Manufacturing Systems Mining: Generation of Real-Time Discrete Event Simulation Models

Giovanni Lugaresi, Marco Zanotti, Diego Tarasconi, Andrea Matta

Manufacturing Systems Mining: Generation of Real-Time Discrete Event Simulation Models

Giovanni Lugaresi[1], Marco Zanotti[1], Diego Tarasconi[1], Andrea Matta[1]

***Abstract*— The recent economic outlook has prompted manufacturers to spend a lot of resources towards automation and Cyber Physical Systems (CPS). One of the requisites to successfully deploy CPSs is the availability of up-to-date digital models coupled with the real system, yet this is not always guaranteed in dynamic and complex environments such as production systems. This paper develops a new method that generates the Petri Net model of a manufacturing system starting from an event log with three data labels. The user decides the number of maximum events to be mapped to control the model level of detail. The method has been applied on a test case and it is promising in terms of applicability to real manufacturing systems.**

I. INTRODUCTION

Recent economic pressure has driven manufacturers towards partial or complete automation of production facilities [1]. Besides, electronic devices such as sensors and data acquisition systems are regularly deployed in manufacturing environments and it is possible to obtain information about the shop floor status almost anytime [2]. *Cyber Physical Systems* (CPS) are based on the coexistence of the real system with its digital counterpart, often called digital twin or digital model [3], [4].

With a planning and control scope, the digital model of a manufacturing system can be represented by Discrete Event Simulation (DES). However, if the model is not a close representation of the actual system, any conclusion derived from simulation results is likely to be erroneous and may culminate in expensive decisions. The possibility of generating digital models with a higher frequency (i.e. through an automated procedure) is crucial for achieving the application of simulation models for short-term decision-making (also called Real-time Simulation models), particularly in highly dynamic environments [5].

Process Mining (PM) is a recent discipline aiming to discover new or hidden information from event logs available in manufacturing systems [6]. Event logs are files containing information about parts flowing in the system (e.g., serial codes associated to the parts, time stamps of each activity, activity tags, so on). PM is defined as composed by three main parts [6]: (1) *system discovery*, that is finding the main logical relationships between activities, (2) *conformance checking*, that finds deviations from a standard model used for comparison, and (3) *enhancement*, that is the process of strengthening some properties of interest of the analysed system model. PM is currently used for multiple purposes, for instance in social network or organizational mining (i.e. Business Process Modeling).

The scope of this work is to discover a production system configuration in an automated way starting from an event log with three data labels, then building a Petri Net model of the manufacturing system. Further, we wish to tune the model generation procedure by fixing the maximum number of activities allowed. This way the user may decide the level of detail of the obtained simulation model. The proposed approach is addressed at production systems applications, hence we refer to it as *Manufacturing Systems Mining (MSM)*.

II. STATE OF THE ART

Most mining algorithms from the literature exploit a generic event log composed by a certain number n of activities $\Psi = \{\psi_1, \dots, \psi_n\}$, each performed by at least one of the parts (i.e. products). Further, it is important to understand to which path the activities belong. For example, consider the simple system of Figure 1a, with three different stations: S1, S2, and S3. Station S1 performs activity a, S2 performs activity b, and S3 activity c. If a batch of P parts has to be processed, it is fundamental to associate each flowing item to the right path: (a, c) or (b, c). Notice that usually the whole set of available paths for parts is not known a priori, and in order to recreate the production sequence an *activity mining* step is necessary. This procedure characterizes the possible sequences of events in the system. Starting from the event log, the paths are built through a string recognition of the activities encountered by each of the P parts in the event log. The next step is the so-called *footprint*, which is a set of causal dependencies regarding the type of connections among the activities in the event log. Usually it can be expressed by a set of strings $\mathbf{F} = \{f_1, \dots, f_P\}$, hence the p-th part performs the activities in the order defined by $f_p \in \mathbf{F}$. For instance, consider the production line of Figure 1a. The activities are $\Psi = \{a, b, c\}$. If the technological cycle of the p-th product type prescribes the visit of two stations in the order (S1, S3), then its corresponding footprint will be $f_p = ac$.

A. The α-mining Algorithm

One of the most used process mining algorithms is the α-mining algorithm [7], which models the activities in the event log as transitions in a Petri Net. The composition of the Petri Net model is done by following the relationships among the activities Ψ of the event log. Since the α-mining algorithm is focused on the discovery of transitions relationships, during

[1] Department of Mechanical Engineering, Politecnico di Milano, Via La Masa 1, 20156 Italy giovanni.lugaresi@polimi.it

the composition of the Petri Net, places must be added in order to maintain the correct alternation between places and transitions. Applications of the α-mining algorithm range between healthcare [8], forensics [9], and banking [10].

B. Other Mining Approaches

Other mining algorithms have been developed as improvements of the α algorithm. We may divide them in three main categories: (1) Heuristic Miners, (2) Genetic Miners, and (3) Fuzzy Miners.

Heuristic Miners use a frequentist approach to discover the different weights related to sequences of activities. Also, timestamps associated to the activities are considered for ordering them and give a perception of the temporal precedences. One of the most known work related to mining algorithms is from Weijters et al. [11], which is based on the creation of a frequency-based dependency graph and takes into account AND/XOR relationships (e.g., two parallel machines). Cook et al. [12] developed an approach focused on mapping the analyzed system which retrieves a model describing the most frequent paths in the event log. This approach aims at overcoming the issues associated to the noise which may populate a generic event log (for instance, due to wrong data entries from sensors along the system).

Genetic Miners algorithms are adaptive search methods capable of following the process evolution starting from an initial population or an initial structure attempt. Genetic algorithms search for a result which is fitting as globally as possible using a comparison technique between the created structure and the possible others [13].

Fuzzy Miners algorithms exploit clustering to remove activities considering the correlation among different types of events. Indeed, they cluster highly correlated activities into a single node with a common attribute, while the most isolated nodes are removed from the representation [13]. These algorithms are typically used for unstructured processes, in which there is no trivial distinction between what is important and what could be discarded. Greco et al. [14], [15], [16] aims at discovering a hierarchical tree of process models that describes the event log at different levels of detail. Gunther et al. [17] developed a new fuzzy miner algorithm through an attribute analysis and a consequent abstraction of the mined event log.

Other approaches use PM for several applications. Among others, Van der Aalst et al. [18] described a particular heuristic approach used for mining the invoices flows in one of the provincial offices of the dutch national public works department. Gouveia [19] illustrated the process mining application for a business platform for software development. Van Dongen [20] introduced a clustering method in order to apply process mining to a dutch municipality real-life case.

C. Limitations of Mining Approaches

Mining algorithms have some limitations and we list the most relevant as follows.

- The availability of all the needed information in the event log is not always guaranteed. For instance, serial numbers (or, in general, IDs) of the parts and the activities are used for identifying the relationships among themselves and to discover the system structure. These pieces of information are essential for a correct process mining activity [11].
- The α-mining in its basic form does not consider the frequencies with which activities are performed. Indeed, only logical relationships among activities through parts flowing are taken into account, which leads to a partial view of the system behavior. Without considering occurrences, a path that is done a lot of times will have the same weight of one that is done only rarely.
- In case of components of the system with finite capacity such as machines and buffers, it is necessary to force the entities flowing the system to respect capacity constraints. In a Petri Net, this corresponds to to limit the number of tokens allowed to flow in certain transitions.

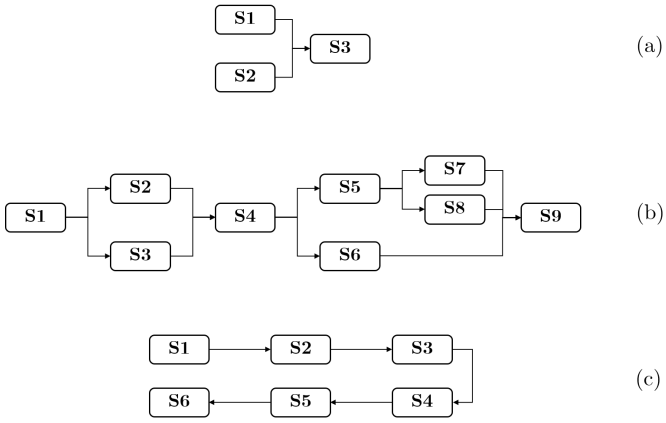


Fig. 1: Sample production systems – (a) 3-station system, (b) 9-station system, (c) 6-station system.

D. Problem Introduction

Let us consider a manufacturing system such as the 9-station line as in Figure 1b and assume its event log is available with at least the following information:

- the product-IDs (entities);
- the activities performed by the entities;
- the timestamps of each activity done by each entity.

The log holds a lot of knowledge from the system and we can think of exploiting the data in the event log to build a discrete event simulation model of the line. Additionally, we may want to obtain a simulator that does not reach the same level of detail of the event log, and to define the maximum number of activities to be modeled (for instance, only seven activities instead of nine).

The aim of this work is to develop an algorithm for process discovery through event logs from manufacturing systems and suitable for the on-line generation of simulation models. The procedure must be compliant with a Real-time Simulation framework [5], hence the system discovery has to be performed automatically, without pre-existing models nor any user interaction during the computation. Initial steps in this direction have already been outlined by [21].

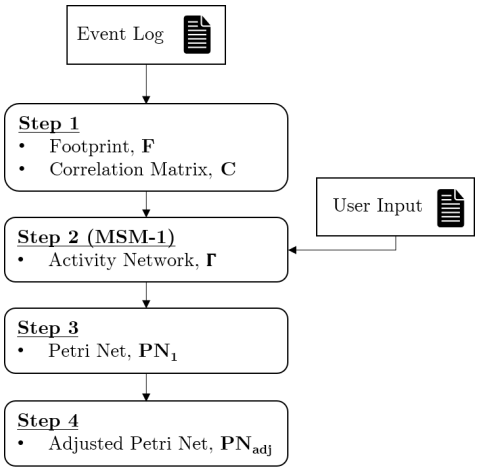


Fig. 2: High-level structure of the proposed approach.

In our procedure, we combined the properties of the mining methodologies presented in section II-B. The frequentist approach (Heuristic Miners) is based on the occurrence of event types and this is coherent with manufacturing systems where multiple paths for items may be present. The genetic approach is visible in the optimization problem which will be further explained in section III. The influence of the Fuzzy Miners approaches is visible from the classification at input phase, in which the user is free to tune the number of events to be mapped or to be discarded.

III. A NEW MINING APPROACH

A. Working Procedure

We have developed a general working procedure which allows for mining manufacturing systems event log datasets. Here, Petri Nets [22] are used – without loss of generality – as formal language capable of reflecting the relationships among activities and entities populating the physical system. Further, we opted for an optimization approach in which the level of detail of the system model can be controlled by the user. Also the selective representation of just a portion of activities or entities is possible. The main steps of our approach are described as follows:

- **Step 1 – Dataset loading.** The event log dataset is used to generate the footprint and a correlation matrix.
- **Step 2 – Optimized mining of event types.** The modified mining algorithm finds the optimal network. This is done by maximising a quantity called *reproducibility*, which takes into account both the logical relationships among the events and the occurrences of arcs and nodes. The result is an activity network. Further details about the proposed method for this step are in section III-B.
- **Step 3 – Petri Net model generation.** At this step we may add places between the activities of the activity network generated in Step 2, thus obtaining the Petri Net model PN_1.
- **Step 4 – Petri Net model adjustment.** The final step is related to adjustments of the PN_1 model in order to obtain a formally correct Petri Net model of the system. Shortly, these steps consists in:
  1) addition of *token-generation* and *token-disposal* places;
  2) addition of places after all transitions;
  3) removal of places in order to guarantee that no tokens are disposed during the simulation, hence that tokens in the PN may represent physical parts in the manufacturing system;
  4) mining finite capacities (e.g., buffers). This step is done through a searching procedure that is not further described in this work for brevity. The reader is referred to [23] for further details. Finite capacities are modeled as additional places between two transitions.

Figure 2 briefly presents the aforementioned steps. The final result is a Petri Net model PN_{adj}.

B. Proposed Method for Step 2

The goal of Step 2 is to find the activity network that maximises the reproducibility of the final graph. This is done by solving the following problem.

Input data

- $\boldsymbol{n}$ is the number of event types present in the log.
- $\boldsymbol{E}$ is the number of event types that the user desires to be mapped: it can be lower or equal to the number of event types in the event log ($E \leq n$).
- $\mathbf{M} = \{m_i\}$ is a vector, where each of the n elements m_i shows the occurrences related to the i-th activity in the event log.
- $\mathbf{A} = \{a_{ij}\}$ is a $n \times n$ matrix where each of its elements a_{ij} represents the frequency of the connection between activities i and j as found in the event log (i.e. how many times the arc $i - j$ is visited).
- $\mathbf{C} = \{c_{ij}\}$ is the correlations matrix. It is a boolean matrix such that:

$$c_{ij} = \begin{cases} 1 & \text{if} \quad a_{ij} > 0 \\ 0 & otherwise \end{cases} \quad \forall i, j$$

- $\boldsymbol{k_{max}}$ is the maximum number of iterations allowed, hence the computational budget for the optimization process.

Decision variables

- $\boldsymbol{\beta} = \{\beta_i\}$ is a boolean vector such that $\beta_i = 1$ if the i-th activity is considered for the inclusion in the network, $\beta_i = 0$ otherwise; hence $\boldsymbol{\beta}$ represents the list of activities that are used in the network: notice that the length of the list will be constrained to be equal to the number provided by the user.
- $\boldsymbol{\Gamma} = \{\gamma_{ij}\}$ is a symmetric, boolean matrix representing an activity network and its elements γ_{ij} are 1 if the event type i is followed by the event type j.

Station s	Entrance	Exit
0	999	–
1	998	997
2	996	995
3	994	993

TABLE I: Illustrative Test Case – System activity codes. Station $s = 0$ corresponds to the system entrance.

Optimization problem (MSM-1)

$$\max \quad \sum_{i=1}^{n} \beta_i \, m_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \, \gamma_{ij} \tag{1}$$

$$s.t. \quad \sum_{i=1}^{n} \beta_i \;\; = \;\; E \tag{2}$$

$$\gamma_{ij} = \min\Big(\beta_i \, c_{ij};\, \beta_j \, c_{ij}\Big) \qquad \forall i = 1 \ldots n, \quad \forall j = 1 \ldots n, \tag{3}$$

$$\beta_i \;\in \{0, 1\} \qquad \forall i = 1 \ldots n, \tag{4}$$

$$\gamma_{ij} \in \{0, 1\} \qquad \forall i = 1 \ldots n, \quad \forall j = 1 \ldots n. \tag{5}$$

The objective function (1) is composed by two terms: the first term represents the frequency of occurrences of the activities in the network, the second one takes into account the frequency of the connections between activities. Here, we assume equal weights for the two terms. Constraint (2) is focusing the number of considered events to the user-specified value E. Constraints (3) are logical constraints needed to define the variables γ_{ij}. Constraints (4) and (5) state the nature of the decision variables.

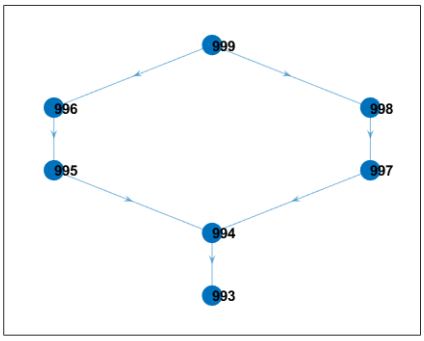
The *MSM-1* problem is solved by a *local search* procedure. Specifically, in each iteration k, the performance of just one neighbor sequence is evaluated, that is a sequence differing from the starting one only for one single activity [24]. The solution $\mathbf{\Gamma}^* = \{\gamma_{ij}^*\}$ identifies the final activity network.

IV. EXPERIMENTS

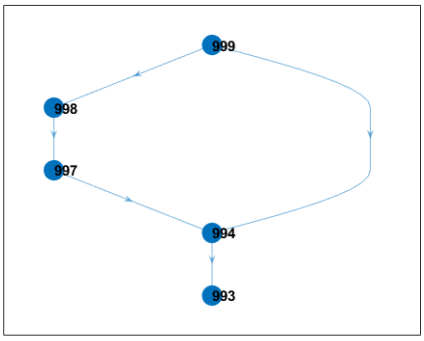
We have tested the mining approach presented in section III-A and solved the MSM-1 problem for (A) an illustrative test case on the system of Figure 1a, and (B) a test case on the LEGO® Manufacturing System model of the system shown in Figure 1c. Following we present the results obtained.

A. Illustrative Test Case – Sample System (Figure 1a)

The event log is composed by seven activities ($n = 7$) which have been coded according to Table I: Figures 3a and 3b show the results obtained in case a different number of input events E is imposed. With $E = 5$, two activities related to the station that is less visited – 40% of the entities – have been removed. This is reasonable also if we consider the simplicity of the system. As a result, the reproducibility of 60% of the entities has been maximized.



(a)



(b)

Fig. 3: Illustrative Test Case – Activity networks obtained in the cases: (a) with $E = 7$, and (b) with $E = 5$.

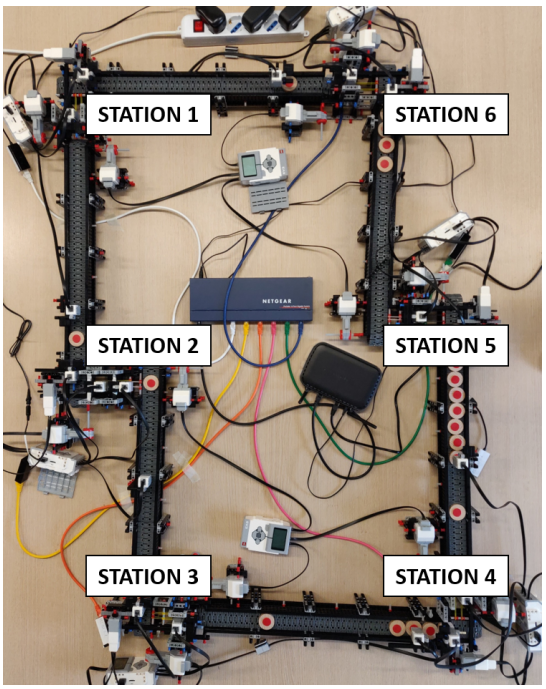


Fig. 4: Test Case – LEGO® Manufacturing System (LMS) model.

B. Test Case – LEGO® Manufacturing System (Figure 1c)

We applied the proposed method for discovering the configuration of a LEGO® Manufacturing System (LMS) that is installed in the Manufacturing Systems Laboratory from the Department of Mechanical Engineering of Politecnico di Milano [25]. The physical system is composed by six stations $s \in \{1, 2, 3, 4, 5, 6\}$ with intermediate conveyors that operate also as buffers (Figure 4). All the stations are controlled by LEGO® EV3® bricks, each programmed through python scripts (EV3DEV OS [26]), and connected to a local network. Conveyors are controlled through proprietary LEGO® software that allows switch on/off and speed setting. Wooden circles tagged with red plates represent pallets that load the workpieces which must be processed by all the stations (single-product line).

Each station has three sensors. The first sensor is placed on the upstream buffer: if it detects that any part is available, the latter is let into the workstation. The second sensor is positioned inside the workstation and identifies if a pallet has entered the station. Buffer capacities are defined by the position of the third sensor on the downstream conveyor. We call B_s the buffer after station s. When the third sensor of the s-th station detects that B_s is full, station s is set to blocking state and does not release pallets in the downstream conveyor. The total buffer capacity is limited and the blocking after service rule is applied. Despite the LMS is a closed-loop system, in this work it is modeled as open and it is assumed that a large number of unprocessed workpieces are waiting outside the system, and that each part arriving at station 1 is a new arrival. The processing times are represented by a time $T_W(s)$ that each workpiece has to wait in the s-th station before being released. Stations $\{1, 3, 4\}$ represent manual operations, therefore their processing times are modeled as stochastic. Stations $\{2, 5, 6\}$ represent automatic workstations and their times are modeled as deterministic. Station 2 involves two sequential operations with deterministic processing time and no buffer slots in between.

In accordance to the literature [27], we have first framed the event log by positioning two additional sensors in each station, one at the entrance and one at the exit of the station, and collecting the transit times of pallets along the system. The generated log is composed by three data labels: (1) a sequential number which is the ID of parts, (2) the sensor ID, which is uniquely referring to its location on the line, and (3) the date and time of the detection of the parts passage by the sensor. The event log activities have been coded in accordance to Table II. We acquired 3 different and independent traces, thus 3 different event logs were produced. Each log records the events encountered by 40 pieces along the LMS.

Figure 5 describes the obtained PN model that has been validated by creating a simulation model of the line in Arena® using the capacities and the processing time distributions obtained. For each of the three event logs we have generated a simulation model with the procedure as in section III-A and compared the performance values of the real system with the ones obtained by simulation experiments (each simulation has been replicated 100 times). Tables III and IV list the results obtained and the comparison between the simulation model and the LMS in terms of the distributions of processing times and buffer capacities. Results confirm that the generated PN is correctly representing the system layout and the cycle times, hence the developed method is effective for the simulation model construction phase. From Table III it can be noticed that the simulation model created from PN_{adj} is over-estimating the System Time of the LMS model, while the Work-in-progress average levels are underestimated. However, we have to take into account that this is due to the lower buffer capacities that have been mined (Table IV).

Station s	Entrance	Exit
1	999	998
2	997	996
3	995	994
4	993	992
5	991	990
6	989	988

TABLE II: Test Case – LMS system activity codes.

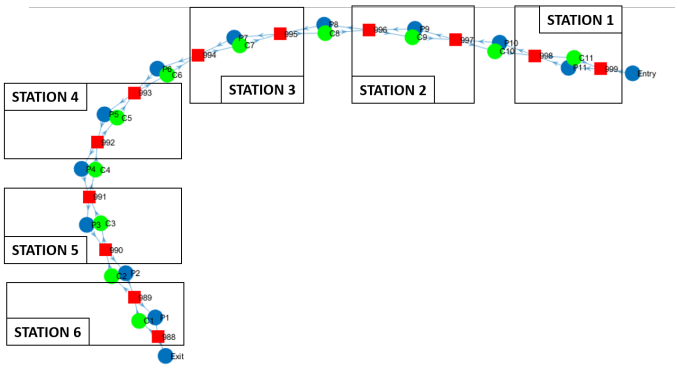


Fig. 5: Test Case – Final PN_{adj} representation and corresponding stations of the LMS.

V. CONCLUSIONS

In this work, we proposed a Manufacturing Systems Mining (MSM) approach that is effective for discovering a production system configuration starting from an event log composed by only three data-type labels. The identified system time and work-in-progress are still not comparable with the real system (Table III). Indeed, if data sampling is performed with an non-saturated line, the buffer capacities of the generated simulation model will be underestimated (Table IV) and eventually the performances estimated by the simulation model will be lower than the real ones. Future research shall aim to overcome this issues, for instance by increasing the amount of input information to allow for a further improvement in the accuracy of the generated simulation models.

Property	System	Mean	St. Dev.	SE Mean	95% CI for Difference	P-Value	Result
Layout	LMS	-	-	-	-	-	- *reference* -
	PN_{adj}	-	-	-	-	-	Correct
Cycle Time [s]	LMS	10.92	4.74	0.31	[-0.56, 0.67]	0.87	- *reference* -
	PN_{adj}	10.86	0.40	0.03			Correct
System Time [s]	LMS	182.8	68.2	4.5	[-207.54, -189.63]	0.00	- *reference* -
	PN_{adj}	381.4	12.4	0.88			Over-estimated
Work-in-Progress [parts]	LMS	16.67	6.67	0.31	[7.69, 8.93]	0.00	- *reference* -
	PN_{adj}	8.35	0.99	0.07			Under-estimated

TABLE III: Test Case – Results comparison between system performances of the LMS and the generated simulation model.

Station s	T_w (PN_{adj})	Buffer B_s	LMS	PN_{adj}
1	$\sim N(9.6, 1.6)$	B_1	5	5
2	$\sim N(10.3, 0.9)$	B_2	9	2
3	$\sim N(9.4, 1.8)$	B_3	3	2
4	$\sim N(8.6, 1.8)$	B_4	9	4
5	$\sim N(10.3, 1.8)$	B_5	3	2
6	$\sim N(8.5, 1.6)$			

TABLE IV: Test Case – Processing times and buffer capacities found through the proposed procedure.

REFERENCES

[1] Y. Rao, F. He, X. Shao, and C. Zhang, "On-line simulation for shop floor control in manufacturing execution system," in *International Conference on Intelligent Robotics and Applications*, pp. 141–150, Springer, 2008.

[2] C.-C. Chen, C.-L. Chen, C.-Y. Ciou, and J.-X. Liu, "Communication scheduling scheme based on big-data regression analysis and genetic algorithm for cyber-physical factory automation," in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics, October 9th-12th, Budapest, Hungary*, pp. 2603–2608, IEEE, 2016.

[3] J. V. M. Grieves, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems(excerpt)," August 2016.

[4] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017.

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

[6] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, *et al.*, "Process mining manifesto," in *International Conference on Business Process Management*, pp. 169–194, Springer, 2011.

[7] W. V. der Aalst, *Process Mining - Data Science in Action*. Springer, second edition ed., 2016.

[8] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro, "Process mining in healthcare: A literature review," *Journal of biomedical informatics*, vol. 61, pp. 224–236, 2016.

[9] M. Jans, J. M. Van Der Werf, N. Lybaert, and K. Vanhoof, "A business process mining application for internal transaction fraud mitigation," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13351–13359, 2011.

[10] A. D. Bautista, L. Wangikar, and S. M. K. Akbar, "Process mining-driven optimization of a consumer loan approvals process," *BPI Challenge*, 2012.

[11] A. A. d. M. A.J.M.M. Weijters, W.M.P. Van der Aalst, "Process mining with the heuristic miner algorithm," *BETA publicatie : working papers*, vol. 166, 2006.

[12] A. W. J.E. Cook, "Automatic process discovery through event-data analysis," *International Conference in Software Engineering*, 1995.

[13] E. Gupta, "Process mining algorithms," *International Journal of Advance Research In Science And Engineering*, vol. 3, November 2014.

[14] L. P. D. S. G. Greco, A. Guzzo, "Mining expressive process models by clustering workflow traces," *Lecture Notes in Computer Science*, 2004.

[15] L. P. D. S. G. Greco, A. Guzzo, "Discovering expressive process models by clustering log traces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, 2006.

[16] L. P. G. Greco, A. Guzzo, "Mining hierarchies of models: From abstract views to concrete specifications," *Lecture Notes in Computer Science*, 2005.

[17] C. W. Günther and W. M. Van Der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," in *International conference on business process management*, pp. 328–343, Springer, 2007.

[18] W. M. van der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. De Medeiros, M. Song, and H. Verbeek, "Business process mining: An industrial application," *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[19] A. Gouveia, "Process mining of enterprise applications based on outsystems agile platform," *Instituto Superior Tecnico Lisboa*, 2010.

[20] B. F. van Dongen and A. Adriansyah, "Process mining: fuzzy clustering and performance visualization," in *International Conference on Business Process Management*, pp. 158–169, Springer, 2009.

[21] M. Mesabbah and S. McKeever, "Presenting a hybrid processing mining framework for automated simulation model generation," in *Winter Simulation Conference*, pp. 1370–1381, IEEE, 2018.

[22] J. Peterson, "Petri nets," *Computing Surveys*, vol. 9, September 1997.

[23] D. Tarasconi and M. Zanotti, "Process mining for manufacturing systems discovery," *M.Sc. Thesis*, 2018.

[24] M. Prodel, "Modelisation automatique et simulation de parcours de soins a partir de bases de donnees de sante," *Ph.D. Thesis*, 2017.

[25] G. Lugaresi, N. Frigerio, M. Zhang, Z. Lin, and A. Matta, "Active learning experience in simulation class using a LEGO®-based manufacturing system," in *2019 Winter Simulation Conference (WSC)*, IEEE, 2019.

[26] ev3dev-Website, "https://www.ev3dev.org/," Accessed: 2019-07-23.

[27] W. S. E. Kindler, V. Rubin, "Process mining and petri nets synthesis," *Lecture Notes in Computer Science*, September 2006.