., Bax, F., Depaire, B., and Caris, A. (2016a). Retrieving resource availability insights from event logs. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*, 1–10. IEEE.

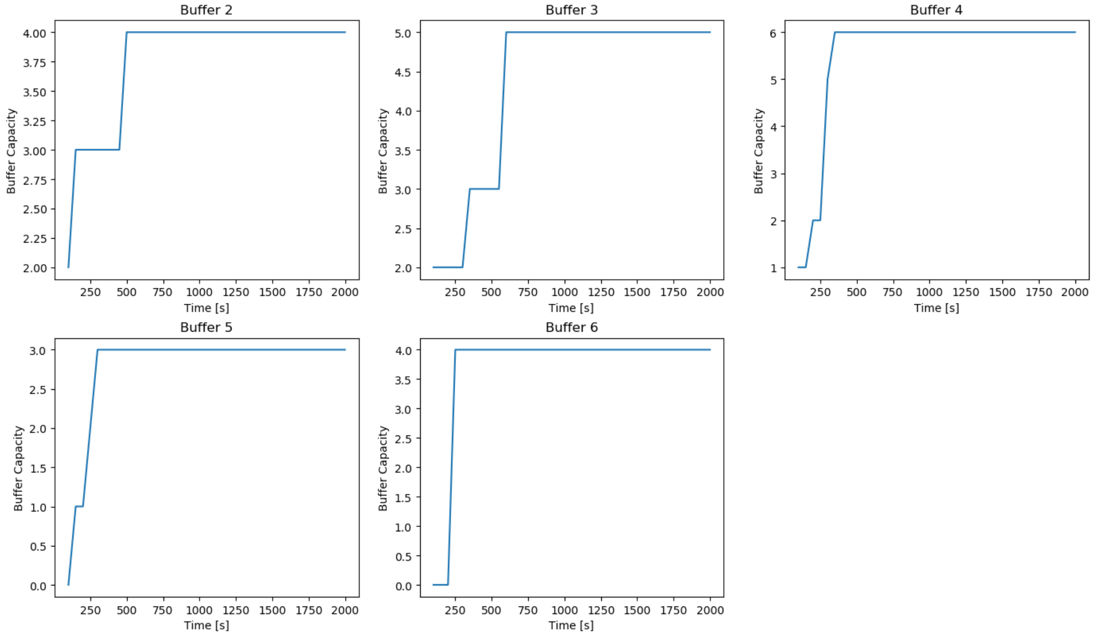


Fig. 5. Test Case: discovered buffers sizes depending on time.

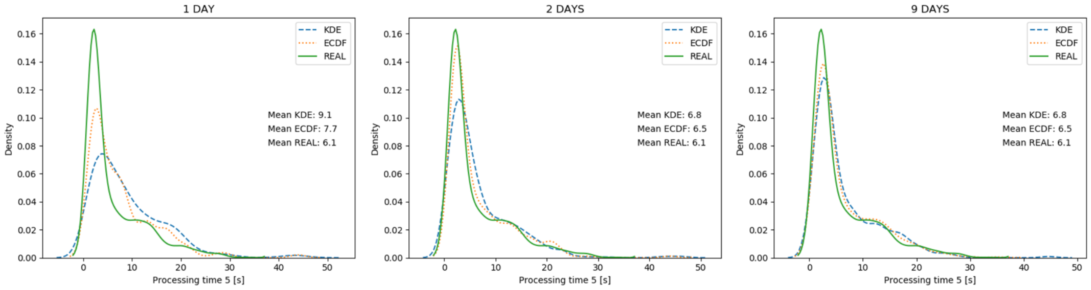


Fig. 6. Test Case: comparison among Kernel Density Estimation (KDE) and Empirical Distribution Function (ECDF) for the estimation of the operation time ϕ_5.

Martin, N., Depaire, B., and Caris, A. (2015). Using process mining to model interarrival times: investigating the sensitivity of the arpra framework. In *2015 Winter Simulation Conference (WSC)*, 868–879. IEEE.

Martin, N., Depaire, B., and Caris, A. (2016b). The use of process mining in business process simulation model construction. *Business & Information Systems Engineering*, 58(1), 73–87.

Martin, N., Swennen, M., Depaire, B., Jans, M., Caris, A., and Vanhoof, K. (2017). Retrieving batch organisation of work insights from event logs. *Decision Support Systems*, 100, 119–128.

Milde, M. and Reinhart, G. (2019). Automated model development and parametrization of material flow simulations. In *2019 Winter Simulation Conference (WSC)*, 2166–2177. IEEE.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2), 621–641.

Negri, E., Fumagalli, L., and Macchi, M. (2017). A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, 11, 939–948.

Negri, E., Pandhare, V., Cattaneo, L., Singh, J., Macchi, M., and Lee, J. (2020). Field-synchronized digital twin framework for production scheduling with uncertainty. *Journal of Intelligent Manufacturing*, 1–22.

Perera, T. and Liyanage, K. (2000). Methodology for rapid identification and collection of input data in the simulation of manufacturing systems. *Simulation Practice and Theory*, 7(7), 645–656.

Rozinat, A., Mans, R.S., Song, M., and van der Aalst, W.M. (2009). Discovering simulation models. *Information systems*, 34(3), 305–327.

Sargent, R.G. (2013). Verification and validation of simulation models. *Journal of simulation*, 7(1), 12–24.

Song, M., Günther, C.W., and der Aalst, W.M.P. (2008). Trace clustering in process mining. In *International conference on business process management*, 109–120. Springer.

Tao, F., Zhang, H., Liu, A., and Nee, A.Y.C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, 15(4), 2405–2415.

Van Der Aalst, W. (2016). Data science in action. In *Process mining*, 3–23. Springer.

van der Aalst, W.M.P. (2015). Extracting event data from databases to unleash process mining. 105–128. Springer.