Automated generation of digital models for manufacturing systems: The event-centric process mining approach

Computers & Industrial Engineering 197 (2024) 110596

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Automated generation of digital models for manufacturing systems: The event-centric process mining approach

Claudio Castiglione

Politecnico di Torino, Department of Management and Production Engineering, Corso Duca degli Abruzzi, 24, Torino 10129, Italy

ARTICLE INFO

Keywords:
Process mining
Discrete event simulation
Petri net
Event relationship graph
Manufacturing system
Industry 4.0

ABSTRACT

Digital and simulation models support the design and management of complex systems. However, system modelling is a time-demanding and knowledge-intensive activity. Moreover, modern manufacturing systems are subjected to frequent changes in production plans and subsequent reconfigurations. Therefore, the quick regeneration of the digital models is necessary to align digital twins and cyber-physical systems. This paper proposes a novel event-centric process mining paradigm, a process discovery algorithm, and a set of Key Performance Indicators for the fast and automated generation of digital models and their benchmarking. The discovery algorithm is based on the Event Relationship Graph of the conceptual model of the physical line. The algorithm is tested in four realistic systems of increasing complexity to verify the accuracy in modelling multi-product systems with re-entrant flows and random reworks in the presence of the assembly, disassembly, and split processes beyond the processing operations, and multi-operation workstations. The Event Relationship Graphs of the four systems are presented through the equivalent Petri nets models. The proposed approach is suitable for systems where the sensor positions are known and meaningful, like manufacturing systems, and it is effective for the quick automated generation of digital models for the activities of production planning and control as it requires a few seconds of computation time and a few hours of system observation.

1. Introduction

Digital and simulation models are important in the manufacturing field. They can support the design and the management of complex systems by investigating the impact of design decisions, analysing the effect of stochastic processes, and testing and developing management strategies even to react to disruptive events in the production phase (Atzori et al., 2010). Moreover, Industry 4.0 (I4.0) technologies improve the effectiveness of digital models and foster their adoption. For example, Internet of Things (IoT) technologies provide a large volume of data to increase digital model accuracy (Lu et al., 2019). At the same time, cloud technologies make economically available advanced computational resources, enabling the concept of simulation-as-a-service to foster the diffusion of simulation techniques (Shekhar et al., 2016) even among small and medium enterprises, characterised by limited resources and knowledge.

In this context, recent literature highlights the critical role of the system modelling activities. On the one hand, accurate digital models are important for manufacturing systems to support analyses and decision-making for enhancing production control, system flexibility, and reconfigurability (Göppert et al., 2021). On the other hand, developing digital models remains a pivotal, time-demanding, and knowledge-intensive activity, often depending on the modelling capabilities of the experts, and poorly reusability-oriented (Schlecht and de Guio, 2023). Furthermore, the manufacturing systems quickly change configurations to adapt to the increasing product customisation and frequent market changes (Yadav and Jayswal, 2018; Bortolini et al., 2018). Also, the presence of different circular economy strategies causes frequent changes and production re-planning (Castiglione et al., 2023). Therefore, digital representations of manufacturing systems can help react to demand changes (Leng et al., 2020), but the digital models can quickly become obsolete.

Furthermore, pursuing the I4.0 paradigm can often lead to introducing new processes and technologies in old systems. Therefore, the activities of digital model development must be implementable in brownfields (Sierla et al., 2022).

Process mining (PM) techniques, initially used in the field of Business Process Management (Van Der Aalst et al., 2007), allow the automated (i) discovery of the digital model of the systems, (ii) comparing the actual system behaviour to its theoretical behaviour (conformance

E-mail address: claudio.castiglione@polito.it.

https://doi.org/10.1016/j.cie.2024.110596
Received 29 September 2023; Received in revised form 4 July 2024; Accepted 20 September 2024
Available online 23 September 2024

check), and (iii) detecting anomalies and changes (Corallo et al., 2020). PM algorithms are based on exploiting an event log, a sequence of events occurring in a system and registered by factory management software (e.g. ERP) through IoT sensors (Rozinat et al., 2009).

This paper proposes a novel event-centric PM paradigm to overcome the risk of obtaining over-representative models (namely, "spaghetti models"), which cannot be effectively exploited (Lugaresi and Matta, 2021), and avoiding system approximations and human user intervention to improve accuracy while reducing modelling time. In particular, the proposed paradigm exploits the IoT technology to collect information on specific points of the manufacturing system to populate an event log, successively exploited for automated generation of the system digital model. Furthermore, the paper also introduces (i) a specific event-centric approach based on the disposition of sensors to detect the three events of job arrivals in the workstation queues and the start and the end of job processing, (ii) an event-centric discovery algorithm for the proposed approach, and (iii) a set of Key Performance Indicators (KPI) to evaluate and compare event-centric approaches.

The proposed event-centric discovery algorithm is based on the Event Relationship Graph (ERG) to model the manufacturing systems. Therefore, the event-centric approach is introduced by ERG, but it is presented together with the Petri Nets (PN) equivalent modelling to highlight the versatility of the entire approach.

A numerical example shows the implementation of the event-centric approach and the discovery algorithm in four realistic manufacturing systems of incremental complexity that consider multi-product systems, assembly, disassembly and split operations beyond processing, re-entrant flows and random reworks. Then, the automated generation of the digital model is evaluated through a set of proposed KPIs to determine a benchmark and highlight the operational and infrastructure requirements of the proposed approach.

The paper is structured as follows. Section 2 presents the background. Section 3 states the problem and summarises the contribution of this paper. Section 4 presents the new PM paradigm, while Sections 5 and 6 summarise the novel event-centric discovery algorithm and the set of Key Performance Indicators for benchmarking, respectively. Section 7 shows the KPIs obtained by implementing the event-centric discovery algorithm in the four realistic systems. Section 8 discusses the experimental results, while Section 9 concludes this paper.

2. Background

In manufacturing, a digital model is the software representation of a physical system by modelling all the relevant characteristics and dynamics that manage and describe its changes over time (Lugaresi and Matta, 2021). The digital models can be exploited by the digital twins of a physical system through simulation techniques, in which the behaviour and the changes of a system are simulated through its digital counterpart (Qiu et al., 2023).

Simulating the digital model of a physical system leads to several advantages in the manufacturing and industrial fields. In particular, simulation is important for analysing complex systems that can hardly be studied through analytical models without a high approximation level (Alfieri et al., 2024). In the system design phase, simulating the digital model of a system can help in sizing the physical counterpart, choosing among several technologies and machines, system configurations, and resource allocation (Mourtzis, 2020). In production planning and control activities, simulation can compare different strategies to react to unexpected events, such as machine breakdowns or changes in customer demand and purchasing costs (Jeon and Kim, 2016).

The advantages of the simulation techniques depend on the reliability of the model and the opportunity to have a digital model aligned with its physical counterpart (Lugaresi and Matta, 2021). Firstly, the data exploited to develop the digital model must allow for an accurate representation of the physical system (Alfieri et al., 2024). Then, the effectiveness and efficiency of the digital model depend on the skills of the expert who developed the digital model itself (Benedettini and Tjahjono, 2009). Finally, the physical system must always keep the same characteristics (e.g. number of machines, product working cycle, and routing policies) and work at the same rates (i.e. stationary condition) to avoid misalignment with the digital models (Law, 2015).

However, the mass customisation trend is leading towards flexible and reconfigurable manufacturing systems to provide a considerable product variety, which is subjected to frequent market demand fluctuations leading to reconfigurations of the manufacturing systems (Yadav and Jayswal, 2018; Bortolini et al., 2018). Moreover, an efficient simulation time is obtained by integrating many different digital models with diverse fidelity levels to properly represent the stand-alone processes or the overall manufacturing line (Zhang et al., 2020).

Hence, the large number of simulation models and the speed with which they become obsolete make the model development phase time-critical. Therefore, several techniques and approaches have been developed in the last three decades to reduce the time and knowledge required by system modelling activities and their dependence on the experts' capabilities.

Some effort has gone into standardising the most common actions exploited by simulation models (Son et al., 2003) (e.g. modules for modelling process activities), while others focused on defining frameworks for outlining the architectures and the rules to obtain digital models. For example, Meng et al. (2013) proposed a UML formal information model to automatically generate simulation models for material handling systems subsequently coded in Arena, while Ra and Choi (2015) provided a framework to develop the digital model based on data predefined in two Excel spreadsheets and a Visio file.

Some research streams focused on interpreting physical documents to create the simulation models directly from there, such as legacy engineering documents released by producers. For example, simulation models have been obtained from IDEF0 and IDEF3 integration (Jeong et al., 2009), technical drawings and instrumentation diagrams provided by machine producers and system designers (Arroyo et al., 2016), or by techniques of object detection and environment and document scanning (Sommer et al., 2023).

Other approaches focused on routines and frameworks to improve the automating generation of the digital model, still partially relying on input and decision-making from users with different degrees of experience. CAD information combined with process information can support the user in the automatic development of a digital model (Dias et al., 2014), or the user can provide the routine that develops the digital model with a set of data defined a priori (Son and Wysk, 2001), or provide the physical information describing the system (Tian and Voskuijl, 2015), or the expert can select among a set of potential digital models the one that most fit the physical system (Capocchi et al., 2020).

Although this automation reduces development time, expert-centred model development remains time-consuming, strictly linked to the person's skills, and scarcely adaptable to unexpected events impacting the system.

In the early 2000s, process mining techniques were developed in Business Process Management to exploit data produced by the systems for three main goals. First, discover the digital model of the system. Second, to compare the system behaviour and its designed requirements to identify deviations (conformance check). The last goal is to analyse the system to identify and address unexpected issues (Corallo et al., 2020).

All the discovery algorithms based on process mining (PM) techniques provide a digital model by interpreting the data produced by a running system (Van der Aalst et al., 2004). The data are the events (e.g. job A starts the transformation process in machine M1 at 3.03 p.m.) registered by sensors displaced along the system; all the events are collected in the so-called event log (Corallo et al., 2020).

The digital model is an ordered collection of elements and links between elements that follow a mathematical formalism to represent the conceptual model of the physical system, and two of the most diffused

formalisms are the Petri net and the Event relationship graph (Schruben, 1983; Savage et al., 2005).

The most common process mining discovery algorithms have been further specialised and adopted in several sectors. For example, in the health sector, a hospital cross-department collaboration algorithm modifies the Petri net by adding specific system-dependent information to improve the digital model quality (Liu et al., 2022). In the software production engineering field, discovery algorithms are specialised by introducing tracing clustering techniques (Ramos-Gutiérrez et al., 2021) for modelling sets of characteristics. In the field of manufacturing systems, process mining techniques, implemented for conformance checking, have been investigated for identifying changes in the manufacturing processes (Rinderle-Ma et al., 2023) and the results of inefficient decisions made at run time on the shop floor (Meincheim et al., 2017). Discovery techniques have been used for identifying and monitoring process-control flows (Mayr et al., 2022) and models representing the sequence of operations of the industrial equipment (Koehler and Jing, 2020). Dos Santos Garcia et al. (2019) provide a comprehensive review and a systematic mapping study of process mining techniques applied in different fields, including manufacturing.

3. Problem statement

The automated generation of digital models by PM discovery algorithms is subjected to the trade-off between over-representative data in the event log and missing data (Lugaresi and Matta, 2021). Missing data can be caused by sensor errors, incomplete information in the event log, or redundancies caused by unusual routing of some jobs within the system (Leemans et al., 2013; Wen et al., 2010). At the same time, exceptional events registered in the event log appear as common behaviour of the system, leading to over-representative digital models, namely "spaghetti models" (Lugaresi and Matta, 2021). These issues can lead to inaccurate digital models or digital models that are mostly different from the physical system due to the noise in the event logs (Weijters and Ribeiro, 2011).

The oldest discovery algorithm in the literature is the Alpha miner, which identifies a Petri net from an event log by analysing the traces (the routing paths) of the elements flowing in the system (Van der Aalst et al., 2004). Alpha miner has been extended to detect hidden activities (Wen et al., 2010), mandatory activities (Wen et al., 2007), and exploits incomplete event logs (Lekić and Milićev, 2021).

The presence of redundancies and poor-quality data in the event logs jeopardises the accuracy of the proposed digital models, fostering the implementation of event log pre-processing algorithms (Chen et al., 2022) and discovery algorithms based on event frequencies. The Fuzzy miner algorithm filters only the activities meaningful for the scope of the analyses (Günther and Van Der Aalst, 2007). The Heuristic miner and the Flexible Heuristic miner discovery algorithms are based on the removal of those connections below a predefined acceptance threshold to handle the noise in the event log (Weijters and Ribeiro, 2011; Weijters and Van der Aalst, 2003), while the Inductive miner algorithm recursively excludes the most infrequent connections by focusing on a part of the digital model at a time (Leemans et al., 2013). The Split Miner algorithm analyses the quality of the graph of the digital model to remove infrequent connections (Augusto et al., 2019). The Business Process Model and Notation Miner follows an approximate approach to remove the dependencies that reduce the digital model quality (Conforti et al., 2016), while the Region-based discovery algorithm selects the smallest model with complete information (Carmona et al., 2008). Further algorithms exploit evolutionary techniques (Buijs et al., 2012) or association rules (Maggi et al., 2012) to reduce the impact of event log noise.

Few papers focused on discovery algorithms in production planning and control of manufacturing systems. Process discovery and conformance checking have been used to detect deviations in the production of modular construction facilities (Rashid and Louis, 2020) and to obtain the digital model of manufacturing systems. Some discovery algorithms for the automated generation of digital twins based on simulation models have been proposed by exploiting approximation techniques to avoid"spaghetti models" (Lugaresi and Matta, 2021) (i.e. over-representative and complicated digital models) and the new object-centric paradigm, which has been exploited for discovering more complex systems that include assembly processes (Lugaresi and Matta, 2023).

3.1. Contribution

The mentioned process mining (PM) discovery algorithms follow an activity-centric approach. In the activity-centric approach, the path (trace) of the entities flowing in the systems is exploited to derive the digital model. The activity-centric paradigm and its recent extension in object-centric, which includes the relationships among entities, can generate over-detailed models (i.e. "spaghetti model"), characterised by a low effectiveness in simulating the real behaviour of its physical counterpart. Hence, activity-centric and object-centric discovery algorithms may require the intervention of a system expert in each new digital model generation to revise the model, provide pre-processed data, or set approximation thresholds for modelling the real system.

Therefore, these algorithms are subjected to the trade-off between fully automated generation without system expert interventions, and the difficulty in modelling complex manufacturing flows as re-entrant flows in the same workstation.

This paper introduces a novel event-centric paradigm for PM discovery algorithms to generate accurate digital models of complex manufacturing systems without the intervention of system experts. The event-centric approach focuses the system expert's role in the sensor placement within the system. Then, the algorithm design phase will exploit the information regarding the sensor positions to ensure that each further regeneration of the digital model will not require revisions and human interventions, even in the case of changes in the physical system.

Hence, in the event-centric paradigm, the role of IoT sensors becomes more pivotal because the effectiveness and efficiency of event-centric approaches depend on the information about sensor positions within the manufacturing system.

The advantage of the proposed event-centric approach is that the behaviour of the productive resources (i.e. machines or workstations) is estimated by exploiting the data coming from the sensors referring to the productive resource itself, neglecting the system configuration (i.e. the connections between different workstations). At the same time, the system configuration is deducted from the positions of the sensors and the data they provide. Hence, the presence of many jobs with heterogeneous traces does not impact the accuracy of the obtained simulation model, and no approximations are required to avoid *spaghetti models*.

Furthermore, the paper presents a discovery algorithm for the novel event-centric PM paradigm for the automated generation of digital models of complex manufacturing systems. The proposed discovery algorithm is tested in a numerical experiment. The results show that the algorithm generates a digital model by identifying re-entrant flows regarding both random reworks and subsequent operations performed by the same workstation, assembly, disassembly, split processes, and processing in a multi-product environment.

The proposed discovery algorithm is the first of this type to explicitly allow the automated generation of digital models of systems under different stationary conditions. Therefore, the algorithm is robust to different states of the physical systems by detecting the buffer levels and dealing with situations in which the event logs contain trunked job traces. In particular, trunked job traces are common when the time windows represented in the event log do not contain the information on the first and last processing of all the jobs involved in the system.

Finally, a set of Key Performance Indicators is provided to measure the accuracy of the digital model obtained, the data redundancy and the number of sensors exploited, and the computation performance

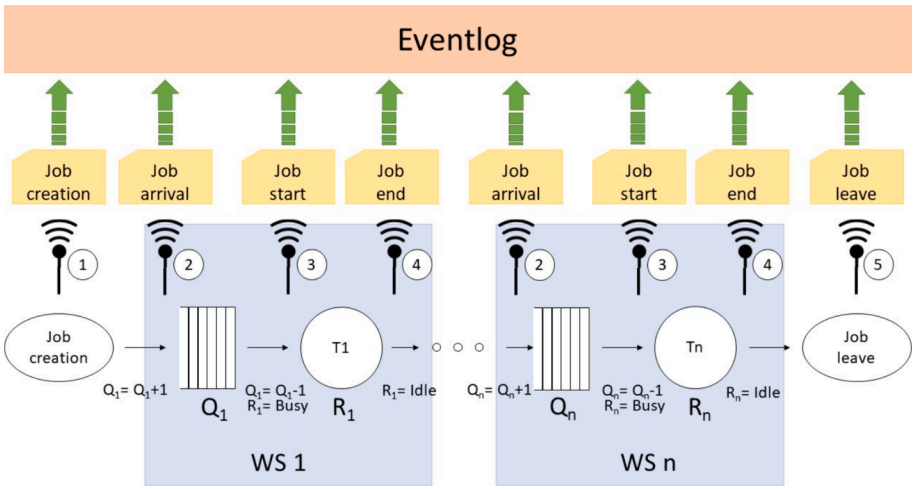


Fig. 1. Sensor positioning for a line with n workstation to capture three events for each workstation: job arrival, processing start, and processing end, detected by sensors of type 2, 3, and 4, respectively.

required.

4. The event-centric approach

In the common activity-centric approaches, trace analysis is sufficient for system discovery: the routing paths performed by the entities are investigated to infer insights regarding the connections among system processes. Conversely, the proposed event-centric approach also requires the knowledge of the criteria for positioning the sensors within the system. This assumption can be acceptable for a manufacturing system in which the production engineers and decision-makers know the position of each sensor. In fact, the positions are chosen to satisfy specific goals of production monitoring.

The positioning criteria intertwine the detection of a job to a particular event, such as the start of production or the identification of a quality issue. The points of observation can be chosen arbitrarily, and the large availability of sensors can allow the detection of many different types of information. In each critical location, one or more IoT sensors can be introduced based on the nature and the characteristics of desired information. The type of required information depends on the level of detail of the simulation model and the objectives to which it is dedicated.

Although the sensors can be positioned arbitrarily in manufacturing systems, some positions are privileged to capture several pieces of information with fewer sensors and data for each single detection. This paper proposes an approach based on three particular locations for the sensor positioning that together are capable of providing the following pieces of information:

- queues;
- resource state and stochastic processing times;
- stochastic job arrivals in the system and their routing paths, even in the presence of assembly and disassembly operations;
- re-entrant flows and random reworks;
- system performance in terms of WIP, cycle time, and inter-departure times.

4.1. The three-event model

Within the novel event-centric PM paradigm, this paper proposes an approach based on three crucial positions that allow the collection of three important events for manufacturing systems:

- job arrival in queue (JA);
- start of job processing (JS);
- end of job processing (JE).

Fig. 1 shows the three positions of the sensors (pins in Fig. 1) in a flow line (flow direction determined by the arrows in the figure) consisting of *n* workstations with unlimited buffers for waiting times before the job processing (circles and rectangles in the figure, respectively). The sensor numbering indicates the type of event the specific sensor is deputed to detect in each workstation: JA, JS, and JE. The type of event (JA, JS, and JE) detected by the sensor is indicated in the yellow tags over the sensors. All the events are registered in the event log.

The minimum pieces of information collected by each sensor every time it detects a new job can be commonly retrieved in realistic systems:

- job ID of the job detected;
- sensor ID of the sensor registering the occurred event;
- time;
- job type ID to distinguish different classes of processed jobs.

Like in the activity-driven and object-driven PM approaches, the event log collects the events of a specific time window, which will be exploited to discover the system. Table 1 reports an example of the event log structure exploited in this paper, in which each row is an event detected by a sensor. For example, in line 1, the job of type"A" and ID"1″ is detected by sensor"1″ at 17:24. The same job is successively detected by sensor"2″ and sensor "3″ one minute and four minutes later, respectively. At the same time, sensor"1" detects a new job of type"B" and ID"3".

Table 1
An example of an event log exploited in this paper: the columns from left to right report the Job ID, the sensor ID, the timestamp, and the type of job, respectively.

Job ID	Sensor ID	Timestamp	Job type
1	1	07/19/2023 – 17:24	A
1	2	07/19/2023 – 17:25	A
2	1	07/19/2023 – 17:28	A
3	1	07/19/2023 – 17:29	B
1	3	07/19/2023 – 17:29	A
2	2	07/19/2023 – 17:30	A
.	.	.	.
.	.	.	.
.	.	.	.

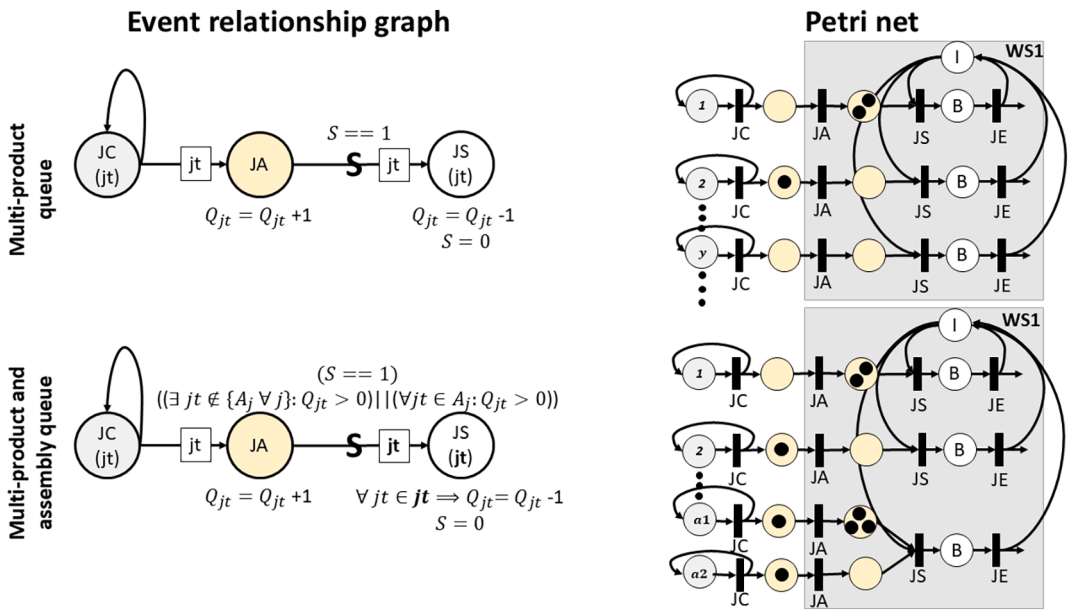


Fig. 2. ERG (on the left side) and PN (on the right side) modelling of a Job arrival event representing a job of type *jt* arriving in the queue of a machine deputed to a processing operation or an assembly operation on the top and the bottom side of the picture, respectively.

4.2. Event connections and job processing cycles

Each one of the event types identified for modelling manufacturing systems triggers a finite and limited set of actions that describe the system dynamics. Generally, the system dynamics can be modelled through Petri nets (PN) and Event Relationship Graphs (ERG), which are alternative methods to model systems like those discussed in this paper. PN and ERG have been broadly adopted to represent manufacturing systems since the early 80 s. The PNs were already applied for similar problems (Cecil et al., 1992), and today, they are broadly accepted for modelling manufacturing systems, while Schruben proposed ERGs in 1983 Schruben (1983). Like PN, the ERG approach can simulate a Turing machine; thus, it can represent any system implemented on a modern computer (Savage et al., 2005), and it can model a PN, while a PN cannot always model an ERG (Matta et al., 2014). ERGs are based on nodes representing the events, arcs oriented (arrows), and conditioned arcs (oriented arrows with an **S** in the middle and a logical condition expressed over them) that connect each event with those that it triggers (Chan and Schruben, 2008).

The event-centric approach can benefit from the ERG representation since it allows a more compact model of both the system structure and the state variables of resources and queues (i.e. whether the machine is busy or idle and the number of entities in the queue). For this reason, the event-centric approach is presented through ERG models, while the PN equivalent (in terms of the conceptual model) counterpart is proposed to introduce ERGs.

This paper assumes production systems in which sensors are positioned as in Fig. 1, except for the events of *Jobcreation* (JC) and *Jobleaves* (JL) that are not detected through the physical system but assumed by the derived digital model. Hence, each workstation has three sensors to detect *Jobarrivals* (JA), *Jobstart* (JS), and *Jobend* (JE) events. A node in the ERG represents an event, so there will be a number of nodes representing JA, JS, and JE events equal to the number of workstations n.

The system characteristics considered in this paper include unlimited buffer capacities, workstations not subjected to failures, and queues following a FIFO priority rule.

The following subsections introduce the three main sub-models of the system model generated by the information provided by each sensor according to its position and the type of event triggered. Therefore, the following three sub-models are replicated for the number of sensors of each type present in the system and combined to obtain the overall digital model through the algorithm presented in Section 5.

4.3. Events triggered by job arrival

Fig. 2 shows the behaviour of the JA event (yellow nodes in the ERGs in the figure) that models the arrival of a job in the buffer preceding the workstation. On the left side of Fig. 2, the events triggered by the JA event are modelled through ERGs, while on the right side, there are the PN counterparts. The JC event (JC nodes in the ERGs of Fig. 2) triggers two events through its outgoing arcs: (i) the JA event representing the arrival in the queue before the workstation 1 (WS1) of the newly created job characterised by a unique job type (jt) that indicates the class of item; (ii) the entry into the system of a new job *i* of job type *jt* (the re-entrant arc triggering a new JC event).

The occurrence of the JA event causes an increment in the queue before WS1 for the job type *jt* (Q_{jt} in Fig. 2). The JA event is connected only with the JS event that can be triggered if and only if the machine is idle (i.e. the S variable indicating the state idle of the resource is equal to 1 over the conditioned arc connecting the two events). The symbol *jt* in the nodes JC and JS indicates that the event refers to the job with job type *jt* specified in the box over the incoming arc.

The ERG representation of JA in the bottom part of Fig. 2 models the case of a queue before a workstation capable of both processing single items and assembling several items. The condition triggering the JS is different with respect to the ERG in the top part of Fig. 2 to specify that the JS event can be triggered if the server is idle (S equals 1) and the newly arrived job is a single item or if it must be assembled with other items, then they have already arrived in queue.

In this case, the arc (JA, JS) and the indication (***jt***) in bold in the JS event indicate a vector of job types because more than one job is simultaneously processed.

The PNs on the right side of Fig. 2 model the same described events by representing the events with the transitions (black boxes), while the places (the circles) determine the system states (e.g. if the resource is idle I or busy B, and the number of tokens (i.e. the black circles) in the queue, waiting for triggering the JS event).

Standard PNs require different independent paths for each job type processed by the workstation WS1 (1, 2, and y, or a1 and a2, in the case of assembly, in the bottom part of Fig. 2). The idle state of the resource

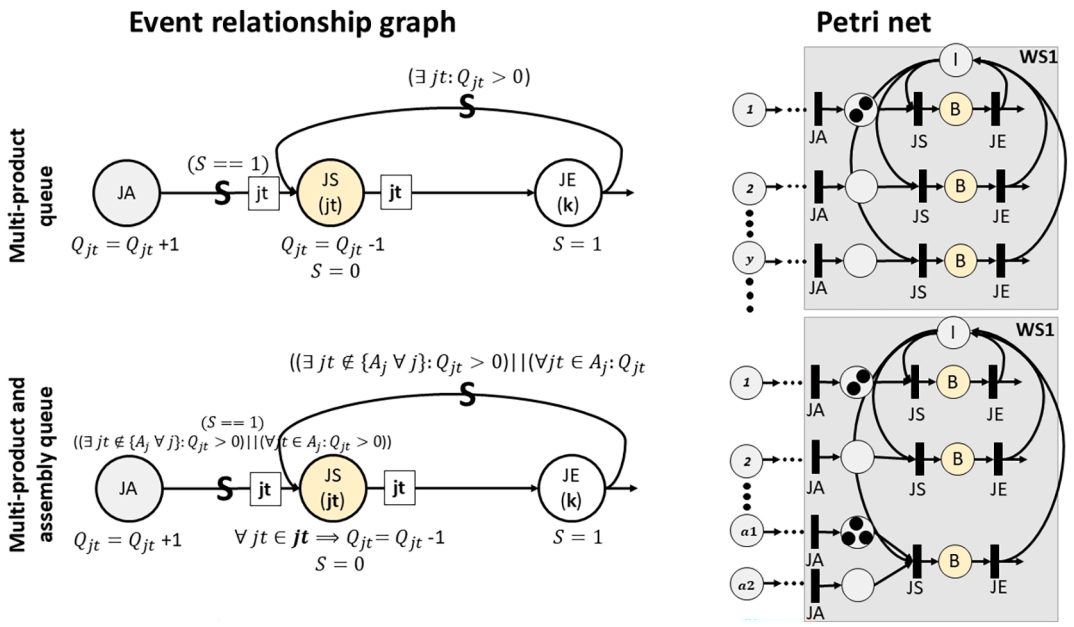


Fig. 3. ERG (on the left side) and PN (on the right side) modelling of a Job start event representing a job of type *jt* starting the processing in a machine deputed to a processing operation or an assembly operation, on the top and the bottom side of the picture, respectively.

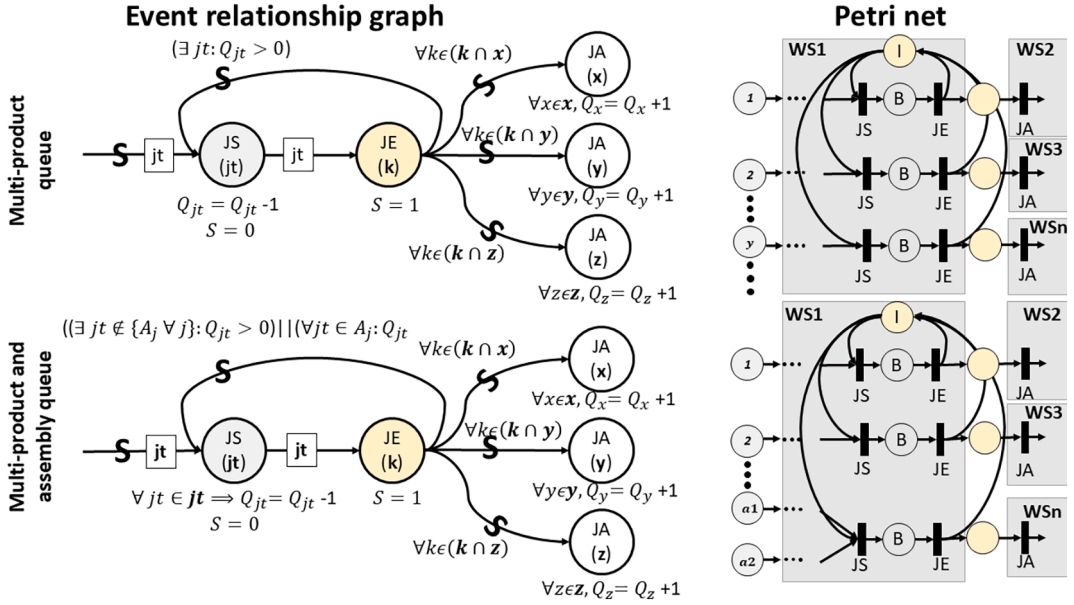


Fig. 4. ERG (on the left side) and PN (on the right side) modelling of a Job end event representing a job of type *jt* ending the processing in a machine deputed to a processing operation or an assembly operation, on the top and the bottom side of the picture, respectively.

(place with the I) connects the different paths.

4.4. Events triggered by job start

Fig. 3 introduces the behaviour of the JS events (yellow nodes in the ERGs of Fig. 3, while in the PNs, yellow nodes are the places triggered by the JS transition) that model the process start and trigger the JE event. The JS event always triggers the same JE event after a stochastic time $t_{jt,r}$ indicating the processing time t required to process jobs of type jt in resource r. When the JS event is triggered, the queues of the job type in production are decreased by the number of simultaneously processed units (Q_{jt} is reduced by 1 in the example in Fig. 3), and the resource state becomes busy (numbers of idle servers S equals 0). In the case of assembly or batch production, the arc (JS, JE) indicates a vector (***jt***) due to the items of different classes simultaneously processed.

4.5. Events triggered by job end

Fig. 4 shows the JE event, which triggers a new JS event conditioned to the presence of items waiting in the queue and triggers at least one JA. The JE events indicate a vector of job types (***jt***) because they can detect one or more jobs belonging to different classes. The jobs exiting from the process can be directed to the same JA event (the queue of the next workstation) or various JA events, or they can go out of the system.

The PNs on the right side of Fig. 4 highlight that a specific path is required for each *jt* exiting from the JE event and sent to one particular JA event. Hence, ERG representation is more compact than PN one since

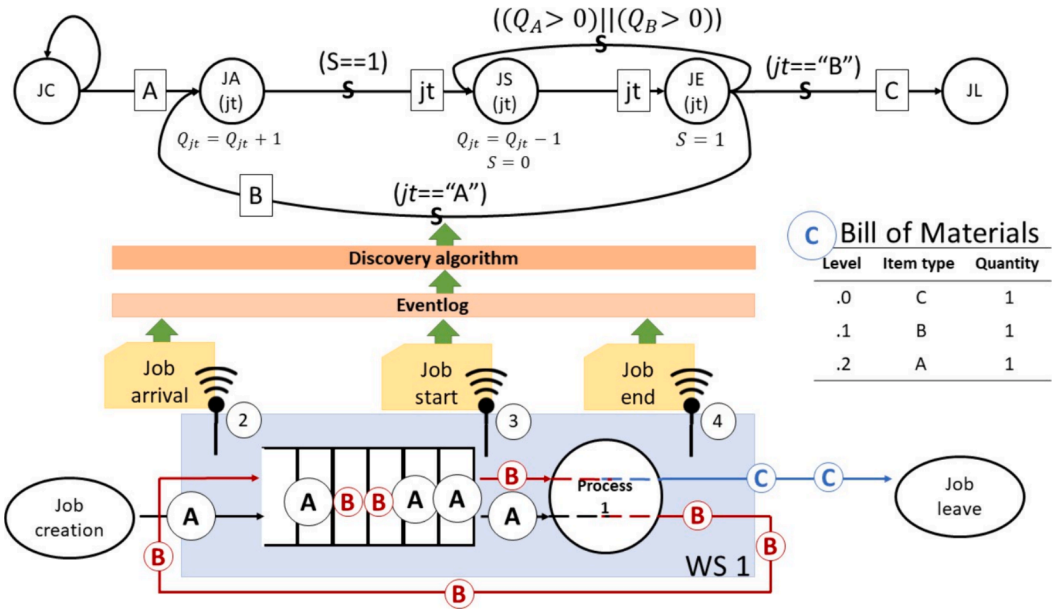


Fig. 5. The digital model (on the top of the figure) created by the discovery algorithm by exploiting the event log that collects all the events of a physical system consisting of a workstation that performs two different processes to obtain job type C from the raw material A.

the latter requires many transitions and many arcs connecting the transitions for the events detected by the same sensor.

4.6. Job identification

The notation used to identify the jobs is crucial for properly connecting the events detected by the sensors. According to the ERP systems, the event-centric approach exploits the convention in which each item belongs to a unique class consisting of all jobs subjected to the same processes and having the same Bill of Materials (BOM). For example, Fig. 5 shows the production processes of the finished product C: the raw material A is processed by workstation 1 (WS1) to obtain item B. Item B is re-processed by WS1 to obtain finished product C. Three sensors (sensor ID 2, 3, and 4) detect the three types of events: new job arrival in the queue (job type A or B), start of the job processing (to produce job type B or C), and end of job processing. The events are collected in the event log exploited by the discovery algorithm to generate the digital model on the top of Fig. 5.

4.7. The event-centric discovery algorithm

This paper proposes a three-event model (3EM) discovery algorithm based on the proposed event-centric approach exploiting three different types of events: JA, JS, and JE. The algorithm is intended for systems characterised by multi-product manufacturing, assembly, disassembly, batch processes, random rework, and re-entrant flows.

The digital model $F(N, A, R, P)$ of a manufacturing system provided with n sensors consists of:

- a set of nodes $N = \{i | i = 1, \ldots, n\}$ in which node i represents the event detected by sensor i, in a system provided with n sensors. N is the union of three subsets $N = N^A \cup N^E \cup N^S$ representing the set of nodes modelling JA, JS, and JE events, respectively. The dynamics of each workstation m are modelled through the three nodes modelling the events JA_m, JS_m, and JE_m;
- a set of arcs $A = \{a | a = (i, j) \forall i, j \in N\}$ for the connections between the nodes;
- a set of routing rules R, which describes the connecting conditions between the JE of a workstation and the JA of the successive one (i.e. model the routing of jobs);

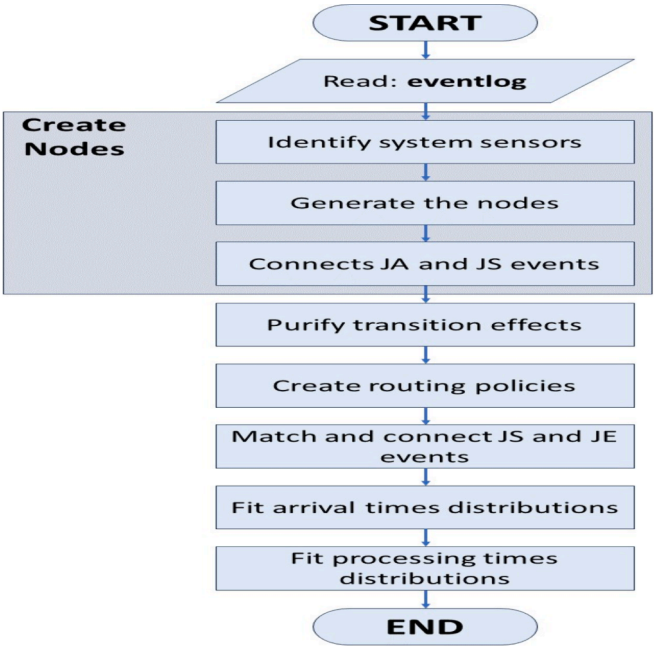


Fig. 6. The main steps of the 3EM discovery algorithm start with the creation of the nodes.

- a set of production rules P to intertwine the output of a process (JE event) with its input raw materials (JS event);

The nodes and the arcs follow the dynamics described in the subsections of Section 4 because the types of event considered by the algorithm are the same as those introduced before (JA, JS, and JE). Hence, the 3EM discovery algorithm must identify the nodes, the type of events modelled by the nodes, the arcs connecting the events, and the production and routing rules are the conditions, the additional arcs, and the probabilities that adapt the conceptual model to the physical system.

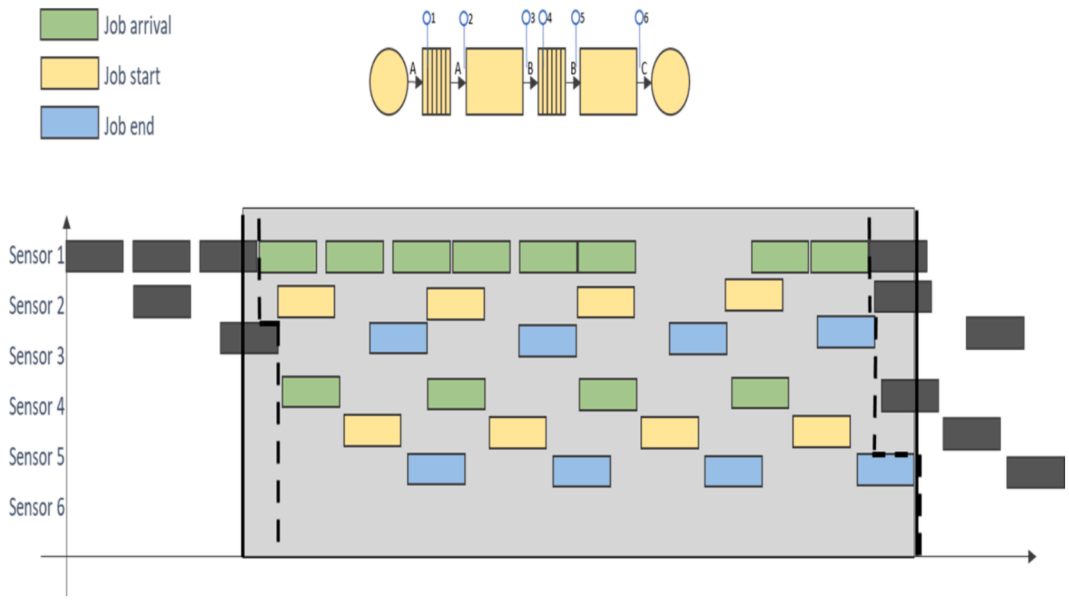


Fig. 7. Representation of the event log time window, delimited by the solid vertical lines and the two rules highlighted by the dashed lines. The dashed line on the left postpones the time window that starts after the end of a JE without its JS. The dashed line on the right anticipates the time window end by discarding the JS event without the respective JE.

4.8. The algorithm structure

The digital model $F(N, A, R, P)$ is a representation of the behaviour of a physical system framed through the events registered in the event log. The event log follows the structure shown in Table 1, describing a time window that includes the events between the first and the last events registered.

The flow chart in Fig. 6 presents the main steps of the algorithm starting from the event log. The creation of the nodes involves three phases. First, the events from the event log are exploited to create a node for each sensor to model the JA_m, JS_m, and JE_m for each workstation m.

The second step of node creation exploits the identified sensors to recognise different types of nodes (JE, JS, and JA). There will be three events for each job ID under the same job type (BoM level), recorded by three sensors. Chronologically, the first event of each job ID with a certain job type jt is a JE_m, in which the job has just been produced by workstation m. Then, the second event is JA_{m+1}, in which the job enters the queue of the next workstation. The last event detected for each job ID under the same job type jt is JS_{m+1}, when it starts to be worked, and when it will obtain a new job type (next BoM level identified by a new job type $jt' \neq jt$). This procedure creates the nodes belonging to the three sets N^A, N^E, *and* N^S, and it is explicated in Algorithm 1 in Appendix A.

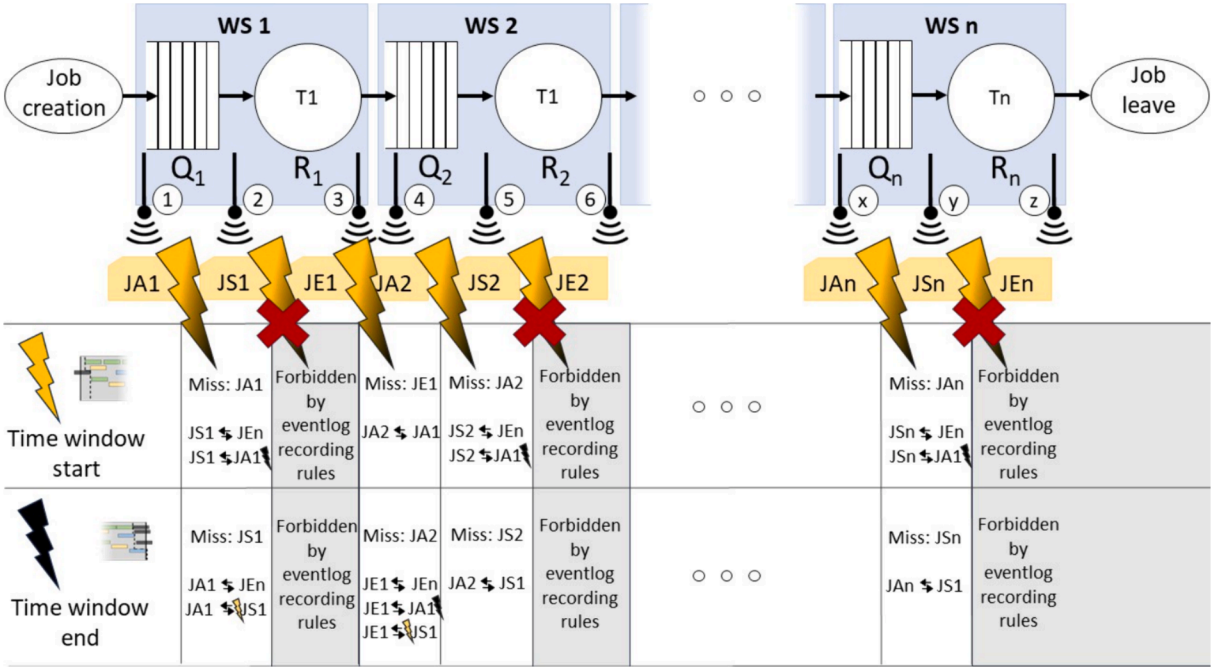


Fig. 8. The redundancies generated by the start (those following the yellow lightning bolt) and the end (those following the black lightning bolt) of the event log time window in the cases of an incomplete event log. The crossed lightning bolts indicate the redundancies avoided by the two event log recording rules. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

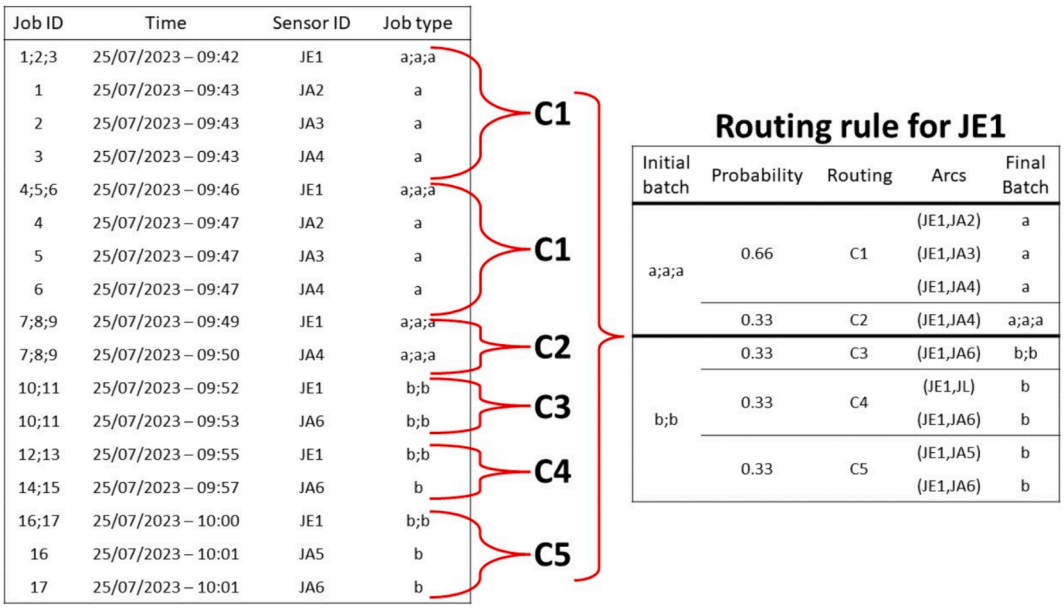


Fig. 9. The event log on the left side shows the productions of JE1. JE1 produces two batches: (i) three'a' products; and, (ii) two'b' products.

The last phase of node creation directly connects nodes belonging to N^A (representing the JA events) are directly connected with nodes belonging to N^S (representing the JS events) by exploiting the job traces. This connection can be made because, after a job arrival in a machine of a specific job, it can only follow the event of the job starting in the same machine.

Following this procedure, the created sets N^A, N^E, *and* N^S contain all the nodes modelling the sensors of the physical system. If the event log contains the JA, JS, and JE of each job under each job type (i.e. all the levels of the BoM), then the sets N^A, N^E, *and* N^S are already correct. Otherwise, when the event log is a subset referring to a limited time window, the exploited event log will contain some incomplete traces. The information provided by the incomplete traces can help the accuracy of the modelling of the single processes, but they can lead to erroneous representations of the overall physical systems. In these cases, some redundancies are caused by the procedure of creating the nodes in chronological order. Hence, some sensors will be modelled with more than one node belonging to different sets because of the misleading information reported in the event log. For example, some jobs will seem to exit from the system because the events of the last operations fall out of the time window. At the same time, other jobs will seem to enter the system from unexpected detection points as their previous events precede the start of the recorded time window.

Two strategies are implemented in the algorithm to mitigate this risk of misleading information in the event log: (i) two rules to record the event log, (ii) the development of a function to delete from the event log the transition effects of the time window.

Fig. 7 shows the time window and the two rules for recording the events in the event log. The first rule states that for each job whose JS is stored, the JE must also be stored (if the JE event concludes after the black solid vertical line, accept it by exceptionally postponing the end of the time window with the dashed vertical line). Similarly, the second rule states that if a JE should be recorded, but its JS misses because it preceded the time window start, then neglect both the JS and the JE. The two rules for recording the events prevent jobs with a JE or a JS from missing its reciprocal JS or JE.

However, some other redundancies remain and are summarised in Fig. 8.

The lightning bolts indicate the positions where the event log time window starts or ends, interrupting the sequence of events. Some lightning bolts are crossed because the two event log recording rules forbid the interruption of event sequence, but the others are still allowed and cause redundancies. The redundancies in the first line (those following the yellow lighting bolt) are caused by the start of the time window, while the redundancies in the second line are caused by the end of the time window (those following the black lightning bolt).

The main redundancies involve: (i) the JA and the JS events in the first workstation after the entry point (in the cases of start and end of the time window), (ii) the last JE before the job exit from the system (JE_n), and (iii) exchanges between JA of other stations and the JS of the first workstation. Therefore, a purify algorithm (Algorithm 2 in Appendix A) is run after the node creation to remove redundancies by double-checking if just a few cases show exceptional behaviour to identify the misled information that will be removed.

After the purifying algorithm, the third step presented in Fig. 6 is the creation of the routing policies, which are a set of arcs connecting the nodes of N^E to the nodes of N^A. Given a node j representing an event *JE* (i.e. $j \in N^E$), it models the end of the production (or assembly) of a products jt_j or batch of products $\boldsymbol{jt}_j$ produced in the workstation j. The set of all the single products jt_j and the batches of products $\boldsymbol{jt}_j$ produced by j is $\boldsymbol{jt}_{j,0}$ (i.e. $jt_j, \boldsymbol{jt}_j \subseteq \boldsymbol{jt}_{j,0}$).

The produced jobs and batches of jobs belonging to $\boldsymbol{jt}_{j,0}$ are sent to the following stations, which are detected by the sensors that trigger the events of type JA. However, the same produced job jt_j can have multiple potential workstations of destination; for example, while jobs usually go to the workstation $j+1$, defective job jt returns to the queue of workstation j to be reworked with a probability $p_{j,jt}$. Therefore, workstation j has a matrix of probabilities $\boldsymbol{p}_j$ where the rows are the vectors of probabilities $p_{j,jt}$ with which each job jt is sent to the following workstations. The vectors $\boldsymbol{p}_{j,jt} \in \boldsymbol{p}_j$ are assessed by counting the frequencies reported in the event log.

Each batch of products $\boldsymbol{jt}_j$ produced in the workstation j could be split in many ways into new sub-batches following different routing rules. Therefore, for each batch $\boldsymbol{jt}_j$, a set of n subsets represents all the combinations $\boldsymbol{jt}_{j,c}$ (i.e. $c = \{1, \ldots, n\}$) in which it can be split. Each subset $\boldsymbol{jt}_{j,c}$ consists of a variable number of m_c new sub-batches $\boldsymbol{jt}_{j,c,s}$ (i.e. $s = \{1, \ldots, m_c\}$) that are dispatched to the following stations with the probability vector $\boldsymbol{p}_{j,jt,c,s}$.

Each node j representing an event *JE* is connected by arcs with a set of JA events represented by a subset of nodes $N_j^A \subset N^A$. An oriented arc a

(j, i) connecting the node j with a node i exists for each produced job jt_j or a batch $\boldsymbol{jt}_j$ or sub-batch $\boldsymbol{jt}_{j,c,s}$ sent to the node i, given that i is a node modelling JA event type (i.e. $j \in N^A$). All the arcs outgoing from node j belong to the set $A^j \in A$. The set of all the arcs that can be followed by a produced product jt_j or batch of products $\boldsymbol{jt}_j$ (even when it is split into sub-batches $\boldsymbol{jt}_{j,c,s}$), according to the probability matrix $\boldsymbol{p}_{\boldsymbol{jt}}$, is $A^j_{jt} \subset A^j$.

A routing rule for a product jt_j, r_{jt_j}, or a batch of products $\boldsymbol{jt}_j$, $r_{\boldsymbol{jt}_j}$, is the tuple consisting of the set of arcs that the product or the batch can follow A^j_{jt} and the related set probability matrix $\boldsymbol{p}_{\boldsymbol{jt}}$. Therefore, $r_{jt_j} = \left\{A^j_{jt}, \boldsymbol{p}_{\boldsymbol{jt}}\right\}$, R^j is the set of the routing rules of all the products and batches of products exiting from node j that $R^j = \left\{r_{jt_j}, \forall jt, \boldsymbol{jt} \in j\right\}$. The set R consists of the set of all the routing rules of all the nodes belonging to N^E that $R = \left\{R^j, \forall j \in N^E\right\}$.

Fig. 9 shows an example of the procedure to identify the routing policies for the node JE_1 through the event log of the system shown in Fig. B.11 in Appendix B. Throw the event log in Fig. 9, five routing rules connecting JE_1 with the JA nodes of the fed workstations can be identified (C1 and C2 for the'a,a,a' initial input; and C3, C4, and C5 for the'b, b' initial input). According to the frequency of occurrences of the different routing rules, a probability is assigned to each connection (probabilities from the probability matrix).

After the purifying algorithm to remove redundancies generated from the incomplete information in the event log, the next step of the 3EM discovery algorithm in Fig. 6 regards the connections between nodes belonging to N^S and N^E. All the events of each node $j \in N^S$ are compared with the events of each node $i \in N^E$ to evaluate whether they can be the starting and ending times of the same job processing.

All the couples of nodes (i, j) that have the same number of events (as there is no JE without a JS and vice-versa due to the two recording rules of the event log) and all the events of $j \in N^S$ precede the respective events of $i \in N^E$ are matched.

In this step, each node c can be matched with more than one node $i \in N^E$, and vice-versa. For each match, the events of the two nodes are coupled to obtain a sample of job processing times for each processing that obtains a specific output by exploiting a given input. The variances of the samples of processing times of each match $Var(i,j)$) are summed to obtain an estimation of the overall variance of the matches. The following integer linear programming (ILP) optimisation model, named Variance Minimization Problem (VMP), is solved to determine the unique couples of nodes $j \in N^S$ and $i \in N^E$ that minimise the overall variability of the system.

$$\min \sum_{j \in N^S} \sum_{i \in N^E} Var(i,j) * x_{i,j} \tag{1}$$

Subjected to

$$\sum_{j \in N^S} x_{i,j} == 1 \forall i \in N^E \tag{2}$$

$$\sum_{i \in N^E} x_{i,j} == 1 \forall j \in N^S \tag{3}$$

$$x_{i,j} \in \{0,1\} \forall j \in N^S, \forall i \in N^E \tag{4}$$

Constraints (2) and (3) ensure that each node $j \in N^S$ is coupled with one and only one node $i \in N^E$ and vice-versa. Constraints (4) define the Boolean variables $x_{i,j}$ that are equal to 1 if the couple of nodes (i, j) contribute to minimise the overall variance, and 0 in the other cases. The presented matching procedure is summarised in algorithm 3 in Appendix A.

Therefore, each node $j \in N^S$ models a JS event, and it is connected with one and only one node $i \in N^E$ modelling a JE event by an arc a(i, j) $\in$ A, which is the set of all the arcs of the digital model. Therefore, JS_j and JE_i will coincide with the detection of the starts and ends of the processing of the jobs processed in workstation w.

Each product $jt_{j,input}$ or batch of products $\boldsymbol{jt}_{j,input}$ detected by JS_j is starting to be processed or assembled/disassembled by workstation j. Product jt_j will be substituted by one or more products ($jt_{j,output}$ or $\boldsymbol{jt}_{j,output}$, respectively) of another job type (new level of the BoM) that will be detected by JE_j that is the event of exiting from workstation j. For example, job $jt_{j,input}$ can become a good product $jt_{j,output}$ or a defective $jt'_{j,output}$. Therefore, a probability matrix $p_{jt_{j,input}}$ will assign to each product $jt_{j,input}$ or batch of products $\boldsymbol{jt}_{j,input}$ a probability of becoming one of the potential finished outputs or batch of outputs. The set of all the potential derived products and batches of products ($jt_{j,output}$ or $\boldsymbol{jt}_{j,output}$, respectively) is $o_{jt_{j,input}}$ (i.e. $jt_{j,output}$, $\boldsymbol{jt}_{j,output} \in o_{jt_{j,input}}$).

A production rule $s_{jt_{j,input}} = \left\{o_{jt_{j,input}}, p_{jt_{j,input}}\right\}$ combines all the potential products or batches of products ($jt_{j,output}$ *or* $\boldsymbol{jt}_{j,output}$, respectively) derived from product $jt_{j,input}$, , or batch of products $\boldsymbol{jt}_{j,input}$, and the probability matrix $p_{jt_{j,input}}$.

The set of production rules $P^j = \left\{s_{jt_{j,input}} \forall s_{jt_{j,input}}, s_{\boldsymbol{jt}_{j,input}} \in j\right\}$ contains the production rules of all the products and batches of products processed by workstation j. The overall set of production rule P contains the set of the production rules of each workstation in the system (i.e. $P = \left\{P^j, \forall j \in N^S\right\}$.

Finally, the last two steps of the 3EM algorithm in Fig. 6 state the fitting of the stochastic distributions of the inter-arrival times and processing times. Therefore, the digital model $F(N, A, R, P)$ is complete, all the nodes are available and connected through the arcs that follow the dynamics explicated in the subsections of Section 4. The production and routing rules provide the connecting conditions and probabilities that allow the digital model to replicate the physical system.

However, based on the discovery algorithm implemented, many digital models can be derived from the same physical system. Therefore, the following section has developed a set of key performance indicators to compare and benchmark the alternative digital models.

5. Key performance Indicators for event-centric approaches

Event-centric approaches can automatise a wide range of systems from product and process industries by focusing on different aspects and levels of detail. However, the wide range of implementation opportunities makes it difficult to evaluate whether an event-centric approach is better than another in terms of efficiency, investment costs, and effectiveness. Furthermore, unlike the activity-centric and the object-centric approaches, the event-centric approach has an explicit dependence on the IoT infrastructure, which may require investment costs and knowledge that can limit the affordability and capability of implementation.

For these reasons, this paper proposes a set of KPIs for assessing event-centric discovery approaches that consider the complexity of the IoT infrastructure in addition to the dimensions of efficiency and effectiveness of the algorithm. After introducing the KPIs in this section, the next section will implement them to evaluate the performance of the proposed 3EM discovery algorithm.

The quantitative KPI set consists of two subsets: IoT infrastructure and discovery algorithm. The two subsets of KPIs must always be coupled with the information set of the physical system that the event-centric approach models.

5.1. Assessing the IoT infrastructure

This subset of KPIs allows the investment cost evaluation by considering the complexity of the IoT system. It consists of five KPIs: the types of sensors required (Sensor type); the number of sensors per type for each workstation (Quantity); the number of types of events detected

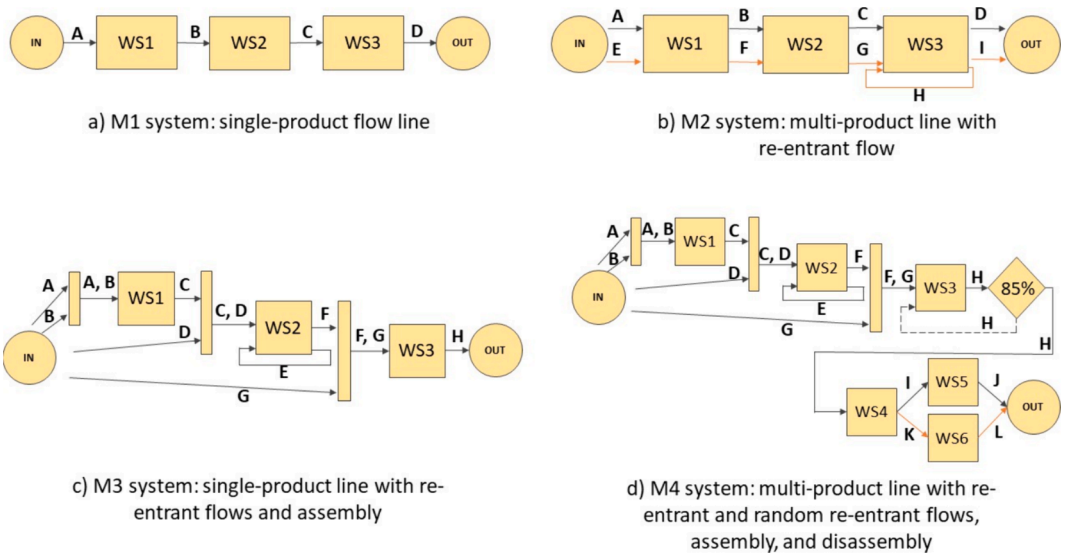


Fig. 10. The four systems used to test the automated generation of the digital model.

by the same sensor (Events per sensor type); the amount of data fields stored per event (Data fields per event); and the average number of events in the event log (Average number of events).

Sensor type. The set of types of sensors S collects and reports the information regarding each single sensor s required by the event-centric approach.

$$S = \{s_i : i = 1, .., I\} \tag{5}$$

Quantity. The total number of sensors Q for each workstation allows to estimate the number of all the sensors required by the event-centric approach. It aims to make different approaches comparable by estimating the size of the IoT network through the quantities q_i for each sensor $s_i \in S$.

$$Q = \sum_{i=1}^{I} q_i \tag{6}$$

Events per sensor type. The KPI E_i indicates the number of the event types detected by the sensor of type i given $s_i \in S$. The larger the number of events detected by each sensor, the more complex the IoT infrastructure and the digital model.

$$E_i = \left\{ e(i)_j : j = 1, ..., J_i \right\} \tag{7}$$

Data fields per event. The number of data fields $Fe(i)_j$ per event type j detected by the sensor $s_i \in S$ represents an estimation of the complexity of the elaborations required by the event-centric approach. The higher the value of $Fe(i)_j$ for each sensor, the more sophisticated the methods for extracting information and the more computational resources involved in processing the event log.

Average number of events. The average number of events A_n estimates the average size of the event log required to ensure the effectiveness of the discovery algorithm with the indicated impacts on its efficiency. The higher the A_n, the larger the event log required by the algorithm is. Therefore, the algorithm requires wide time windows of data to provide the digital model of the system. An algorithm with a large A_n may be unhelpful for systems characterised by frequent changes, for example, in the type of produced jobs or the resource allocation within the workstations.

5.2. Assessing the event-centric discovery algorithm

The event-centric discovery algorithm is assessed in terms of efficiency and effectiveness. The effectiveness is the level of accuracy achieved by the au- tomated generated digital model when compared with the theoretical model (e.g. the Petri net or the Event relationship graph). The efficiency regards the required time to automatically generate the digital model, as the quickest algorithms allow the frequent regeneration of the digital model to keep it aligned with the physical counterpart.

The effectiveness of the algorithm is evaluated by comparing the com- pleteness of the digital model with respect to the theoretical model along the following five KPIs:

- the percentage of the identified number of nodes (Nodes);
- the average number of correct incoming connections per node (Incoming connection);
- the average number of correct outgoing connections per node (Outgoing connection);
- the average number of correct modelling of the BoM levels (Bom levels);
- the accuracy of the routing connections (Routing accuracy).

The efficiency of the algorithm is assessed along two dimensions of time:

(i) the computation time required by the algorithm and (ii) the duration of the event log time window required by the algorithm to obtain a reliable digital model.

6. Numerical experiment

The event-centric discovery algorithm 3EM is tested in the automated generation of the digital models of four realistic manufacturing systems with different configurations (M1, M2, M3, and M4) characterised by increasing complexity. Fig. 10 shows the four manufacturing systems from M1 to M4. M1 is a single product line with three processes, while M2 is the multi-product version of M1 in which product I (orange arrows) must be processed two times in the last workstation (fixed re-entrant flow). System M3 combines re-entrant flows, like the one in M2, with assembly processes. In particular, WS1 and WS3 perform only assembly tasks. WS2 perform both assembly tasks to obtain component E from components C and D and process tasks to transform component E into component F. System M4 extends M3 by including: random reworks after WS3 (probability of rework equals 15 %); a disassembly workstation M4 that simultaneously produce components I and K from H; and WS5 and WS6 to obtain the finished products J and L, respectively. The BoMs of the finished products of the systems from M1 to M4 are reported

Table 2
The experimental results of the automated generations of the digital models.

System typ	Bottleneck utilisation	Starting Time (min.)	Time window width (min.)	event log width (min.)	Events	Computation budget (s)	Accuracy
1	0.48	0	7.80	8.58	34	0.01	✓
1	0.48	2500	15.09	15.36	31	0.01	✓
1	0.48	5000	11.54	13.02	50	0.01	✓
1	0.48	7500	14.53	15.42	36	0.01	✓
1	0.48	10,000	15.33	16.66	24	0.01	✓
2	1.43	0	18.84	18.84	100	0.02	✓
2	1.43	2500	1961.61	1961.68	12,414	1.66	✓
2	1.43	5000	3709.87	3710.92	23,133	4.89	✓
2	1.43	7500	5462.79	5464.15	34,213	8.92	✓
2	1.43	10,000	7214.78	7215.24	45,016	19.51	✓
3	0.96	0	11.20	11.20	49	0.01	✓
3	0.96	2500	148.85	149.14	738	0.04	✓
3	0.96	5000	124.26	124.53	618	0.05	✓
3	0.96	7500	185.04	186.20	874	0.04	✓
3	0.96	10,000	360.54	361.71	1913	0.16	✓
4	0.96	0	16.37	17.27	84	0.02	Miss the random rework
4	0.96	2500	105.67	106.42	879	0.05	✓
4	0.96	5000	243.32	244.38	1900	0.12	✓
4	0.96	7500	341.33	341.93	2775	0.22	✓
4	0.96	10,000	305.48	306.81	2496	0.19	✓

for clarity in Fig. B.12 in Appendix B.

The four systems have been modelled with Arena Simulation Software to simulate the system dynamics and collect events to populate the event logs exploited by the 3EM algorithm. The event logs have been used for the automated generation of the digital model of the physical systems (the ones simulated through Arena). The obtained digital models are compared with the conceptual models of the physical systems. The experiments have been performed on a machine with Windows 10, an Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz, and 16 GB of installed RAM.

The accuracy of the 3EM algorithm depends on the capability of purifying the model from the transition effects of the start and the end of the event log time window. Some factors referring to the physical system influence the purifying capability, such as bottleneck utilisation and whether the system has reached a stationary state. These factors influence the number of jobs waiting in queues and the presence of interrupted traces in the event log.

Longer event log time windows can mitigate these effects, improving the accuracy level of the digital model. However, excessively larger event logs require longer periods of event collection by observing the physical system and larger computation time to obtain the physical model.

It has been experimentally observed that the 3EM discovery algorithm requires the event logs to contain at least two entire traces (i.e. all the events from the entry point JC to the exit point JL of the system) of the subproduct with the highest cycle time. This condition represents the upper bound to the system observation time to collect events. In other cases, shorter system observation times are required, but the aforementioned condition guarantees, due to the structure of the transition purify algorithm, the same accuracy levels regardless of the physical system, its bottleneck utilisation, the WIP in queues, and the stationary state.

The automated generation of the digital models of the system in Fig. 10 has been tested with event logs extracted starting from 5 five different instants leading to different stationary states of the systems, WIP in queues, and bottleneck utilisation. The five instants in which the event log starts recording are: minute 0, minute 2500, minute 5000, minute 7500, and minute 10000. Minute 0 means the event log starts recording events with the first job entering the empty system. The other instants indicate that the event log starts recording after the indicated working period.

The processing times of all the workstations of all the system configurations from M1 to M4 consist of a deterministic time of 1 min and stochastic part following a triangular distribution with the parameters Tr(0.1, 0.3, 0.9) in minutes (mean processing time $t_e = 1.43$ min), while the arrival times of all the components into the systems follow an exponential distribution with parameters Exp(3.0) minutes (mean arrival rate $r_a = 0.33$ min). It is worth observing that even though all the workstations of all the systems have the same processing time distributions, the systems are not balanced due to the re-entrant flows and the arrival of jobs from different routing paths. Therefore, the experiment will have bottleneck utilisation rates influenced by different stationary states.

Table 2 presents the performance of the 3EM discovery algorithm in the automated generation of digital models. Column 1 refers to the system configurations from M1 to M4, shown in Fig. 10. Column 2 reports the theoretical average bottleneck utilisation for the system at a stationary state. Therefore, in the case of starting time 0, the average bottleneck utilisation will be inferior. Column 3 reports the 5 starting instants in which the event log record starts. Columns 4–6 report information regarding the duration of data recording in the event log (i.e. columns 4 and 5 indicate the width of the event log time window and the final width after applying the two recording rules of the event log, respectively) and the size of event log (i.e. column 6 reports the number of events collected in the event log). The last two columns indicate the time the algorithm requires to provide the digital model and the accuracy, that is, the fidelity with the system conceptual model.

The check mark indicates equality between the digital model obtained through the 3EM discovery algorithm $F(N, A, R, P)$ and the system conceptual model $F'(N', A', R', P')$ (i.e. the digital model has the same sets of nodes N, arcs A, routing rules R, and production rules P).

The only case of an incomplete digital model is in system M4 – starting time 0. The empty system makes the achievement of the condition quick to end the event log, that is, recording two complete traces of the job with the largest cycle time, and in this reduced amount of time, no random reworks occurred. Therefore, the event log does not report data about random reworks, and the digital model is not 100 % accurate.

The entire event-centric approach is assessed by the KPIs introduced

Table 3
The assessment of IoT infrastructure required by the Event-centric approach.

Sensor type	Quantity	Event/sensor type	Data fields/ event	Average number of events
1	3W	1	4	N.A.

Table 4
The assessment of the 3EM discovery algorithm in each of the performed experiments.

System type	Starting Time (min.)	Nodes (%)	Incoming connections (%)	Outgoing connections (%)	BoM levels (%)	Routing accuracy (%)	Computation time (seconds)	event log time window (hours)
1	0	100 %	100 %	100 %	100 %	100 %	0,01	0,14
1	2500	100 %	100 %	100 %	100 %	100 %	0,01	0,26
1	5000	100 %	100 %	100 %	100 %	100 %	0,01	0,22
1	7500	100 %	100 %	100 %	100 %	100 %	0,01	0,26
1	10,000	100 %	100 %	100 %	100 %	100 %	0,01	0,28
2	0	100 %	100 %	100 %	100 %	100 %	0,02	0,31
2	2500	100 %	100 %	100 %	100 %	100 %	1,66	32,69
2	5000	100 %	100 %	100 %	100 %	100 %	4,89	61,85
2	7500	100 %	100 %	100 %	100 %	100 %	8,92	91,07
2	10,000	100 %	100 %	100 %	100 %	100 %	19,51	120,25
3	0	100 %	100 %	100 %	100 %	100 %	0,01	0,19
3	2500	100 %	100 %	100 %	100 %	100 %	0,04	2,49
3	5000	100 %	100 %	100 %	100 %	100 %	0,05	2,08
3	7500	100 %	100 %	100 %	100 %	100 %	0,04	3,10
3	10,000	100 %	100 %	100 %	100 %	100 %	0,16	6,03
4	0	100 %	97 %	97 %	100 %	92 %	0,02	0,29
4	2500	100 %	100 %	100 %	100 %	100 %	0,05	1,77
4	5000	100 %	100 %	100 %	100 %	100 %	0,12	4,07
4	7500	100 %	100 %	100 %	100 %	100 %	0,22	5,70
4	10,000	100 %	100 %	100 %	100 %	100 %	0,19	5,11

in Section 6 and grouped in (i) IoT infrastructure assessment and (ii) KPIs of the 3EM discovery algorithm. The IoT and discovery algorithm KPIs are shown in Tables 3 and 4, respectively.

The IoT infrastructure changes from one experiment to another as the number of sensors changes according to the number of workstations (W), considering 3 sensors per workstation. In this case, the average number of events cannot be determined a priori; therefore, the KPI is not applicable (N.A.).

The discovery algorithm is assessed in each performed experiment since they involve 4 different systems under different stationary states. Generally, given a system and the scope of the discovery algorithm to identify whether the algorithm is optimised for systems in a stationary state or still unloaded, only one row of KPIs is sufficient to compare the performance of different discovery algorithms.

Except for system 4, with the event log time window starting at minute 0, all the other experiments model the physical system with a 100 % accuracy level. The percentage inferior to 100 % indicates the percentages of missing arcs and routing policy caused by the not intercepted events of the random reworks.

7. Discussion

The novel process mining event-centric approach is particularly suitable for systems in which the data collection points (points of event detection) are predictable and meaningful for the automated generation of the digital model, such as manufacturing systems. Different dispositions of the shop floor sensors can change information regarding the system behaviour, queue fillings, routing rules, and overall performance. In particular, the disposition of the sensors proposed in this paper (three sensors per workstation to detect new job arrival in the queue, start and end of the job processing) is versatile to identify the general behaviour of the systems.

The generated digital models can accurately represent assembly and disassembly or split activities, random reworks, and re-entrant flows in the same workstation.

This disposition of the sensors requires 3 sensors per workstation, and the IoT structure remains consistent even in the cases of Reconfigurable Manufacturing Systems (i.e. shop floor reconfiguration) and Flexible Manufacturing Systems (i.e. workstations performing many different operations). Therefore, it is applicable not only for manufacturing lines but also in cases of general systems with multi-product routings.

The proposed set of KPIs highlights a critical issue for the automated generation of digital models: the algorithm computation time to derive the digital model from the event log and the event log time window in which the physical system is observed to collect events.

The 3EM discovery algorithm requires a few seconds of computation time (also in the cases of event logs with hundreds of events) to provide an accurate digital model for systems with random reworks, re-entrant flows, and multi-products in the presence of assembly and disassembly processes. Moreover, unlike the other algorithms in the literature, no user intervention is required, enhancing the opportunity to provide system simulation services, integrating the algorithm in more extended tools like digital twins, and supporting other areas of production planning, such as production re-scheduling.

The 3EM discovery algorithm can obtain sufficient data to provide digital models by collecting information from the physical system for a time window lasting less than a height-hour shift, even in cases of high bottleneck utilisation, multi-products, and complex product routing paths.

Finally, to the author's knowledge, this algorithm is the first to be tested with unloaded systems and systems in stationary states, showing its performance and accuracy in both situations that extend the potential fields of application. For example, some fields could include benchmarking system performance, comparing different behaviours of the same system under different load conditions or detecting issues happening in the loading period before reaching the stationary state.

8. Conclusion

This paper proposes a novel process mining paradigm for the discovery of systems in which the position of sensors (generally, the entry points for the collection of system data) are known and meaningful for the automated generation of the system digital model. The proposed paradigm is event-centric because each detected event of the event log is mapped on one of the events of the digital model. The digital model exploits the Event relationship graph formalism. A set of KPIs to assess and compare the event-centric approaches and the discovery algorithms is proposed, and it focuses on the (i) IoT infrastructure required by the approach and (ii) the discovery algorithm used.

An event-centric approach based on the presence of 3 detection points for each workstation is proposed and coupled with a novel discovery algorithm (the 3EM discovery algorithm) to provide the automated generation of digital models of manufacturing systems. The 3 detection points per workstation populate the event log with job arrival events in the workstation queue and the start and end of job processing.

The proposed approach and the 3EM discovery model can provide the digital model of a multi-product manufacturing system in the presence of re-entrant flows, complex operations (i.e. assembly, disassembly, and split processes beyond simple job processing), and random reworks, with no human user intervention. Furthermore, its performance has been evaluated for low and high bottleneck utilisation of the system, unloaded systems, and different levels of system stationarity.

The proposed approach can be helpful for production planners and engineers. It is also implementable by small and medium enterprises with limited knowledge and resources. It is interesting for research in the fields of process mining and the automation of manufacturing systems, crucial for Industry 4.0 and 5.0 paradigms.

The event-centric approach has been tested through simulation software and requires lab-scale tests with physical sensors to reach higher TRL levels. Furthermore, further tests will be performed to evaluate its robustness when the system characteristics change.

Future research involves applying the proposed approach to identify changes in the system triggering an update of the digital model or its automated regeneration. Moreover, the IoT infrastructure is worth investigating to observe whether other sensor configurations can improve the detection of quality issues and whether they can allow evaluating sustainability performance.

CRediT authorship contribution statement

Claudio Castiglione: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Algorithms

Algorithm 1: Create nodes

Input: event log
Output: N^A, N^S, N^E, A
Definition: *Sort*(x | y) *returns vector x sorted by y*

STEP 1: Identify the set of job J;
STEP 2: Assign to each job j its set of events $\varepsilon(j)$;
STEP 3: if $\varepsilon(j)$ contains one event, create a node and connect it with job leave;
STEP 4: if $\varepsilon(j)$ contains two events, create two nodes and connect the JA with job creation;
STEP 5: Create a node for each sensor:
for *job* $j \in J$ **do**
  $i \leftarrow 1$;
  for *event* $e \in Sort(\overline{\varepsilon(j)}$ | *Event_time*) **do**
    if $e\_sensor \notin S$ **then**
      Create node n;
      $n\_sensor \leftarrow e\_sensor$;
      $n\_sensor \rightarrow S$;
      $n \rightarrow N$;
    else
      $n \leftarrow m : m \in N \wedge m\_sensor == e\_sensor$;
    if $i == 1$ **then**
      $n \rightarrow N^E$;
    else if $i == 2$ **then**
      $n \rightarrow N^A$;
    else if $i == 3$ **then**
      $n \rightarrow N^S$;
  $i \leftarrow i + 1$;
STEP 6: Connect nodes representing JA and JS events:
for $n \in N^A$ **do**
  for $m \in N^S$ **do**
    for *job* $j \in J$ **do**
      if $(\exists i \in \varepsilon(j) : i\_sensor == n\_sensor) \wedge (\exists i \in \varepsilon(j) : i\_sensor == M\_sensor)$ **then**
        Create Arc a;
        $a \leftarrow (n, m)$;
        $a -> A$;

Return: N^A, N^S, N^E, A

Algorithm 2: Purify transition effects

Input: N^A, N^S, N^E, A
Output: N^A, N^S, N^E, A
Definition: *Delete*$(n, N \mid M)$ *delete node* n *and all its arcs from set* N *by adding to the copy of* $n \in M$

STEP 1: Eliminate event node duplicates present in more than one subset:

for $n \in N$ **do**
  if $n \in N^A$ **then**
    if $n \in N^S$ **then**
      Delete$(n, N^S \mid N^A)$
    else if $n \in N^E$ **then**
      if $\exists a = (n, m) : m \in N^S$ **then**
        Delete$(n, N^E \mid N^A)$
      else
        Delete$(n, N^A \mid N^E)$
  else if $n \in N^S$ **then**
    if $n \in N^A$ **then**
      Delete$(n, N^S \mid N^A)$
    else if $n \in N^E$ **then**
      Delete$(n, N^E \mid N^S)$
  else if $n \in N^E$ **then**
    if $n \in N^A$ **then**
      if $\exists a = (n, m) : m \in N^S$ **then**
        Delete$(n, N^E \mid N^A)$
      else
        Delete$(n, N^A \mid N^E)$
    else if $n \in N^S$ **then**
      Delete$(n, N^E \mid N^S)$

Return: N^A, N^S, N^E, A

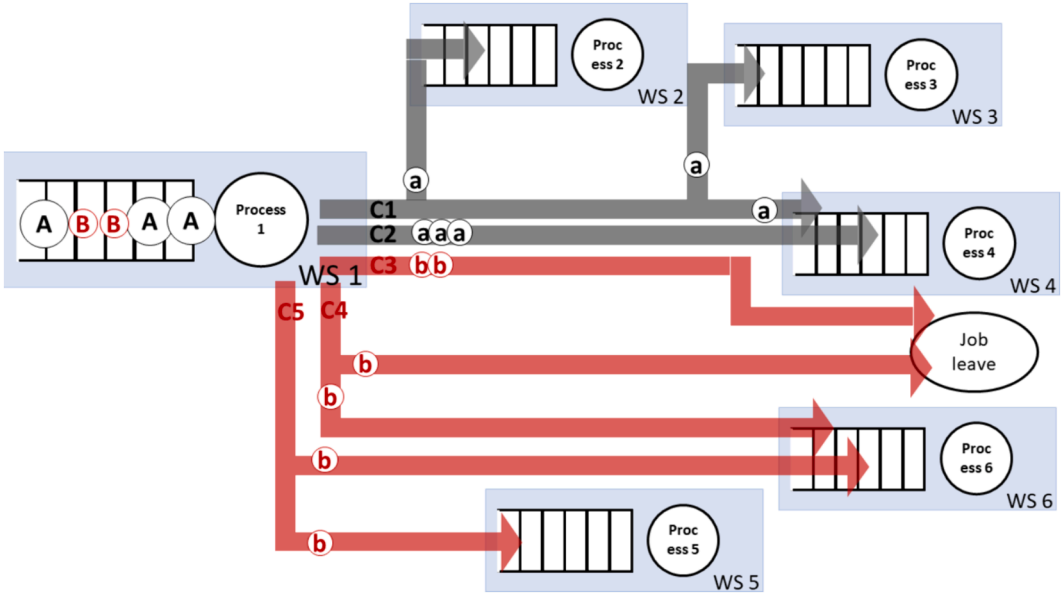
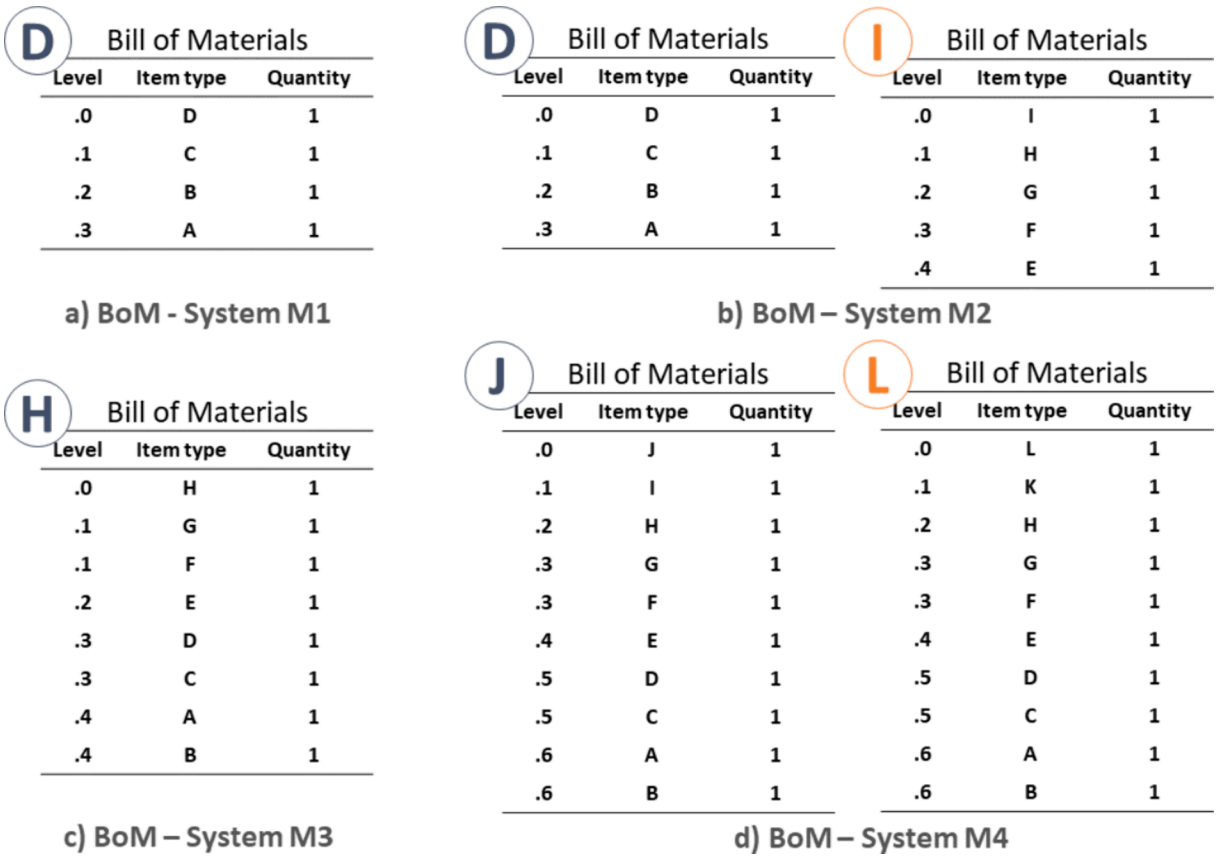


Fig. B.11. System representation of a workstation (Process 1) that, starting from the raw materials'A' and'B', feeds the downstream workstations in 5 different ways (5 routing rules identified by outgoing coloured arrows): C1 and C2 for finished products'a', and C3, C4, and C5 for finished products'b'.

Algorithm 3: Connect JS and JE events

Input: N^S, N^E, A, **event log**
Output: A
Definition: *Sort*$(x \top y)$ *returns vector x sorted by y*
Definition: $\Gamma(x)$ *returns all the events associated to node x*
Definition: *Var*(h) *returns the sum of the variances of the identified subgroups of match h*

STEP 1: Identify the potential matches between the JS and JE events:
for $\forall$ *node* $n \in N^S$ **do**
  for $\forall$ *node* $m \in N^E$ **do**
    if *card*$(\Gamma(n))$ == *card*$(\Gamma(m))$ **then**
      if $e\_time_i < f\ time_i, e_i \in Sort(\Gamma(n) \mid event\_time) \wedge f_i \in Sort(\Gamma(m) \mid event\_time), \forall i, i = 1, \ldots, card(\Gamma(n))$ **then**
        Create match $h_{n,m} = \mathrm{h}(\Gamma(n), \Gamma(m))$;
        $\underline{h}_{n,m} \rightarrow H$;
      else
        Create match $h_{n,m} = BigM$;
        $\underline{h}_{n,m} \rightarrow H$;
    else
      Create match h = $BigM$;
      $\underline{h}_{n,m} \rightarrow H$;
STEP 2: Solve the Variance Minimization Problem that provides the boolean variables $x_{n,m}$ for each match $h_{n,m}$
STEP 3: Create and save the JS-JE arcs:
for $n \in N^S$ **do**
  if $m \in N^E$ **then**
    if $x_{n,m}$ == 1 **then**
      Create arc a = a(n,m);
      Create arc a' = a'(m,n);
      a,a' $\rightarrow A$;
Return: A



D Bill of Materials

Level	Item type	Quantity
.0	D	1
.1	C	1
.2	B	1
.3	A	1

a) BoM - System M1

D Bill of Materials

Level	Item type	Quantity
.0	D	1
.1	C	1
.2	B	1
.3	A	1

I Bill of Materials

Level	Item type	Quantity
.0	I	1
.1	H	1
.2	G	1
.3	F	1
.4	E	1

b) BoM – System M2

H Bill of Materials

Level	Item type	Quantity
.0	H	1
.1	G	1
.1	F	1
.2	E	1
.3	D	1
.3	C	1
.4	A	1
.4	B	1

c) BoM – System M3

J Bill of Materials

Level	Item type	Quantity
.0	J	1
.1	I	1
.2	H	1
.3	G	1
.3	F	1
.4	E	1
.5	D	1
.5	C	1
.6	A	1
.6	B	1

L Bill of Materials

Level	Item type	Quantity
.0	L	1
.1	K	1
.2	H	1
.3	G	1
.3	F	1
.4	E	1
.5	D	1
.5	C	1
.6	A	1
.6	B	1

d) BoM – System M4

Fig. B.12. The Bill of Materials of the products of the four systems used to validate the 3EM discovery algorithm.

Appendix B. Additional figures

References

Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering, 16*, 1128–1142.

Alfieri, A., Castiglione, C., & Pastore, E. (2024). Variability propagation in manufacturing systems: The impact of the processing time distribution. *Journal of Industrial and Production Engineering*, 1–15.

Arroyo, E., Hoernicke, M., Rodríguez, P., & Fay, A. (2016). Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams. *Computers & Chemical Engineering, 92*, 112–132.

Atzori, L., Iera, A., Morabito, G., 2010. The internet of things: A survey. Computer networks 54, 2787–2805.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., & Polyvyanyy, A. (2019). Split miner: Automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems, 59*, 251–284.

Benedettini, O., & Tjahjono, B. (2009). Towards an improved tool to facilitate simulation modelling of complex manufacturing systems. *The International Journal of Advanced Manufacturing Technology, 43*, 191–199.

Bortolini, M., Galizia, F. G., & Mora, C. (2018). Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems, 49*, 93–106.

Buijs, J.C., Van Dongen, B.F., van Der Aalst, W.M., 2012. On the role of fitness, precision, generalisation and simplicity in process discovery, in: On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I, Springer. pp. 305–322.

Capocchi, L., Santucci, J. F., Pawletta, T., Folkerts, H., & Zeigler, B. P. (2020). Discrete-event simulation model generation based on activity metrics. *Simulation Modelling Practice and Theory, 103*, Article 102122.

Carmona, J., Cortadella, J., Kishinevsky, M., 2008. A region-based algorithm for discovering petri nets from event logs, in: Business Process Management: 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings 6, Springer. pp. 358–373.

Castiglione, C., Alfieri, A., & Pastore, E. (2023). Circular economy strategies at the manufacturing system scheduling level: The impacts on makespan. *Italian Manufacturing Association Conference, Materials Research Proceedings.*, 250–257.

Cecil, J., Srihari, K., & Emerson, C. (1992). A review of petri-net applications in manufacturing. *The International Journal of Advanced Manufacturing Technology, 7*, 168–177.

Chan, W. K., & Schruben, L. (2008). Optimisation models of discrete-event system dynamics. *Operations Research, 56*, 1218–1237.

Chen, Q., Lu, Y., Tam, C. S., & Poon, S. K. (2022). A multi-view framework to detect redundant activity labels for more representative event logs in process mining. *Future Internet, 14*, 181.

Conforti, R., Dumas, M., Garcìa-Bañuelos, L., & La Rosa, M. (2016). Bpmn miner: Automated discovery of bpmn process models with hierarchical structure. *Information Systems, 56*, 284–303.

Corallo, A., Lazoi, M., & Striani, F. (2020). Process mining and industrial applications: A systematic literature review. *Knowledge and Process Management, 27*, 225–233.

Dias, L. M., Pereira, G. A., Vik, P., & Oliveira, J. A. (2014). Layout and process optimisation: Using computer-aided design (cad) and simulation through an integrated systems design tool. *International Journal of Simulation and Process Modelling, 9*, 46–62.

Göppert, A., Grahn, L., Rachner, J., Grunert, D., Hort, S., & Schmitt, R. H. (2021). Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems. *Journal of Intelligent Manufacturing*, 1–20.

Günther, C. W., & Van Der Aalst, W. M. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management* (pp. 328–343). Springer.

Jeon, S. M., & Kim, G. (2016). A survey of simulation modeling techniques in production planning and control (ppc). *Production Planning & Control, 27*, 360–377.
Jeong, K. Y., Wu, L., & Hong, J. D. (2009). Idef method-based simulation model design and development. *Journal of Industrial Engineering and Management, 2*, 337–359.
Koehler, W., & Jing, Y. (2020). Trace induction for complete manufacturing process model discovery. *The International Journal of Advanced Manufacturing Technology, 110*, 29–43.
Law, A. M. (2015). *Simulation Modeling & Analysis* (5th ed.). New York, NY, USA: McGraw-Hill.
Leemans, S.J., Fahland, D., Van Der Aalst, W.M., 2013. Discovering block-structured process models from event logs-a constructive approach, in: Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceed- ings 34, Springer. pp. 311–329.
Lekićć, J., & Milićev, D. (2021). Weakly complete event logs in process mining. *Computing and Informatics, 40*, 341–367.
Leng, J., Liu, Q., Ye, S., Jing, J., Wang, Y., Zhang, C., Zhang, D., & Chen, X. (2020). Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model. *Robotics and Computer-Integrated Manufacturing, 63*, Article 101895.
Liu, C., Li, H., Zhang, S., Cheng, L., & Zeng, Q. (2022). Cross-department collaborative healthcare process model discovery from event logs. *IEEE Transactions on Automation Science and Engineering*.
Lu, Y., Min, Q., Liu, Z., & Wang, Y. (2019). An iot-enabled simulation approach for process planning and analysis: A case from engine re-manufacturing industry. *International Journal of Computer Integrated Manufacturing, 32*, 413–429.
Lugaresi, G., & Matta, A. (2021). Automated manufacturing system discovery and digital twin generation. *Journal of Manufacturing Systems, 59*, 51–66.
Lugaresi, G., & Matta, A. (2023). Automated digital twin generation of manufacturing systems with complex material flows: Graph model completion. *Computers in Industry, 151*, Article 103977.
Maggi, F.M., Bose, R.J.C., van der Aalst, W.M., 2012. Efficient discovery of understandable declarative process models from event logs, in: Advanced Information Systems Engineering: 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings 24, Springer. pp. 270–285.
Matta, A., Pedrielli, G., Alfieri, A., 2014. Event relationship graph lite: Event based modeling for simulation-optimisation of control policies in discrete event systems, in: Proceedings of the Winter Simulation Conference 2014, IEEE. pp. 3983–3994.
Mayr, M., Luftensteiner, S., & Chasparis, G. C. (2022). Abstracting process mining event logs from process-state data to monitor control-flow of industrial manufacturing processes. *Procedia Computer Science, 200*, 1442–1450.
Meincheim, A., dos Santos Garcia, C., Nievola, J.C., Scalabrin, E.E., 2017. Combining process mining with trace clustering: manufacturing shop floor Process: an applied case, in: 2017 IEEE 29th international conference on tools with artificial intelligence (ictai), IEEE. pp. 498–505.
Meng, C., Nageshwaraniyer, S. S., Maghsoudi, A., Son, Y. J., & Dessureault, S. (2013). Data-driven modeling and simulation framework for material handling systems in coal mines. *Computers & Industrial Engineering, 64*, 766–779.
Mourtzis, D. (2020). Simulation in the design and operation of manufacturing systems: State of the art and new trends. *International Journal of Production Research, 58*, 1927–1949.
Qiu, H., Chen, Y., Zhang, H., Yi, W., & Li, Y. (2023). Evolutionary digital twin model with an agent-based discrete-event simulation method. *Applied Intelligence, 53*, 6178–6194.
Ra, H. W., & Choi, S. H. (2015). Framework of a conceptual simulation model design tool. *Indian Journal of Science and Technology, 8*, 435–442.
Ramos-Gutiérrez, B., Varela-Vaca, Á. J., Galindo, J. A., Gómez-López, M. T., & Benavides, D. (2021). Discovering configuration workflows from existing logs using process mining. *Empirical Software Engineering, 26*, 1–41.
Rashid, K.M., Louis, J., 2020. Process discovery and conformance checking in modular construction using rfid and process mining, in: Construction Research Congress 2020, American Society of Civil Engineers Reston, VA. pp. 640–648.
Rinderle-Ma, S., Stertz, F., Mangler, J., Pauker, F., 2023. Process mining—discovery, conformance, and enhancement of manufacturing processes, in: Digital Transformation: Core Technologies and Emerging Top- ics from a Computer Science Perspective. Springer, pp. 363–383.
Rozinat, A., Mans, R. S., Song, M., & van der Aalst, W. M. (2009). Discovering simulation models. *Information Systems, 34*, 305–327.
dos Santos Garcia, C., Meincheim, A., Junior, E. R. F., Dallagassa, M. R., Sato, D. M. V., Carvalho, D. R., Santos, E. A. P., & Scalabrin, E. E. (2019). Process mining techniques and applications–a systematic mapping study. *Expert Systems with Applications, 133*, 260–295.
Savage, E. L., Schruben, L. W., & Yücesan, E. (2005). On the generality of event-graph models. *INFORMS Journal on Computing, 17*, 3–9.
Schlecht, M., de Guio, R., & Köbler, J. (2023). Automated generation of simulation model in context of industry 4.0. *International Journal of Modelling and Simulation*, 1–13.
Schruben, L. (1983). Simulation modeling with event graphs. *Communications of the ACM, 26*, 957–963.
Shekhar, S., Abdel-Aziz, H., Walker, M., Caglar, F., Gokhale, A., & Koutsoukos, X. (2016). A simulation as a service cloud middleware. *Annals of Telecommunications, 71*, 93–108.
Sierla, S., Azangoo, M., Rainio, K., Papakonstantinou, N., Fay, A., Honkamaa, P., & Vyatkin, V. (2022). Roadmap to semi-automatic generation of digital twins for brownfield process plants. *Journal of Industrial Information Integration, 27*, Article 100282.
Sommer, M., Stjepandíc, J., Stobrawa, S., & von Soden, M. (2023). Automated generation of digital twin for a built environment using scan and object detection as input for production planning. *Journal of Industrial Information Integration, 33*, Article 100462.
Son, Y. J., Jones, A. T., & Wysk, R. A. (2003). Component based simulation modeling from neutral component libraries. *Computers & Industrial Engineering, 45*, 141–165.
Son, Y. J., & Wysk, R. A. (2001). Automatic simulation model generation for simulation-based, real-time shop floor control. *Computers in Industry, 45*, 291–308.
Tian, F., & Voskuijl, M. (2015). Automated generation of multiphysics simulation models to support multidisciplinary design optimisation. *Advanced Engineering Informatics, 29*, 1110–1125.
Van Der Aalst, W. M., Reijers, H. A., Weijters, A. J., van Dongen, B. F., De Medeiros, A. A., Song, M., & Verbeek, H. (2007). Business process mining: An industrial application. *Information Systems, 32*, 713–732.
Weijters, A., Ribeiro, J.T.S., 2011. Flexible heuristics miner (fhm), in: 2011 IEEE symposium on computational intelligence and data mining (CIDM), IEEE. pp. 310–317.
Weijters, A. J., & Van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering, 10*, 151–162.
Wen, L., Van Der Aalst, W. M., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery, 15*, 145–180.
Wen, L., Wang, J., van der Aalst, W. M., Huang, B., & Sun, J. (2010). Mining process models with prime invisible tasks. *Data & Knowledge Engineering, 69*, 999–1021.
Yadav, A., & Jayswal, S. (2018). Modelling of flexible manufacturing system: A review. *International Journal of Production Research, 56*, 2464–2487.
Zhang, F., Song, J., Dai, Y., & Xu, J. (2020). Semiconductor wafer fabrication production planning using multi-fidelity simulation optimisation. *International Journal of Production Research, 58*, 6585–6600.