Discovery and digital model generation for manufacturing systems with assembly operations

Giovanni Lugaresi, Andrea Matta

Discovery and digital model generation for manufacturing systems with assembly operations

Giovanni Lugaresi[1] and Andrea Matta[1]

Abstract— **Industry 4.0 determined the emergence of technologies which allow for data-based production planning and control approaches. Digital twins can be used to take decisions based on the current system state. Hence, their performance strictly depends on the capability to correctly represent their physical counterparts at any time. The development of digital twins for manufacturing systems can be significantly accelerated by automated model generation techniques. However, production systems including assembly stations suffer from event records with multiple part identifiers, resulting in the wrong finding of the system structure. In this paper, we define the problem of the proper discovery of assembly operations. Then, we describe an algorithm to generate a complete digital model exploiting the new concept of object-centric process mining. In a case study, a flow shop including assembly stations is successfully discovered, allowing for the automated building of a simulation model with the proper logical behavior.**

I. INTRODUCTION

Recently, manufacturing environments have become significantly complex in the attempt to satisfy a high demand for customized products, while aggressive digitization efforts have sponsored production enterprises to invest in new technologies toward higher levels of automation [1]. In modern organizations, information systems are playing a substantial role in the support of day-to-day operations. Within the new industrial context, such systems must evolve toward satisfying new and more demanding requirements. Manufacturing Execution Systems need to be extended toward advanced production planning and control, to allow for the efficient online management of shop floors. Modern decision support tools are based on the coexistence of the real system with its digital counterpart, often called digital twin [2]. In the planning and control phases, digital twins can be represented by discrete event simulation models. The addition of real-time streams of data can help production planners to evaluate solutions that are optimal for the current system state [3]. However, if the model is not a accurate representation of the current system state and configuration, simulation experiments will likely be biased and lead to erroneous decisions [4].

Meanwhile, Industry 4.0 contributed also to the rise of new technologies for data handling. Such devices can provide information about the shop floor status almost anytime [5]. The consequent increased integration of functionalities allows a closer coupling of previously separated decisional levels [6]. Indeed, quicker data exchange capabilities allow for a tighter decisional loop, which can be based on current streams of data from the shop floor, resulting in better online decisions [7]. The availability of real-time data also hints that if a model could be generated from the field data in the manufacturing system, the development phase may be significantly shortened, enabling digital twins to be automatically aligned with their real counterparts [8].

Recent approaches use process mining techniques for model generation [9], [10]. The exploitation of process mining allows for an effective development of simulation models for manufacturing systems such as flow lines [4]. However, the automated development of digital twins for more complex manufacturing systems reaches the limitations of the available methodologies. For instance, assembly processes are very common in the production enterprises (e.g., automotive industry). In such environments, several material flows converge in assembly stations. During or after the assembly of parts, it is common that new part IDs are assigned to the just introduced assembled part. Traditional process mining techniques assume a single case identifier to derive the logical flows (e.g., activity precedences). As a consequence, the production flows of components and assemblies can be discovered only as separate models, and the assembly logic is effectively lost in translation.

In this paper, we introduce a method to infer assembly operations in manufacturing systems, aimed at the development of an appropriate discrete event simulation model. The rest of the paper is organized as follows. Section II summarizes the state of the art for automated simulation model generation in manufacturing systems. Section III outlines a graph-based model mining procedure which is taken as reference in this work and the problem of discovering assembly operations, while section IV describes the proposed method to generate complete digital models. Section V presents a test case together with numerical results. Our final remarks are in Section VI.

II. STATE OF THE ART

This section lists the significant contributions for generating simulation models of manufacturing systems, with a focus on assembly processes.

A. Generation of Simulation Models for Manufacturing Applications

Process mining is a collection of approaches, techniques, and tools, which provides a wide range of knowledge about the processes inside the organizations based on available datasets. It is mainly composed by three activities [9]: (1)

*This work was not supported by any organization

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

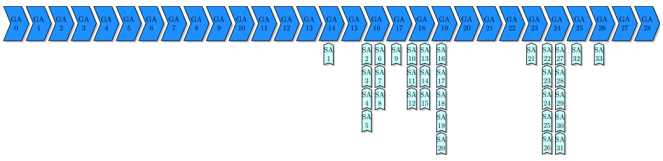

Fig. 1: Example of manufacturing system logical flow including assembly operations (from [21]).

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 analyzed system model. Process mining allows not only to estimate parameters and causal relationships, but also logical connections such as precedences among activities. Thus, it may be used to infer the topological structure of a manufacturing system and support the generation of simulation models [11].

Several contributions in the literature exploited process mining to develop forward-looking models (i.e., simulation) for manufacturing applications. Rozinat [11] firstly proposed to exploit process mining for developing models able to predict the future behavior with respect to the performance of a system. Bergmann et al. [12] introduced a methodology for recognizing the production policies applied on a manufacturing system. Milde and Reinhart [13] developed an approach for discovering the material flow, estimating parameters, and identifying the control policies from manufacturing event logs. Lugaresi and Matta [4] developed a method for generating simulation models with a proper level of detail. In other applications, process mining has been used to retrieve specific parameters of a manufacturing system such as interarrival times [14], determinants of process delays [15], resource availabilities [16], batching operations [17], and production system structure [18].

B. Discovery of Assembly Processes

Few contributions in the literature explicitly mention the automated model generation of assembly operations. Denno et al. [18] mined an automotive under-body assembly system with the goal to optimize its production schedule. Rashid and Louis [19] presented a framework that utilizes RFID tracking and process mining techniques to automatically generate the model of an assembly line. The goal is to detect any deviation in the performed process from the predefined plan. Knoll et al. [20] developed a methodology to apply process mining to internal logistics for a mixed-model assembly line. The authors used multi-dimensional process mining to automatize and improve the Value Stream Mapping methodology. The goal is to allow for a precise process discovery and classification, including performance analysis to find the parts of the plant where the most waste is produced.

The aforementioned papers address mining assembly operations. Yet, none of them focus on simulation model generation. Despite being very common in manufacturing environments, the discovery of assembly operations has a scarce representation in the literature. This may be due to the problem of flattening data in event logs [22].

Event logs are files containing information about parts flowing in the system (e.g., serial codes associated to the parts, activity time stamps). We may assume that all data gathered from the manufacturing system can be aggregated and collected in an event log. In general, the event log may contain several types of information, such as part flows, resource identifiers, and quality check outcomes. Traditional process mining techniques assume an event log with three main information types: (1) the activity identifier $n \in \mathbb{N}$, (2) the work-piece identifier $i \in \mathbb{I}$, and (3) the timestamps $t_S(n,i)$ and $t_F(n,i)$ indicating the moment in which the n-th activity has been started and finished by the i-th work-piece, respectively. Table I shows an example of event log.

In manufacturing systems with assembly operations, it is common to observe different material flows converging toward assembly stations (e.g., Fig. 1). In such points, the work-piece identifiers are likely to change from a production stage to the following one. For instance, a car frame is usually assigned an identifier, while sub-components such as doors have other dedicated IDs. Once assembled, the work-in-progress part may have either a new identifier or keep one of the sub-components ID. Either way, the following events in the production system refer to the assembled product, which is linked to multiple sub-components IDs. Hence, when aggregating events data, there are multiple flattening choices possible which lead to different views that are disconnected. The result is that the overview of the system structure is quickly lost as event data need to be extracted multiple times for the different views.

A further limitation is evident in the generated model, where the flattening choice may force a wrong representation of the system behavior. Indeed, the simulation model must consider the assembly phase to correctly model the behavior of the system. Namely, the availability of all needed material upstream is a blocking condition on the assembly points. Neglecting this condition in a model generation technique may result in a model overestimating the performances of the real system. For example, Fig. 2 graphically shows two different modeling options using a Petri Net. Transition 3 represents the assembly operation, while transitions 1 and 2 are the last operations done by the sub-components. In the first case, transition 3 may fire even if only one of the upstream operations has been completed. Differently, in the second case, the assembly transition is enabled by the availability of sub-components material.

C. Object-centric Process Mining

Van der Aalst [22] discussed the gap between real event data and flattened event logs exploited by traditional process mining techniques. The author proposes a new mining paradigm called Object-centric Process Mining (OCPM), together with a specific log format. The object-centric log is a collection of events, where each event is be related to objects of different types. Moreover, basic notations and a baseline

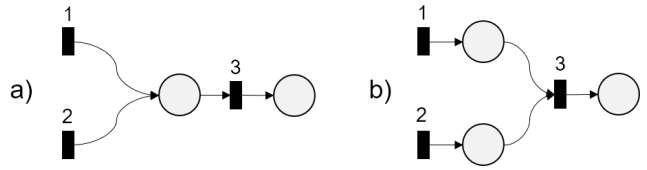

Fig. 2: Example of assembly process modeled by Petri Nets: a) non-blocking condition (improper modeling), b) blocking condition (assembly properly modeled).

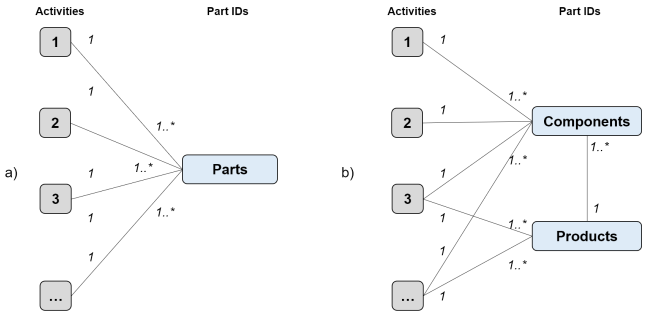

Fig. 3: Cardinality diagrams applied to manufacturing processes, relationships between part identifiers and activities: a) traditional process mining approaches, b) object-centric process mining.

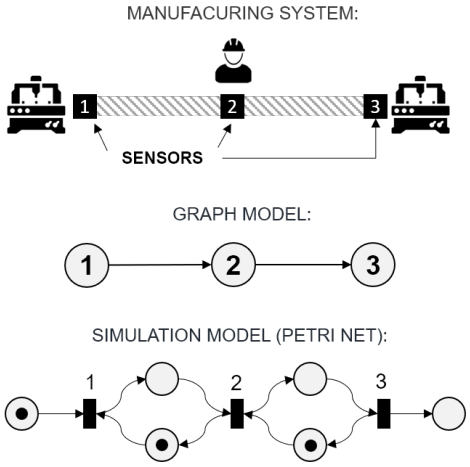

Fig. 4: Graphical overview of a graph-based model generation procedure.

discovery approach are presented to facilitate discussion and understanding.

OCPM can be applied to better describe production systems with assembly operations. Fig. 3 graphically explains the difference among two mining views. In traditional mining approaches, only one part ID notion is allowed. Hence, for assembly processes in which material flows converge, information is effectively lost. Object-centric process mining allows for using several notions of part identifiers, thereby representing objects that may refer to multiple items (e.g., a component, an assembled product). Such representation is feasible in a manufacturing environment. The only assumption is the availability of a Bill of Material (BOM), which is necessary to define the object relationships. Such records are widely used in production systems and commonly available within the Product Life-cycle Management or the Enterprise Resource Planning tools.

Nevertheless, applications of OCPM for assisting model generation in manufacturing systems are missing in the literature. This paper addresses this gap by identifying the scope of work and proposing the development directions for this research.

III. PROBLEM DESCRIPTION

In this section, we outline a graph-based mining approach that is used to generate a simulation model starting from the data of a real manufacturing system. Then, we describe the problem of considering assembly processes in the model generation procedure.

A. Model Generation

Let us assume the manufacturing system is equipped with data collection system, and events in the system are aggregated in an event log. The log can then be used to discover the manufacturing system logical structure and its parameters. Model generation is the procedure which links the system data contained in the event log with a digital model. In the most basic form, a digital model can be rendered by a set of nodes representing the activities, and a set of arcs which correspond to the material flow relationships among activities (i.e., precedences). Let us then define a graph model as a tuple $(\mathbb{N}, \mathbb{A})$ where $\mathbb{N}$ is the set of nodes and $\mathbb{A} = \mathbb{N} \times \mathbb{N}$ is the set of arcs in the model. For instance, the graph model obtained for the system in Fig. 4 is defined by the set of nodes $\mathbb{N} = \{1, 2, 3\}$ and the set of arcs $\mathbb{A} = \{(1, 2), (2, 3)\}$. A model is generated by representing all the relational properties (i.e., precedence relationships) in an activity network, namely a directed graph in which nodes correspond to the activities and arcs express the material flow relationships between activities. A unique set of activities $\mathbb{N}$ 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 series of activity identifiers. Hence, each i-th part has a corresponding trace $\theta_i = \{n^{(1)}, n^{(2)}, \ldots, n^{(N_i)}\}$, where N_i is the number of the activities performed by the i-th part. For example, with reference to the event log of Table I, $\theta_1 = \{1, 2\}$. 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 (n, m) exists if $\exists i \in I | t_F(n, i) < t_S(m, i)$.

Fig. 4 shows the main results of a model generation procedure. The manufacturing system is composed by three main activities, each recording the events in an event log. The graph model is the result of mining the direct-follow relationships from the traces. Namely, the graph model is such since at least one trace is $\theta = \{1, 2, 3\}$. Next, the graph model can be enriched with properties of nodes and arcs. For instance, the finite capacity of the conveyor between activities 1 and 2 may be expressed as a property of the arc $(1, 2)$. Once a graph model is created, it can be converted into a simulation model through formalisms such as Petri Nets or

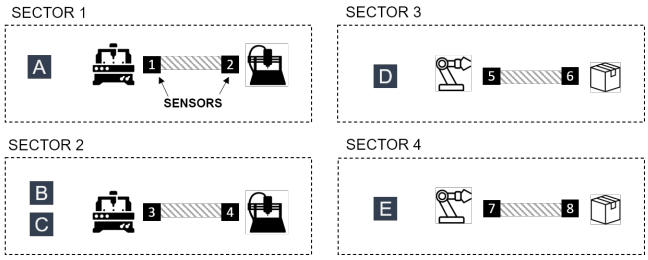

Fig. 5: Logical schema of the flow shop manufacturing system analyzed in this work.

Event Relationship Graphs. The properties of nodes and arcs may be exploited by the model generation procedures and for conversion to other formalisms. For instance, in the Petri Net model of Fig. 4, the finite capacity between transitions 1 and 2 is represented by a re-entrant flow through a finite-capacity place. Further details on the model generation steps are available in related works [4].

TABLE I: Extract from the event log of the manufacturing system in Fig. 4.

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
2020-11-23 16:37:58	3	1	finish

B. Generation of Complete Models

Let us consider the flow shop depicted in Fig. 5. The manufacturing system consists in four sectors. Each sector produces specific part types. Sectors 1 and 2 include machining operations that produce components of type A, B, and C, while sectors 3 and 4 assemble the components into final products of type D and E, respectively. Several items of part types D and E may be produced, and each item has a unique identifier (e.g., data-matrix quick response code).

The model generation procedure listed in section 3.2 produces the graph model shown in Fig. 6a. From the figure, it can be noticed that the result is a model with sectors treated as separate from one another. This is because model generation is strictly based on the single part identifier hypothesis. Since assembled part have different identifiers from components, assembly relationships are neglected.

In order to generate a more realistic model logic, OCPM can be used to enrich the description of the manufacturing system. To identify objects relationships, the Bill of Material (BOM) of the assembled products can be exploited. The BOM includes the component-product relationships among part types. For instance, the BOM of part type C lists part types A and B (Table III). BOMs are widely available in manufacturing environments, hence they can be exploited to generate object-centric event logs.

The first step to allow for OCPM is to develop an object-centric event log. Table II shows an extract of the object-centric log in which one component of type A and B are assembled into a part type C. The components are coded A001 and B001, respectively, while the assembled part is coded with D001. In the new log representation, the part identifier column has been substituted by two object columns: (1) components, and (2) assembled products. Notice that the number of object columns is directly linked to the levels of the BOM.

TABLE II: Object-centric event log for the assembly of product nr. D001.

		activity		objects involved	
event	time-stamp	*name*	*type*	*components*	*assembly*
1	0.00	1	start	A001	-
2	0.00	3	start	B001	-
3	0.35	1	finish	A001	-
4	0.35	2	start	A001	-
5	0.45	3	finish	B001	-
6	0.45	4	start	B001	-
7	0.51	4	finish	B001	-
8	0.62	2	finish	A001	-
9	0.62	5	start	{A001,B001}	D001
10	0.76	5	finish	{A001,B001}	D001
11	0.76	6	start	{A001,B001}	D001
12	0.88	6	finish	{A001,B001}	D001

The availability of an object centric log allows for enhancing the simulation generation procedure, considering assembly operations. Let us assume that a model generation procedure as the one described in [4] has been done on the traditional available log. After the system discovery step has been completed, the graph model must be corrected to account for assembly operations. In the next section, we propose a procedure to solve the graph completion problem.

IV. PROPOSED APPROACH

Let us define $\mathbb{P}_n, \mathbb{S}_n$ as the sets of predecessors and successors of node n, respectively. We assume the traces are perfectly reliable, hence that data have been properly collected and no outliers remain in the event log. Furthermore, we assume the BOMs of all the products that flow in the system are available. Define $\pi(i)$ as the part type of item i (for instance, $\pi(1) = A$). Further, given a generic part type p, the function $BOM(p, \#)$ returns the set of parts types which compose the $\#$-th level of the BOM of part type p. For example, $BOM(D, 1) = \{A, B\}$.

Algorithm 1 lists the pseudo-code of the proposed procedure. The algorithm is based on the hypothesis that – after a traditional mining step – nodes with no predecessors are the starting points of their respective graph-models. Consequently, they are candidate assembly points. To find such nodes, traces are explored, searching for component parts. The assembly points are then connected to the last explored nodes by the sub-components.

In summary, a model generation procedure including the proposed algorithm to consider assembly points can be described by the following steps:

- Step 1. Data collection from the real system and event log preparation: data is organized in event logs recording the production steps followed by each part in the

Algorithm 1: Proposed algorithm for mining assembly processes (pseudo-code)

Step 1: List all nodes with no predecessors: $\mathbb{N}' = \{n \in \mathbb{N}\} | \mathbb{P}_n = \emptyset$;
for $n \in \mathbb{N}$ **do**
  Step 2: List parts such that node n is in the trace: $i \in \mathbb{I}' | n \in \mathbb{N}' \& n \in \theta_i$;
  if $BOM(\pi(i), 1) \neq \emptyset$ **then**
    Step 3: Find components parts: $j \in \mathbb{I} | \pi(j) \in BOM(\pi(i), 1)$;
    Step 4: Find last nodes in the trace of component parts: $\mathbb{N}'' = \{n'' \in \theta_j\} | \mathbb{S}_n = \emptyset$;
    Step 5: Add the nodes to the predecessors of node n: $\mathbb{S}_n \leftarrow n''$; mark node n as assembly point;

system and the corresponding time stamps.

- Step 2a. System discovery. The mining algorithm considers both the logical relationships among the events and the occurrences of arcs and nodes. The result is a graph model [4].
- Step 2b. Proposed addition for graph model completion: assembly operations search. Apply Algorithm 1 exploiting BOMs to search and mark assembly points.
- Step 3. Statistical analysis. Processing and waiting times are retrieved from the event log, hence also statistical distributions describing the process can be obtained.
- Step 4. Model conversion. At this step, the digital model may be converted to another modeling framework, for instance to a Petri Net.
- Step 5. Model validation and usage. The model can be used for estimating the manufacturing system performance.

V. TEST CASE: ASSEMBLY FLOW SHOP

In this section, we present a test case of the algorithm presented in Section 4. The manufacturing system is the flow shop represented in Fig. 5. Two types of assembly products are present: D and E. Their BOM is listed in Table III. Without loss of generality, we consider the four sectors composed by stations with balanced operations: namely, each station is dedicated to operations that require a processing time $p_a \sim Exp(1) \, \forall a \in A$. Stations 5 and 7 are assembly stations, hence they may not start working until at least one part has finished at stations 2 and 4, respectively. It is also assumed that no space constraints are limiting stations 2 and 4 from downloading parts when finished.

Five event logs have been generated with Rockwell ARENA™, representing the production of 1000 parts. Each event log has been used within a model generation procedure as described in [4]. Next, the algorithm described in Section 5 has been applied to the generated graph. Fig. 6b shows the result of the algorithm application. It can be noticed that, differently from the traditional mining results, nodes 5 and 7 are now connected to their predecessors in material flow. Further, nodes 5 and 7 have been enriched with a property that identifies them as assembly stations. Such result has been obtained for all the event logs. The results confirm that the additional algorithm can be used to enrich simulation model generation procedures for assembly production systems. Indeed, the addition of the assembly property can be exploited to generate a proper representation by the simulation model (i.e., Fig. 2b).

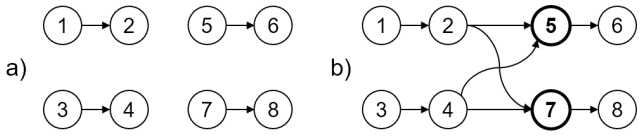

Fig. 6: Test Case – Difference among graph-models mined from the event logs of the flow shop of Fig. 5: a) traditional model generation procedure; b) procedure with Step 2b: assembly-oriented algorithm (bold circles represent assembly stations).

VI. FINAL REMARKS

This work has introduced the problem of considering assembly processes within an automated model generation procedure. Next, we report our final considerations.

A. Improvement Areas

Several limitations still must be solved. The following improvements can be considered for the proposed approach:

- The proposed algorithm assumes perfect traces and the complete availability of BOM data. Realistic datasets are more unreliable, and proper adjustments may have to be done.
- In real systems, several components may be used by different assembly products. In such cases, the proposed algorithm shall be extended to include multiple combinations of assembly.
- The complete spectrum of systems that can be discovered with the proposed technique should be identified.
- Real systems are much more complex than the studied use case, with more assembly operations which might take place on different stations along the same part type flow. In such a case, multiple assembly locations have to be identified for each part type.
- Techniques to validate the proposed approach shall be developed. Indeed, different methods may provide similar results among several options, or vice-versa. Specific indicators could be developed to assist the benchmarking of methods.

B. Conclusions

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

TABLE III: Test Case – Bill of material of part types D and E.

Part Type	Components
D	$\{A, B\}$
E	$\{A, C\}$

manufacturing systems. In this work, we have introduced an algorithm that allows for the discovery and modeling of assembly processes. The algorithm can be exploited in a simulation model generation procedure, since it identifies nodes in which components are assembled into final or work-in-progress products. With this addition, the blocking condition related to the availability of parts can be added to a simulation model and allow for the proper estimation of system performances. Notice that the developed technique can be also used to investigate disassembly and de-manufacturing operations, since the material flow dynamics are comparable to assembly processes, which suffer from the same problem of dimensionality in the part identifiers. In the future, the improvement areas of this work shall be explored, together with applications to real instances.

REFERENCES

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

[2] E. Negri, S. Berardi, L. Fumagalli, M. Macchi, Mes-integrated digital twin frameworks, Journal of Manufacturing Systems 56 (2020) 58–71.

[3] S. Tavakoli, A. Mousavi, A. Komashie, A generic framework for real-time discrete event simulation (des) modelling, in: 2008 Winter Simulation Conference, IEEE, 2008, pp. 1931–1938.

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

[5] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, Journal of Manufacturing Systems 48 (2018) 157–169.

[6] D. Rossit, F. Tohmé, Scheduling research contributions to smart manufacturing, Manufacturing Letters 15 (2018) 111–114.

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

[8] H. Reinhardt, M. Weber, M. Putz, A survey on automatic model generation for material flow simulation in discrete manufacturing, Procedia CIRP 81 (2019) 121–126.

[9] W. Van Der Aalst, Data science in action, in: Process mining, Springer, 2016, pp. 3–23.

[10] W. M. van der Aalst, Process mining and simulation: a match made in heaven!, in: SummerSim, 2018, pp. 4–1.

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

[12] S. Bergmann, N. Feldkamp, S. Strassburger, Approximation of dispatching rules for manufacturing simulation using data mining methods, in: 2015 Winter Simulation Conference (WSC), IEEE, 2015, pp. 2329–2340.

[13] M. Milde, G. Reinhart, Automated model development and parametrization of material flow simulations, in: 2019 Winter Simulation Conference (WSC), IEEE, 2019, pp. 2166–2177.

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

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

[16] N. Martin, F. Bax, B. Depaire, A. Caris, Retrieving resource availability insights from event logs, in: 2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2016, pp. 1–10.

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

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

[19] K. M. Rashid, J. Louis, Process discovery and conformance checking in modular construction using rfid and process mining, in: Construction Research Congress 2020: Computer Applications, American Society of Civil Engineers Reston, VA, 2020, pp. 640–648.

[20] D. Knoll, G. Reinhart, M. Prüglmeier, Enabling value stream mapping for internal logistics using multidimensional process mining, Expert Systems with Applications 124 (2019) 130–142.

[21] M. S. Uysal, S. J. van Zelst, T. Brockhoff, A. Farhang, M. P. Ghahfarokhi, R. Schumacher, S. Junglas, G. Schuh, W. M. van der Aalst, Process mining for production processes in the automotive industry.

[22] W. M. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: International Conference on Software Engineering and Formal Methods, Springer, 2019, pp. 3–25.