Journal of Manufacturing Systems 85 (2026) 287–306

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Technical paper

Automated extraction of comprehensive digital twin models for smart manufacturing systems

Atieh Khodadadi [a,*], Sanja Lazarova-Molnar [a,b]

[a] Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Karlsruhe, Germany
[b] The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark

ARTICLE INFO

Keywords:
Comprehensive Digital Twins
Smart Manufacturing Systems
Data-driven Simulation Model Extraction
Multi-flow Process Mining
Petri Nets
Industry 4.0
Sustainability
Energy Efficiency
Waste Management
Multi-Objective Optimization

ABSTRACT

Manufacturing systems involve multiple, often conflicting, objectives referred to as performance indicators, including production efficiency, resource utilization, energy consumption, carbon emissions, and waste reduction, which correspond to different dimensions of the system, such as time, energy consumption, and waste generation aspects. Digital Twins have emerged as a powerful tool, integrating data-driven simulation and analysis of complex systems, such as manufacturing systems. Process Mining (PM), along with data analysis, enables the automatic discovery of executable discrete-event simulation models directly from production event logs. These data-driven models are the key to enabling near-real-time Digital Twins of discrete-event systems. Stochastic Petri Nets (SPNs) offer a robust and intuitive modeling formalism well-suited for representing the extracted models derived from PM, particularly in the context of manufacturing systems. However, standard SPNs face challenges in incorporating dimensions beyond time, such as energy consumption and waste generation. This limitation often results in suboptimal decision-making and reduced system efficiency. In this paper, we propose a Comprehensive Digital Twin (CDT) framework that employs Multi-Flow Process Mining (MFPM) to automatically extract Multidimensional Stochastic Petri Nets (MDSPNs) as underlying models of manufacturing systems. To support the modeling and simulation of extracted MDSPNs, we introduce and utilize our tool, MDPySPN. The CDT framework supports multi-objective decision-making for various performance indicators of the system. Through an illustrative case study of hot forging process chains, we showcase the development of CDT for time, energy consumption, and waste generation dimensions. We further illustrate the utilization of CDT to analyze and support decision-making to enhance the case study system according to its objectives.

1. Introduction

Manufacturing is moving from single-objective optimization (such as throughput, cost) to multi-objective operation, including energy efficiency, carbon-footprint reduction, and waste minimization. This move is driven by the policy and market pressures, such as the European Climate Law [1] mandating at least 55 % GHG by 2030, and climate neutrality by 2050 [2], the 2024 revision of the Industrial Emissions Directive, and the U.S. DOE Industrial Decarbonization Roadmap [3]. These objectives reflect different system dimensions, and effective decision-making depends on understanding the system's behavior across all these dimensions. Data-driven technologies are pivotal in advancing Smart Manufacturing Systems (SMS), enabling industries to improve operational efficiency, optimize processes, and drive innovation [4]. SMS refers to an interconnected, adaptive production system that utilizes data analytics and advanced technologies to facilitate informed decisions and enable more adaptive responses to market changes and customer demands, enhancing efficiency, flexibility, and cost-effectiveness in manufacturing [5]. SMSs leverage Internet of Things (IoT) technologies, artificial intelligence (AI), and machine learning (ML) algorithms to utilize extensive datasets, enabling real-time insights, operational optimization, enhanced efficiency, and increased productivity [6]. This digitalization shift in SMSs aligns with the Industry 4.0 paradigm, which focuses on networked cyber-physical production systems that connect machines, logistics, and production resources through the Internet across the value chain [7]. Beyond decarbonization and resource efficiency, Industry 5.0 complements the Industry 4.0 paradigm by emphasizing human-centricity, sustainability, and resilience, positioning industry as a resilient provider of prosperity beyond purely technology-driven efficiency goals [8]. In manufacturing

* Corresponding author.
E-mail address: atieh.khodadadi@kit.edu (A. Khodadadi).

https://doi.org/10.1016/j.jmsy.2026.01.012
Received 11 December 2025; Received in revised form 20 January 2026; Accepted 21 January 2026
Available online 26 January 2026

systems, resilience has been defined as the capability to recover system functions after partial damage to lead to successes from failures [9].

In this context, Digital Twins (DTs) offer a dynamic platform for data-driven simulation of real-world systems in (near) real-time. DTs accurately replicate physical systems, enhancing process monitoring, analysis, and control in manufacturing systems [10]. The constituent parts of a DT consist of (1) a real-world entity, which can be an individual machine, a cluster of machines, or an entire production system; (2) a data-driven simulation model that incorporates algorithms that define the simulation logic, supported by ML and data mining techniques for extracting models from data, along with their corresponding software implementations and connectivity infrastructure, such as IoT systems; and (3) the bidirectional (near) real-time data flow that is generated by the real-world entity. Data plays a vital role in a DT, directly influencing the effectiveness of its modeling objectives [11].

Model extraction from data, as a key enabler of DTs, is essential for automatically capturing the actual behavior of complex systems without relying on manual modeling, which is often time- and cost-consuming and subject to errors, especially when systems change. Process Mining (PM) enables the automatic modeling of the underlying discrete-event behavior of these systems from event logs [12]. Event logs, which record the execution of activities over time, are the standard form of data collected from SMSs [12]. Stochastic Petri Nets (SPNs) serve as a well-suited modeling formalism for representing the extracted models from PM, as they can represent and analyze concurrency, timing, synchronization, and priority in processes [13] in discrete event systems such as SMSs. However, standard SPN formalism focuses only on the system's progression over time and cannot capture the system's behavior over other dimensions, such as energy and waste. This limitation results in an incomplete representation of the system, reducing the effectiveness of decision-making and constraining holistic optimization.

To enable the automated extraction of underlying multidimensional models of a system from its generated event logs, we developed an innovative PM methodology, termed Multi-Flow Process Mining (MFPM) [14]. MFPM extends standard PM by separately extracting process flows that correspond to different system dimensions. In our recent work [15], we proposed the Comprehensive Digital Twin (CDT) framework, where we introduced a methodology to integrate the extracted SPNs from the MFPM of different dimensions into a single Multidimensional Stochastic Petri Net (MDSPN). The final unified model captures system behavior across multiple dimensions in a single MDSPN formalism.

In this paper, we wrap up our recent research and present the complete methodology for automatically extracting MDSPNs of manufacturing systems, which serve as the foundation for developing CDTs. Furthermore, we discuss potential methodologies for enhancing decision-making in SMSs using their corresponding CDTs. Finally, through a case study on a hot forging system, we demonstrate the CDT development process and showcase how CDT supports multi-objective decision-making to improve system performance across its various operational goals.

We structured the paper as follows: In Section 2, we present the background, including an overview of DTs in SMSs, modeling and simulation within DTs, and the role of PM and relevant modeling formalisms. We further introduce MFPM and discuss formalisms capable of capturing multidimensionality. In Section 3, we introduce the concept of CDTs, detailing MDSPNs, data collection and preprocessing requirements, supporting data ontologies, and the MFPM methodology. We also describe the MDPySPN simulation tool and outline how CDTs support multi-objective decision-making. In Section 4, we present the case study on a hot forging system, including system description, data preparation, model extraction and construction, simulation setup, and validation, followed by example multi-objective decision-making scenarios. We discuss the key challenges in developing multidimensional DTs in Section 5. Finally, in Section 6, we conclude the paper with a summary of contributions and future directions.

2. Background

In the following subsections, we provide the foundational context for our research. For this, we review (i) DTs in SMSs; (ii) modelling and simulation approaches in DTs of SMSs; (iii) the role of PM in DTs, (iv) modelling formalisms for presenting the discovered model by PM; (v) multidimensional extensions of PM; and (vi) modelling formalisms capable of representing multidimensional behavior.

2.1. Terminology

To avoid ambiguities stemming from different DT- and PM-related literature, we provide descriptions of the terms that we consistently use throughout the paper. Specifically, we adopt the following definitions:

- **Data** refers to raw, time-stamped records captured within SMSs by sensing and logging infrastructures (including machine signals, controller traces, and system records) before they are organized into an analysis-ready structure [16].
- **Data model** refers to an abstract representation that defines how data is structured, organized, stored, and related within a system.
- **Information** refers to data that has been structured under a predefined data model so that it can be interpreted and queried consistently. In our paper, information corresponds to validated and structured event data organized as an event log (explained in Section 2.4.1) [17,18].
- **Knowledge** refers to actionable “know-how” extracted from recorded executions. in this paper, knowledge corresponds to an executable model. PM and data analysis are explicitly positioned as a means to extract such knowledge from event logs [16].
- **Digital** refers to information encoded in a machine-interpretable form. Following ISO 23247–1 [19], we use digital representation to mean the corresponding data element that captures a set of properties of an observable manufacturing element.
- **Digital model** refers to a computational model of a system in the digital representation that is not necessarily synchronized automatically with the physical system. In manufacturing, DT classification distinguishes between a Digital Model (no automated data exchange), a Digital Shadow (automated one-way data flow), and a DT (automated bidirectional data exchange) [20].
- **Data-driven model** refers to a digital model whose structure and parameters are inferred from observed execution data. In our work, the model structure is automatically discovered from event logs via PM and data analysis to obtain an executable simulation model [18,21].
- **Digital Twin**, as defined in ISO 23247 [19], is realized when the executable data-driven model is synchronized with the observed system through automated data exchange, thereby serving as a practical behavioral core for monitoring and what-if analysis. In manufacturing, DT classification distinguishes between a Digital Model (no automated data exchange), a Digital Shadow (automated one-way data flow), and a DT (automated bidirectional data exchange) [20]. We use “DT” in the remainder of this paper to denote the bidirectional coupled case.

2.2. Digital twins in smart manufacturing systems

Digitalization in manufacturing offers organizations the opportunity to achieve increased productivity and efficiency [22], while addressing challenges such as rapid changes in demand, heightened global competition, and the need for sustainable practices [23]. SMSs collect operational data through sensors and communication technologies. This enables the use of advanced intelligent systems to enhance their performance, such as operational efficiency, increase production rates, and reduce errors and waste in manufacturing [24]. DTs bridge these

technologies by utilizing the collected data from the system to develop a real-time virtual copy of the system, and make it possible to monitor the system continuously [25]. The term DT was first introduced by Michael Grieves in 2003 as a "digital equivalent to a physical product" [26]. Although the term DT is widely used, its meaning is not uniform across domains [20]. The "Twin" notion has also been the subject of debate, in part because any computational representation is necessarily an abstraction and cannot fully "mirror" the real system [27]. Definitional differences largely reflect whether and how the digital representation must exchange data with its physical counterpart. Some authors explicitly require bidirectional data exchange, describing it as a "two--way data flow" that allows changes in either the physical or digital entity to propagate to the other. For instance, Friederich et al. detailed a DT framework for SMSs where they explicitly adopt a DT definition that requires bidirectional ("two-way") data flow [11].

In parallel, Bi et al. [28] have argued that the commonly used DT concept ("DT-I") can be insufficient when the objective is lifecycle-level information integration. Bi et al. define DT-I as mapping a physical object to a digital model so that it can be modeled, monitored, and controlled, but note limitations for lifecycle support under continuous change. They therefore propose the Digital Triad (DT-II) concept and the Internet of Digital Triad Things (IoDTT) as an enterprise-architecture solution for lifecycle information integration in digital manufacturing enterprises. At the same time, it is important to distinguish DT research from earlier product and lifecycle data modeling. Prior work established database-driven approaches for product data model formulation [29], formal product data representations using EXPRESS/EXPRESS-G and extensions to represent imprecision and uncertainty [30], and database design strategies for engineering/product databases, including the Instance-As-Type (IAT) dependency and methods to control it to avoid inconsistency [31]. In contrast, the DT pipeline proposed in this paper targets operational, process-level behavior. We automatically derive an executable discrete-event model from runtime event logs and use it for simulation-based what-if analysis and decision support, rather than redefining product or lifecycle information models.

As shown in Fig. 1, developing a data-driven DT begins by identifying the relevant entities (machines, lines, control elements) and setting up pipelines that continuously collect diagnostics, logs, traces, and operator inputs from sensors and the factory network. The data collected is integrated and stored as structured event logs (information) in a consistent format. Because model accuracy depends on data quality, the next step is data validation: running validity checks, filtering noise, handling missing values, and applying anomaly-detection methods (supervised or unsupervised) to flag deviations from normal operation [33]. The workflow then performs knowledge extraction: (i) event detection and labeling using ML methods such as classification [34] or clustering [35], with domain experts verifying the labels; and (ii) process discovery from event logs to derive an initial, unified simulation model [36]. This model is connected to live data streams so that it updates automatically as the system changes. The twin is validated continuously by comparing its outputs with predefined Key Performance Indicators (KPIs) (such as throughput rate, output counts). After validation, we run simulations and what-if analyses to identify improvements, and we compare the outcomes with the same KPIs to provide actionable guidance for optimizing the SMS.

Despite rapid progress, most DT deployments in SMSs are still limited to one or two systems' objectives (such as time and energy) rather than multidimensional twins that jointly couple different system dimensions into a single coherent model. Our focus in this paper is operational executable behavior modeling from the system's event data to support decision-making, rather than lifecycle-wide enterprise information integration in the DT-II [28] sense.

2.3. Modeling and simulation within digital twins of manufacturing systems

Simulation is the behavior replica of a real-world process or system over time by generating and analyzing an artificial history using a representative model [37]. In the context of Digital Twins, simulation forms the computational backbone that enables prediction, optimization, and informed decision support. Simulation studies are typically categorized into four approaches: System Dynamics (SD), Discrete-Event Simulation (DES), Agent-Based Simulation (ABS), and Hybrid Simulation (HS). Each approach has distinct strengths and application areas, providing unique insights into systems. Following, we detail each simulation paradigm for DT models and review recent research contributions of DTs in SMS for each approach. SD captures the behavior of complex systems and represents systems where state variables change continuously over time using differential equations. SD is ideal for analyzing physical processes, fluid dynamics, and engineering systems [38]. DES models the systems as sequences of instantaneous events, each event triggering a discrete change in the system's state. DES is particularly useful for systems with distinct, individual events, such as production lines and logistics networks [39], and is extensively utilized for most production planning and control tasks [40]. ABS models systems as populations of autonomous agents (such as machines, operators, or products), each with distinct behaviors and attributes, whose decisions and interactions contribute to the system's overall behavior. The ABS method is particularly useful for studying systems where individual components operate autonomously but collectively influence the overall system performance [41]. HS integrates two or more simulation approaches, such as DES, SD, and ABS, into a single modeling environment, enabling the analysis of complex systems that exhibit both discrete and continuous behaviors. The HS approach provides a

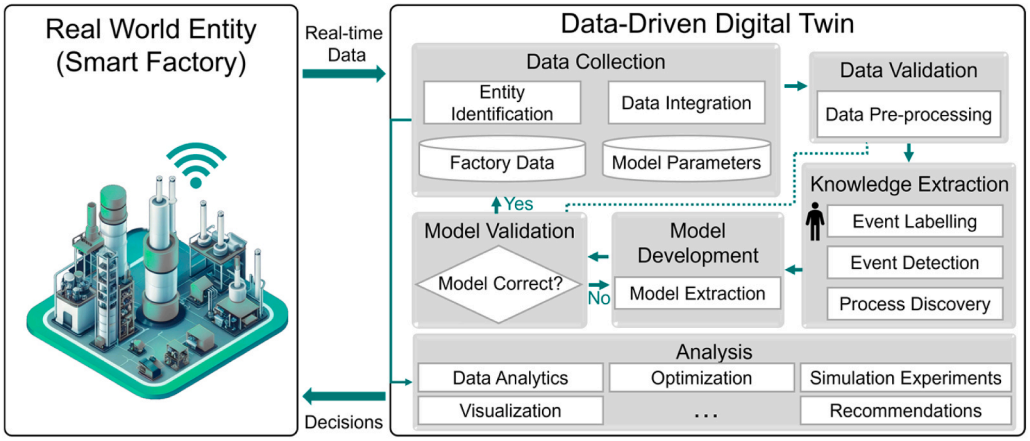

Fig. 1. Framework for data-driven Digital Twins of smart factories [32].

framework for capturing the multifaceted dynamics of real-world systems [42].

Choosing the proper simulation modeling paradigm is essential to model the system's required aspects while leaving out unnecessary details, thereby achieving a balance between oversimplification and overcomplexity in a DT. Manufacturing systems are fundamentally event-driven [43], with state transitions that occur at discrete points of time in response to events such as part arrivals, operation starts and completions, failures, and subsequent repairs. Cause of the ability of DES to accurately present the event-driven processes, it is a frequently preferred approach for developing DTs of SMSs [44]. For instance, to investigate the application of DES in DTs for SMSs, Magnanini and colleagues [45] implemented DES in a DT to support short-term, multi-product production planning in the railway sector. Magnanini et al. introduced a framework that couples a DES-based DT with a multi-objective optimization engine, enabling the generation of multiple best-possible production schedules and improving shop-floor flexibility and overall operational efficiency. In the same context, Tsinarakis et al. [46] created a Petri-net-driven DES-based DT for a steel-reinforcement plant. The DT runs in both offline and online operations, enabling dynamic what-if analyses, performance prediction, and real-time monitoring and management of industrial systems, enhancing operational efficiency and responsiveness. In this paper, we concentrate on developing DES-based DTs.

2.4. Process mining for digital twins

Many systems, including SMSs, record all pertinent events in a structured format, typically as logs, which are also referred to as event logs [47,48]. In SMSs, the events required to construct an event log are often distributed across enterprise systems—such as Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP)—and shop-floor control and sensing infrastructures, including Programmable Logic Controller (PLC) and sensor networks. Due to their heterogeneous structure, granularity, and semantics, these data records are commonly transformed, aligned, and integrated into a unified event-log format prior to process discovery [16]. PM enables the automatic extraction of the system's process model, directly from event data generated by SMSs [49]. PM consists of three categories: (i) Process discovery, (ii) Conformance checking, and (iii) Process enhancement. Process discovery is the automatic extraction of process models from the event logs. Discovery tools and algorithms, such as the alpha algorithm and heuristic miners, enable organizations to visualize and understand their operational processes in detail [18]. Conformance checking compares a process model with its event log to identify differences and determine how closely the model matches the actual process execution using four main quality dimensions: fitness, simplicity, precision, and generalization [50]. Process enhancement builds on the insights gained from discovery and conformance checking and refines the model based on specific performance or goals [51].

In this paper, we mainly focus on the process discovery techniques along with the data analysis to extract the DT's simulation model using event log data. For instance, Friederich et al. [52] showed that PM discovery techniques can be used to extract reliability models directly from the event and state logs generated by SMSs. Friederich et al. showcased their approach through a case study on a flow shop manufacturing system with parallel operations. Similarly, Bemthuis et al. [53] introduced a proof-of-concept method to extract ABS models from event log data using PM. Bemthuis et al. employed Schelling's segregation model as an illustrative example and confirmed that ABS models can be extracted utilizing PM techniques.

Furthermore, PM enables the automatic updating of the system model in response to changes in the real-world system, and keeps the DT synchronized and accurate for predictive analysis and informed decision-making [54]. In safety-critical domains such as safe autonomy, Flammini et al. [54] show that a DT equipped with PM continuously contrasts real-time traces with the reference model to flag anomalies early and trigger adaptive responses, thereby reducing downtime and enhancing operational safety. In the same context, Kumbhar et al. [55] investigate a data-driven method that combines data integration, PM, and analytics based on factory physics to identify bottlenecks. Kumbhar et al.'s approach utilizes data from multiple sources, including production planning, process execution, and asset monitoring, to generate an event log that feeds into PM. The generated DT is then used to identify bottlenecks in the system.

Standard PM methodology primarily focuses on extracting the processes and activities over time. However, relying only on the standard PM to extract a complete system model may lead to an incomplete understanding of process dynamics, especially in complex environments such as manufacturing systems, and thus hinders the potential process optimization.

Literature review shows the efforts to extend PM to discover and present system models beyond the time dimension, which are mainly termed as "multi-perspective" or "multidimensional" PM. For instance, in the healthcare domain, Vogelgesang and Appelrath introduced multidimensional PM to analyze processes across patient factors (including age, region, treatment type) [56], and later presented PMCube Explorer, an OLAP-style tool that slices logs into homogeneous sub-logs for side-by-side model comparison [57,58]. These contributions improve visibility of the developed model by partitioning logs and comparing variants rather than producing a single, integrated model. Further studies extend multi-perspective analysis in different domains. Erdoğan and Tarhan propose a goal-driven framework for emergency departments that combines activity, resource, and temporal views for performance evaluation [59]. Xu et al. develop a constraint-based, multi-perspective declarative approach inspired by openEHR for clinical modelling and conformance checking [60]. Mannhardt's thesis systematizes multi-perspective PM across control-flow, data, resource, time, and function dimensions [61].

From a data-engineering perspective, Bolt and van der Aalst formalize process cubes to isolate distinct behaviors across attribute slices, enabling comparison of performance and conformance between process variants [62] In internal logistics, Knoll et al. use multidimensional PM to support value-stream mapping and identify wasteful activities [63]. More recently, Kroeger et al. propose a data model that stores scenario-rich, multidimensional datasets for value-stream analysis in production networks [64]. Methodological Advancements also target analytics. Guzzo et al. compute holistic trace embeddings that encode multiple perspectives for clustering, classification, and anomaly detection [65], while Sim et al. automate multi-perspective process discovery using reinforcement learning [66]. These approaches enrich analysis across perspectives but do not yield formal, simulation-ready models.

Although the reviewed multi-perspective or multidimensional PM approaches improve the representation and analysis of processes from multiple perspectives, they rarely yield a formal model that comprehensively captures process behavior per dimension, such as labor effort, energy consumption, or waste generation. As a result, these PM approaches are unable to represent cross-dimensional dependencies (for example, how a delay in the time-oriented flow might lead to increased energy consumption), which remain hidden when only the time dimension is analyzed. Moreover, these approaches primarily provide descriptive analyses and comparative views, while formal, executable multidimensional models such as Petri Nets (PNs) [36] remain outside their scope. Consequently, existing approaches are not directly applicable for the simulation of multidimensional systems, which highlights the need for an extension of current PM methods. Multi-perspective or multidimensional PM approaches do not align with our conceptualization of multidimensional models, which leads us to define the term MFPM to distinguish our proposed methodology.

2.4.1. Event logs

The IoT devices continuously monitor and record various operational parameters and activities, including machine status, production level, maintenance tasks, and system anomalies. Event logs, generated by IoT devices, capture the various activities involved in a process. PM techniques provide an accurate view of actual processes of a system, pinpoint differences from predefined models, and identify inefficiencies [67]. An event log EL consists of a collection of traces, where each trace (σ) is a finite sequence of events $\sigma\epsilon\ \mathfrak{E}^*$. Formally event log is expressed as $EL = \{\sigma\,|\sigma \in \mathbb{C}\}$, with $\mathbb{C}$ representing the set of all possible traces [68]. The formal representation of each event's attributes is defined as $E_i = (t, o, e)$ where:

- *TimeStamp* $\tau(e_i)$: : The exact time of the event occurrence.
- Case *ID* $\gamma(e_i)$: A unique identifier of a trace that the event belongs to.
- *EventIdentifier* $\alpha(e_i)$: The type of activity that took place in the process instance.

The events within a trace are chronologically ordered such that for any two events e_i and e_j in the event log, if $i < j$, then $t(e_i) \leq t(e_j)$, to ensure a non-decreasing order of timestamps in a trace. The structure of event logs can be extended with additional attributes depending on the modeling needs. In Table 1, we present an example of an SMS event log that documents the sequence of events in the production process of a single product. The production process starts with a "New Order" and moves through various stages, including assembling and painting. Some events are related to a specific asset (K), which is indicated as an "Asset ID" that informs which asset was used in that activity.

2.4.2. Modeling formalisms for representing the discovered process models

PM employs different types of modeling formalisms to represent the discovered process models from event logs. These modeling formalisms include imperative models, declarative models, and hybrid models that combine both approaches [69]. Imperative models, such as PNs and Business Process Model and Notation (BPMN) [70], define the exact sequences of activities that must occur in a process. Declarative models, such as Declare rules [71], set rules on how the process can behave, without specifying every action in detail. Declarative process models are particularly suitable for less structured processes, as they define behavioral constraints rather than fixed sequences of activities [72]. In contrast, most existing process discovery techniques focus on generating imperative models, which explicitly define the control flow of the process [72]. The main types of Imperative modeling formalisms that can be automatically discovered from event logs are BPMN [73], process trees [74], Transition Systems (State Automata) [75], and PNs [36], among which, PNs are widely used formalisms for modeling and simulating manufacturing systems [76].

Since the 1980s, PNs have been extensively applied to model SMSs for purposes of analysis, performance evaluation, simulation, and control [77]. PNs offer a clear and visually intuitive way to represent systems, which facilitates understanding of systems' structure and dynamic behavior [78]. PNs are well-suited for representing the concurrent and asynchronous activities typical in manufacturing systems since they can model resource sharing, synchronization, and parallel operations. PNs have been employed in areas of manufacturing modeling, from individual machines and workstations up to entire production lines and flexible manufacturing systems [79]. PNs also support the analysis of properties in manufacturing systems, including deadlocks, state reachability [80], and throughput. Furthermore, a number of simulation tools, for instance, SNAKES [81], CPN tools [82], and PySPN [68] support the simulation of PN models, enabling performance evaluation and what-if analysis in SMSs.

Table 1
Event log example for a smart manufacturing system.

Time Stamp	Case ID	Event	Asset ID
…	…	…	…
08:00:00	112	New Order	NAN
08:01:00	112	Product Assembling	Robot A1
08:30:00	113	New Order	NAN
08:45:00	112	Product Painting	Robot P1
09:10:00	110	Product Finish	NAN
…	…	…	…

2.4.3. Stochastic petri nets

PNs are categorized into several types, each utilized for specific modeling requirements and application domains. The classical PN (also known as the Place/Transition (P/T) net) encompasses several extensions that differ mainly in how they represent tokens, handle timing, and manage system complexity. Including Fuzzy Petri Nets (FPNs) [83] that integrate fuzzy logic into PNs to address uncertainty in the modeling process, Colored Petri Nets (CPNs) [84] that extend Petri nets by assigning colors or data values to the tokens, and Stochastic Petri Nets (SPNs) [85] that reflect the stochastic behavior of the system activities by transitions following a probabilistic firing time.

Besides PNs' semantic extensions (such as CPNs and SPNs), manufacturing-oriented PN research also employs architectural structuring schemes (template-based model construction) to support scalability and reuse of PN models. In particular, Zhang et al. [86] propose a generic PN modeling scheme in which a model is first described as a template, and application-specific models are then established as instances of the template (schema/instance pattern), which reduces the complexity of large manufacturing nets. Similarly, Wang et al. [87] provide guidelines with templates to reduce the complexity caused by large amounts of construct instances in Petri net models of complex networks [87]. Importantly, these are model-construction or organization strategies, and once instantiated, the resulting models remain executable nets with the chosen semantics (such as SPN/timed semantics).

SPNs are particularly useful for systems where randomness is a key factor, such as reliability engineering. Formally, the class of SPNs follows the definition provided by [88] describes an SPN as $SPN = (P, T, A, G, m_0)$, where:

- $P = \{P_1, P_2, \ldots, P_m\}$ represents the set of places, illustrated as circles;
- $T = \{T_1, T_2, \ldots, T_n\}$ includes the set of transitions, each associated with its respective distribution functions or weights, represented as bars;
- $A = \{A^I \cup A^O \cup A^H\}$ classifies the arcs, with $A^O = \{a_1^o, a_2^o, \ldots, a_k^o\}$ indicating output arcs which connect transitions to places, $A^I = \left\{a_1^i, a_2^i, \ldots, a_j^i\right\}$ denoting input arcs that connect places to transitions, and $A^H = \{a_1^h, a_2^h, \ldots, a_i^h\}$ representing inhibitor arcs that prevent a transition from firing if a certain number of tokens are present in its connected place. Each arc has a multiplicity assigned to it, which defines the number of tokens destroyed (for input arcs) or produced (for output arcs) during the firing of a transition, and $A^O \subseteq (T \times P \times N)$ and $A^H, A^I \subseteq (P \times T \times N)$;
- $G = \{g_1, g_2, \ldots, g_r\}$ represents the set of guard functions linked with different transitions, which introduce logical conditions that further restrict transition firing based on the system state;
- and m_0 indicates the initial marking that specifies the initial token distribution across the places.

To support the integration of SPNs with PM, we define a transition as a tuple $T_i = (e, f, type)$, where e corresponds to the event label derived from event $\{E.e\}$ in a trace $E \in EL$, f represents either the probability distribution function for timed transitions or a firing weight for immediate transitions, and *type*, indicates whether the transition is *timed* or *immediate*. A transition is enabled if (1) all its input places contain a sufficient number of tokens as specified by the arc multiplicities, (2) no

inhibitor arc is actively blocking it, and (3) any associated guard functions evaluate to true. When a transition fires, it consumes (destructive) tokens from its input places and produces tokens in its output places according to arc multiplicities. This operation updates the marking of the PN from the marking M to M'. In this research, we utilize an extended version of SPNs to present the CDT simulation models.

In Fig. 2, we illustrate four foundational control-flow patterns among several commonly used in workflow modeling. PN(a) presents a sequential flow $(a \rightarrow b)$, PN(b) shows an XOR-split pattern ($a > b, a > c$ and $b \# c$), PN(c) presents an AND-split pattern ($a > b, a > c$ and $b || c$), and PN(d) represents an AND-join pattern ($a || b \rightarrow c$). More details are provided in the work of van der Aalst et al. [18].

2.5. Modeling formalisms capable of capturing multidimensionality

In multidimensional discrete-event models, the occurrence of an event in the time dimension impacts simultaneously across various system dimensions, including cost, energy consumption, and waste generation. With this improved definition of events in discrete event systems, multidimensional DES modeling can better understand system dynamics and capture the different dimension-related behavior of complex discrete event systems such as SMSs. Several modeling formalisms can represent multiple-dimensional systems in a single model. For instance, CPNs [84], as an extension of Petri nets, are enriched with colored tokens, which enable the representation of multiple types of entities or attributes concurrently in one model. Similarly, hybrid modeling combines discrete and continuous components into a single model, where some places or transitions represent continuous quantities (such as energy or fluid levels) and others capture discrete events (such as machine operations). Discrete Event System Specification (DEVS) [90] is a formal, modular framework used to model hierarchical discrete-event and hybrid systems, which can be used to present the multidimensional models. DEVS can define multiple interacting atomic models, each with its own state transitions. BPMN can also be extended to support annotations that describe additional aspects of a process, such as energy, cost, or emissions.

While the mentioned modeling frameworks contribute significantly to multidimensional modeling, they each present specific limitations when applied to complex systems such as manufacturing systems. Although CPNs are powerful in encoding multiple attributes, they have limited support for automatic extraction from basic event logs [91], and often require manual construction when modeling complex, multidimensional systems. This restricts their scalability and real-time adaptability. Hybrid models typically require detailed physical mathematical equations and as well as expert-driven modeling, making them less practical for plug-and-play DT systems. DEVS models, though modular and mathematically rigorous, are not natively compatible with data-driven approaches and often require manual setup [92]. Similarly, BPMN extensions remain mostly illustrative, lacking formal semantics for dynamic simulation or performance prediction across dimensions. BPMN is not directly executable, and manual efforts are involved in making (standard) BPMN models executable [93]. In contrast, our proposed multidimensional modeling, MDSPN, can be automatically discovered by employing PM and data analysis methodologies and enables an automatic data-driven approach, proper for CDTs of SMSs. MDSPN represents the system's behavior in various dimensions in an intuitive, unified model.

2.6. Related work on simulation of multidimensional models

Several tools facilitate the simulation of discrete-event systems with multidimensional impacts and support analysis of system behavior across multiple dimensions, such as time, energy consumption, and waste generation. Notable examples are AnyLogic® [94], ExtendSim® [95], and MATLAB®/Simulink® [96]. Currently, our library, MDPySPN [97], is the only open-source tool that provides simulation of MDSPNs, where each event in the system can impact multiple dimensions simultaneously.

3. Comprehensive digital twins of smart manufacturing systems

To facilitate the optimized multi-objective decision-making in complex systems, such as SMSs, we require a thorough understanding of system behaviors and their interdependence across their related dimensions. CDTs enable data-driven, multidimensional modeling and simulation of complex systems, thereby supporting and enhancing multi-objective decision-making.

As illustrated in Fig. 3, which is organized for direct comparison with the baseline DT workflow from Fig. 1, CDTs retain the main DT components for SMSs while introducing the following key modifications: (i) data collection and preprocessing, where MELs are used instead of conventional event logs to represent both operational events and dimension-specific measurements; (ii) process discovery and data analysis, where MFPM and dimension-aware feature extraction are applied to extract a set of unidimensional SPN models, one for each dimension of interest, rather than a time-oriented SPN only; (iii) a unification step that integrates the extracted unidimensional SPNs into a unified MDSPN; and (iv) MDSPN-based simulation, validation, and multi-objective decision-making, where model validation and what-if analysis are conducted using dimension-specific KPIs to support multi-objective improvements in parallel. Depending on the system's objectives and the dimensions under study, IoT devices capture a variety of dimension-specific data. For example, if the objective is to improve both productivity and energy efficiency, the collected data must capture process execution details (such as activity timing) alongside energy consumption across all energy-consuming stages. The subsequent subsections detail MEL construction, MFPM-based model extraction and unification, and KPI-based validation and multi-objective decision support.

3.1. Multidimensional stochastic petri nets

As we noted in Section 2.3.2, SPNs are a suitable formalism for representing models discovered through PM and can serve as the DT model for discrete-event systems such as SMSs. Current SPN (or, in general, PN) formalisms are limited to representing the system behavior in only one dimension, typically time, and cannot mirror the system's multidimensional behavior. To address this limitation, we extend the SPNs formalism to MDSPNs.

MDSPNs result from the integration of multiple unidimensional SPN

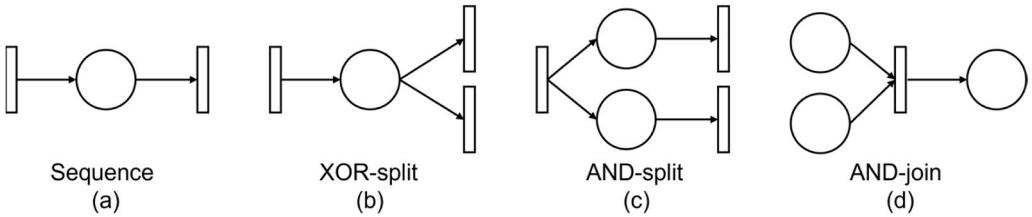

Fig. 2. Examples of foundational Petri net control-flow patterns [89].

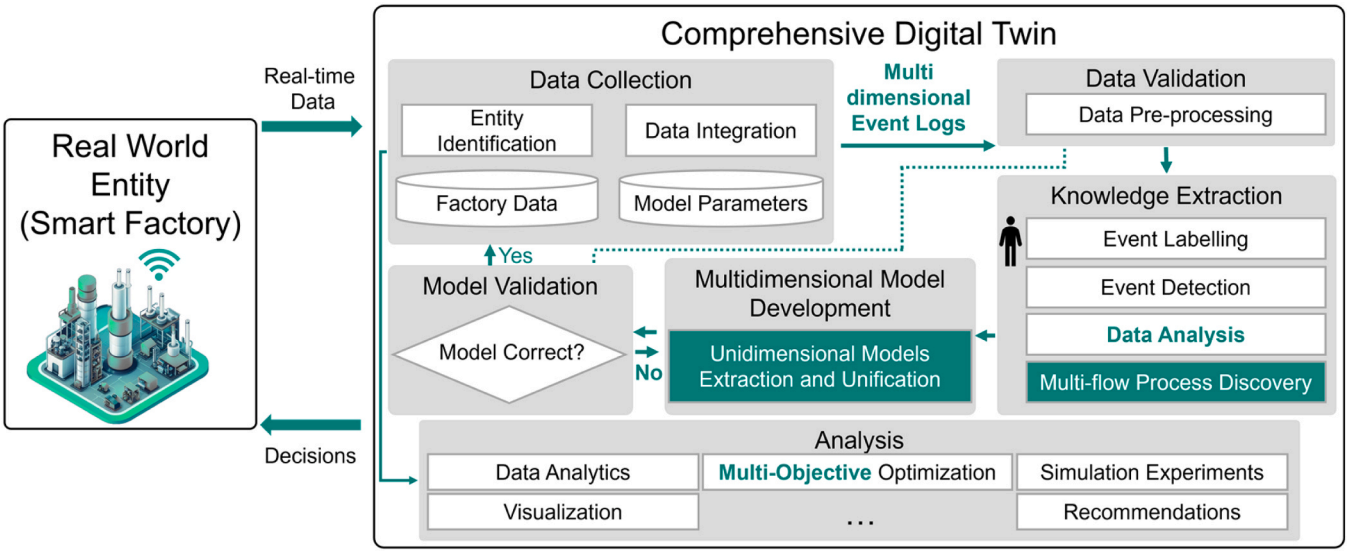

Fig. 3. Comprehensive Digital Twin framework for smart manufacturing systems.

models, each representing the system's behavior in a single dimension of interest. These unidimensional models can also function independently to support system behavior analysis of that specific dimension. All of the system's unidimensional models follow the same processing flow, with differences mainly in their transition type and impact. For instance, consider a small manufacturing system with a robotic arm that assists in production. The systems' objectives are to increase productivity and energy efficiency, and to reduce material waste, which aligns with the dimensions of time, energy consumption, and waste generation. In this scenario, when raw materials arrive, the robotic arm starts operating and completes the task over a duration of time. During the production activity, the robotic arm consumes energy and generates waste. When the production is completed, the product undergoes a manual packaging process and, following a short delay and generation of material waste, proceeds to the next stage. When no raw materials are available for the robotic arm, it will enter to idle state, during which it consumes energy until the next task arrives. This example highlights how a system shows distinct dimension-dependent behavior.

As shown on the left side of Fig. 4, the unidimensional models of this system share a similar process flow but differ in their transition type, depending on their dimension. For instance, in the time dimension, transition T1 (new material arrival) is time-consuming, while in the energy (E1) and waste (W1) dimensions, the new material arrival activity does not contribute to energy consumption or waste generation dimensions. Similarly, T2 is an immediate transition in the time dimension, which means the robot moves directly to production once raw materials are available. However, the corresponding transition E2 consumes energy in the energy consumption dimension, which indicates the robotic arm's energy consumption in the idle state. Transitions T3, E3, and W3 represent the production activity and all contribute to their related dimension (T3 takes time, E3 consumes energy, and W3 generates waste).

MDSPN represents all the different behaviors of an activity in different dimensions in a single transition, and this transition can impact from none to all identified dimensions. In MDSPNs, transitions create and destroy tokens and update simulation clocks across different dimensions, such as time, energy consumption, and waste generation. Unlike the time dimension, which progresses only in a forward

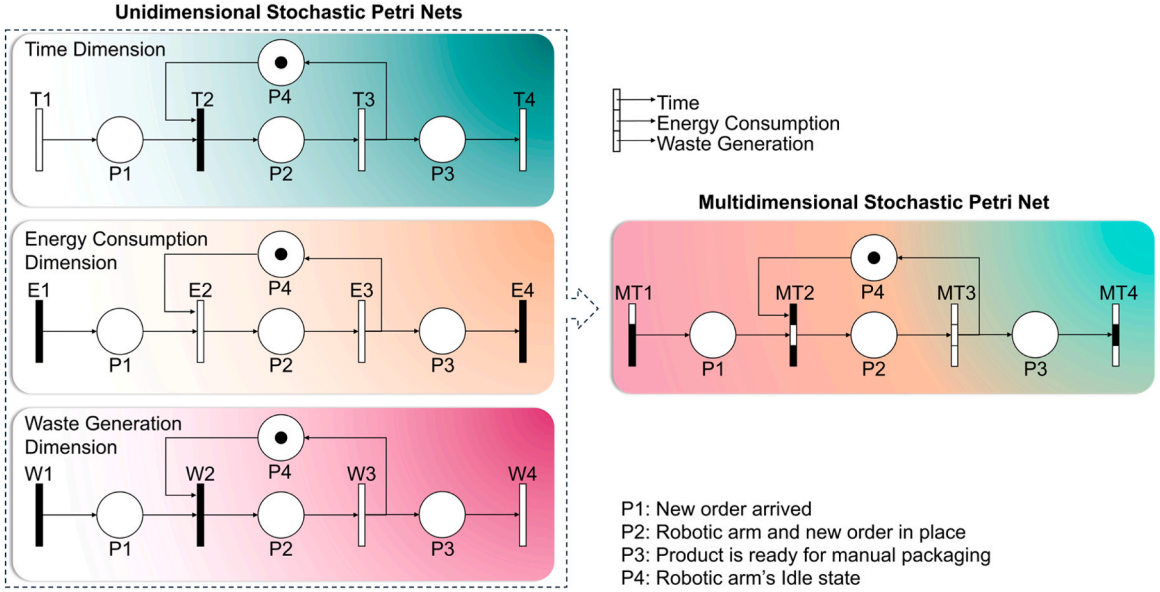

Fig. 4. Multidimensional stochastic Petri net model of the example system.

direction, updates in other dimensions can be either positive or negative. This reflects the fact that some activities may increase or reduce values in a given dimension, for example, an activity in a system may reuse the waste generated in another state and so reduce the system's waste generation indicator.

In MDSPN, we assume identical topologies across dimensions without losing generalizability. This is possible because transitions can vary in behavior, for example, they may be immediate or, as defined in our SPN extension, noncontributing in certain dimensions. To support modeling across multiple dimensions, we extended the original SPN formalism with the following enhancements:

- **Transition Definition:** Transitions are redefined to include the dimension' specific behavior, expressed as $T_i = (\text{dim}, type, F)$, where dim indicates the affected dimension, and $type \in \{\text{contributing}, \text{noncontributing}\}$. For contributing transitions, F is defined as a fixed value or a probability distribution, indicating the expected change in that dimension. Except in the time dimension, the impact function F for contributing transitions can represent either a positive or negative effect. For noncontributing transitions, F typically holds a value to show the weight of probability in fork situations.
- **Graphical Element:** We introduced a new graphical representation for multidimensional transitions, visually segmented to reflect their effects on each relevant dimension. For example, in the above example system, transitions are divided into three (the number of the system's dimensions). The top segment of the transition shows time behavior; the middle shows energy consumption; and the bottom shows waste generation.
- **Distinct Clocks:** We defined separate simulation clocks for each dimension. The time clock functions as a conventional simulation clock for the time progress, while clocks for other dimensions update according to changes in their respective indicators.

In Fig. 4 (right), we proceed by integrating each unidimensional model into a single multidimensional model (MDSPN). In MDSPN, tokens follow their progress in the time dimension, while transitions associated with other dimensions update the corresponding values, which are tracked by their own clocks. For instance, MT1 only contributes to the time dimension, and MT2 to the energy consumption dimension, while MT3 contributes to all dimensions, and MT4 time and waste generation dimensions. This example demonstrates how MDSPNs simultaneously capture both the temporal progression of process steps and their impacts on other dimensions.

3.2. Data collection and preprocessing for comprehensive digital twins

Building upon our previous research [98], to enable the extraction of MDSPN models for CDTs, we initially expand the structure of event logs, termed Multidimensional Event Logs (MEL). This expansion includes capturing basic event log data, such as "Time Stamp (t)," "Case ID (o)," and "Event Identifier (e)," as well as additional data points that reflect the involved dimension's related behaviors, such as energy consumption, carbon footprint, and waste generation-related data at the occurrence of each event. We formalize the MEL as follows:

$$MEL = \{E_1, E_2, \ldots, E_m\},$$

where:

$$E_i = (t, o, e, d).$$

Each E_i records the data points required for the CDTs model extraction. In this context, d represents the set of domain-specific attributes that capture related-dimensional impacts associated with each event, such as energy consumption, carbon emissions, and waste generation resulting from the event's occurrence. For the preprocessing of MELs, it is essential to remove incomplete or unfinished traces to prevent the discovery of inaccurate or misleading process models. To enable the automatic extraction of multidimensional models, we use a uniform, coherent structure for the MEL, including consistent column labeling for all generated logs. The key components are as follows:

- Asset (K): A unique identifier for each asset (such as a labor-intensive, machine, or device) or group of assets that specifies which resource is involved in an activity.
- Dimension Stamps: In addition to timestamps, MEL also includes the dimension stamp that records the value of each relevant dimension at the event occurrence. For example, the log entry for the production event with an asset connected to the grid includes an energy stamp.
- Resource Types: For the dimension with different types of resources, we categorize resources accordingly. For instance, a system that consumes different types of energy sources, such as electricity, battery, or fuel, or generates different types of waste, including plastic, metal.
- Activity Begin/End Events: Each activity is represented by two events, a start (Begin) and a completion (End). The time between Begin and End directly points to task duration, enabling the extraction of the transitions' contributing values, such as time distribution, energy consumption per unit time, and waste per operation. Additionally, comparing an activity's end with its successor's start reveals bottlenecks and inefficiencies between them.

Notably, a uniform naming must be applied across all MELs, including the main system log and asset-specific logs, to enable the automatic CDT model extraction. In Table 2, we illustrate an example of an MEL from the example system in Section 3.1, with its corresponding dimensions as energy consumption, carbon footprint, and waste generation. Each entry in the MEL records the main attributes, including time, event, asset, ID, and other dimension attributes (if contributed) such as energy usage in kilowatts/hours, the type of power used, and the type and quantity of waste generated in kilograms. It is noted that some entries might not contain data for certain dimensions, which indicates these dimensions do not pertain to specific activities, for instance, manual packaging does not consume energy.

In addition, we collect individual MELs for each asset K, defining asset-oriented activities. These logs combine the concept of state logs into the event logs and capture both state transitions and their corresponding impacts in all related dimensions. Consistent and unique naming of assets in system MELs is essential to enable the automatic model extraction and accurate representation of asset behavior in the model. For instance, Table 3 presents an excerpt of a single electrical asset MEL. Events such as "Idle Begin" and "Idle End" denote asset state change and their impact on the related dimensions.

Table 2
Multidimensional event log excerpt of an example manufacturing system.

Time Stamp	ID	Event	Asset	Energy Stamp (kWh)	Energy Type	Waste Stamp (kg)	Waste Type
08:00:00	1001	New Order	None	NA	NA	NA	NA
08:01:00	1001	Robot and Order in Place	Robot 1	0.11	Grid	NA	NA
08:01:26	1001	Production Begin	Robot 1	0.11	Grid	0.2	Plastic
08:09:00	1001	Production End	Robot 1	1.27	Grid	0.8	Plastic
08:09:00	1001	Manual Packaging Begin	Robot 1	NA	NA	0.8	Plastic
…	…	…	…	…	…	…	…

Table 3
Multidimensional event log excerpt of an example system asset.

Time Stamp	ID	Event	Energy Stamp (kWh)	Power Type	Waste Stamp (kg)	Waste Type
09:00:00	100	Idle Begin	0.11	Grid	NA	NA
09:01:00	100	Idle End	0.23	Grid	NA	NA
09:01:26	101	Active Begin	0.23	Grid	0.2	Plastic
09:09:00	101	Active End	1.27	Grid	0.8	Plastic
09:09:00	102	Idle Begin	1.27	Grid	NA	NA
…	…	…	…	…	…	…

The structure of MELs is not limited to sustainability dimensions (such as energy, waste) and can also be extended to capture resilience-relevant operational dimensions. In particular, resilience can be represented in the MEL by adding explicit disruption and recovery events (such as fault start/end, repair start/end, maintenance, rework, rerouting decisions), and by recording recovery actions (such as rework or rerouting) as event attributes, enabling resilience analysis in CDTs [87,99].

3.3. Multi-flow process mining

In MFPM, we apply the standard PM techniques to the MEL (the system's MEL, as shown in Table 2, and when required for assets' behavior, the asset MELs in Table 3) to extract separate unidimensional models for each dimension of interest. As we illustrated in Fig. 5, the MFPM workflow is divided into two main phases. The first phase results in a distinct unidimensional SPN model for each relevant dimension (such as time, energy, or waste) using system MEL. For this, we extract the main process flow structure of the target system, which is the same for all dimensions. Furthermore, we identify and map the sets of places (P) and transitions (T) following the PN formalism. Furthermore, we define the set of arcs (A) that connect places and transitions, specify guard functions (G), assign probabilities for fork situations, and define the initial marking (m_0). The second phase identifies transition attributes in the unidimensional SPN models from system MEL. For this, for each contributing transition we specify, in its related dimension, a contribution (impact) given by a distribution, rate, or fixed value. For example, we define the time distributions for timed transitions or energy consumption rates in the energy dimension. The output of the MFPM is a collection of unidimensional SPN models (U), where each SPN model reflects the behavior of the system in each dimension of interest. We next detail automatic model extraction via MFPM.

3.4. Automated MDSPN model extraction

In this subsection, we present the automated extraction process of MDSPN models for representing CDTs in complex discrete-event systems such as SMSs. As we illustrated in Fig. 6, first, we apply process discovery techniques to the cleaned MELs to discover the system's process flow. Subsequently, we identify key structural features of the extracted

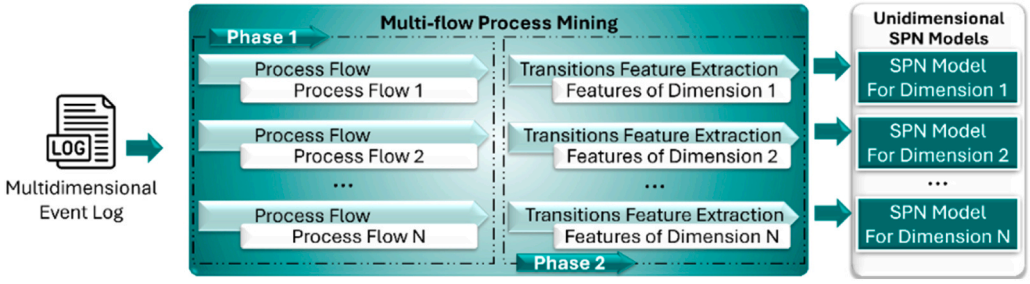

Fig. 5. Multi-flow process mining framework [14].

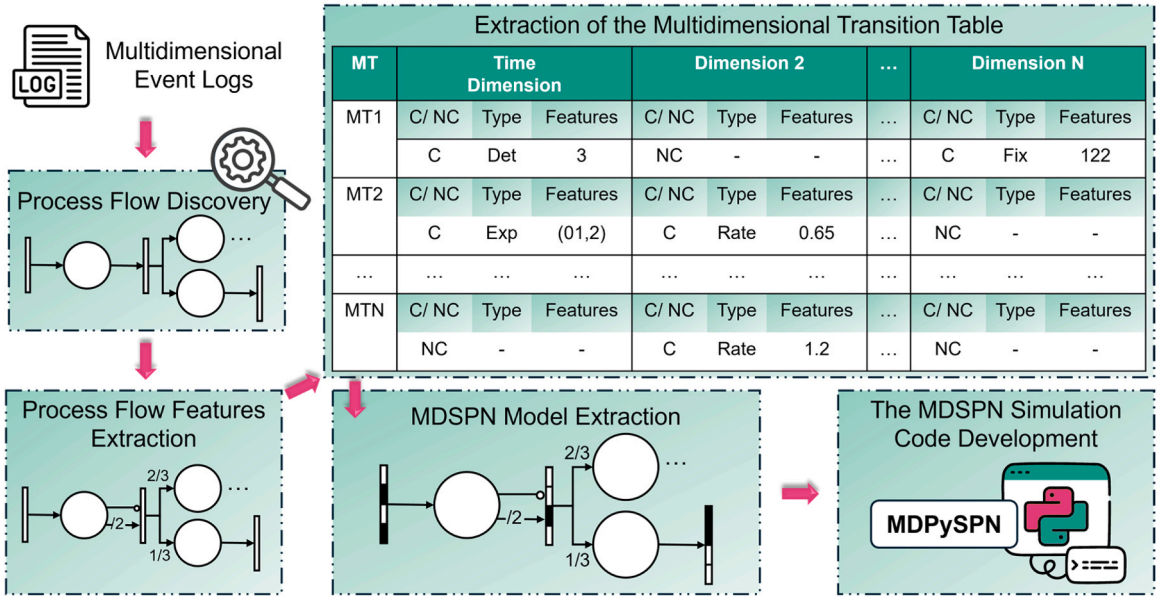

Fig. 6. The process of automatic MDSPN model extraction.

process, including inhibitor arcs, arc multiplicities, and branching probabilities in fork situations. Based on this information, we construct a Multidimensional Transition Table (MTT) that captures the behavior of each transition across different dimensions of the system. MTT specifies, for each transition, its contribution to every dimension, its type (fixed, rate, or distribution type, such as deterministic, exponential, etc.), and its impact feature, including rate values or distribution parameters. Based on the transitions table, we can either extract a unidimensional model of the system corresponding to a dimension of interest or construct the MDSPN model with all its detailed characteristics. Finally, we generate the corresponding simulation code for the extracted MDSPN model, which is implemented using the MDPySPN simulation library. It is important to note that the time dimension must be included when simulating the MDSPN model, as the progression of the simulation depends on the advancement of time.

The MDSPN extraction pipeline comprises two core algorithms (Algorithms 1–2) followed by an MDSPN construction (Section 3.4.3). For readability, we first summarize the core innovations of Algorithms 1–2 and then detail them in Sections 3.4.1–3.4.3. Algorithm 1 describes the extraction of the structural SPN skeleton from the MEL. Algorithm 2 introduces MTT as a unified transition-level parameterization that assigns each transition a dimension-specific contribution form and estimated parameters, enabling the construction of unidimensional SPNs and their unification into an executable MDSPN model.

3.4.1. *Process flow discovery and process flow features extraction*

In Algorithm 1, we outline the first step of automatic MDSPN model extraction, the system's process flow discovery, defined as $F_d = (P_d, T_d, A_d)$, where P_d and T_d represent the set of places and transitions, and A_d represents the arcs. In the following, we detail Algorithm 1, which is an extended version of our earlier work for automatic SPN model extraction [89].

1. **Integration of All Collected *MEL* Data**: We collect separate MELs for multiple production runs and store each log independently (such as in distinct database tables). With this separation, we ensure that the MDSPN feature extraction process can use the variability across runs, such as parameter estimation and pattern discovery.
2. **Trace Extraction and Data Pre-processing:** We first sort the input $MEL = \{E_1, E_2, \ldots, E_n\}$ by their case IDs (o) and timestamps (t). Next, we utilize PM to extract the underlying process flow of the system and keep the unique trace paths in the trace list (TR). In parallel, we perform the same trace-extraction procedure on each asset's MEL, obtaining a separate set of traces for every asset (or asset group). To avoid the invalid process flow extraction, we drop the incomplete traces for each unique case ID before the PM. From TR we define the first events list.
3. **Petri Net Structure Construction:** In order to define the PN's structure, we utilize MEL and TR to set a transition (T_i) for each unique event (E_i). Next, we assign a place P, labeled as ($T_i\_to\_T_{i+1}$) between each unique consecutive pair of transitions (T_i, T_{i+1}), which can be identified from TR.
4. **Assets Behavior Implementation:**

From the previous steps, we obtain (i) the main process-flow model and (ii) separate models for each asset (or asset group). In this step, we integrate all asset models into the main PN so that resource behavior is explicitly represented. This integration is useful even for a standard SPN, because it ensures that constraints that might otherwise be modeled via inhibitor arcs are instead captured by explicit asset-state places and transitions. Thus, assets can participate in activities only when they are available. To connect assets to the process, we identify, for each asset K, the activities in which it is involved, and the first event that enables its active state.

For each asset K, we introduce an "Asset_Ready" transition T^K_{Ready} that represents the change of state from its current non-active state (such as idle/free) to the active state. We add an asset-available place P^K_{avail} representing this non-active state, which holds one or more tokens in the initial marking m_0 to indicate the number of available units of that asset (a single asset or a group). We then update the SPN model by inserting T^K_{Ready} before the enabler transition that marks the start of the asset's active state. The place P^K_{avail} is connected to T^K_{Ready}; T^K_{Ready} then leads to a new intermediate place P^K_{new}, which connects to the first active transition that uses asset K. Finally, we add an output arc from the last active transition using asset K back to P^K_{avail}, returning the token when the asset completes its activity. This structure accurately models asset state transitions and, in later steps, supports the integration of dimension-specific behavior. For instance, in Fig. 7, we contrast the raw SPN skeleton (a) with its extension with assets' behavior (b). In panel (a), the main control flow consists of a transition $T1$, place $P1$, and transition $T2$. Panel (b) updates the SPN model for an asset with two states: idle and active. For this, we add an idle place (*P_IDLE*) that permanently holds one or more tokens (to show one or a group of assets) in the initial marking m_0; an *Asset_Ready* transition (*T_Asset_Ready*) that when fires, signals the asset's move from idle to active state; a new place (*P_new*) inserted between *T_Asset_Ready* and the first active transition ($T2$); and a return arc from $T2$ back to *P_IDLE* Returns the token once the activity finishes, closing the asset's state cycle. Furthermore, we add initial tokens to place (*P_new*) based on the number of assets included in the activity (this can be modeled for each individual asset).

5. **Inhibitor Arcs Detection:** Inhibitor arcs (A^H) can introduce delays in the execution of associated activities by preventing transitions from firing until specific conditions are met. Using this information, we can pinpoint inhibitor arcs through a two-step procedure. First, we detect skews in the time duration of activities; second, we assess whether this skew is caused by the execution of subsequent activities (for example, when a later transition in TR is enabled by an earlier case ID). For this, we first record, for each activity's duration time, using:

$$\Delta_i = \sum_{j=1}^{m} \mathrm{t}_{\mathrm{j,i}} - \min(t_{j,\ \ i-1},\ \ t_{j\_enabled}),$$

where $\mathrm{t}_{\mathrm{j,i}}$ is the time stamp of the case ID_j for the event E_i, t_{i-1} is the time stamp of the case ID_j for the E_{i-1} and $t_{j\_enabled}$ is the time that case ID_j is included in the activity (enabled corresponding transition T_i). If $\Delta_{i,j}$ for E_i is not zero, we must check if the value for Δ_i is affected by an inhibitor condition. For this, we first drop outliers in the Δ_i durations by discarding any values outside the 15th–85th percentile range (thereby trimming extreme observations) and then compute the median activity's time duration t_{med}. For each occurrence of even E_i with *case* $ID_j = 1, 2, \ldots, m$, we then estimate the expected firing timestamp as follows:

$$t_{j\_expected} = \ \ t_{j_{enabled}} + \ \ t_{med}.$$

If for a given *case* ID_j the observed delay $\Delta_{i,j}$ of event E_i exceeds the expected delay $t_{j\_expected}$ exactly the moment when the next event E_{i+1} becomes enabled, we insert inhibitor arcs from the outgoing place to its corresponding transition T_i for the event E_i. Finally, we remove any inhibitor-induced delay from $\Delta_{i,j}$ for E_i and return $\Delta'_{i,j}$. Next, we return Δ'_i as the refined duration measure for E_i corresponding to T_i.
6. **Multiplicity Detection:** We separately define multiplicity for input and output arcs. For input multiplicity, we scan how many tokens (case IDs) each transition requires from its place when it fires. For instance, for the input arc from place P($T_i\_to\_T_{i+1}$) to T_{i+1}, corresponding to E_i and E_{i+1} we calculate the number of occurrences of E_{i+1}, before the first occurrence of the event E_i in every such trace,

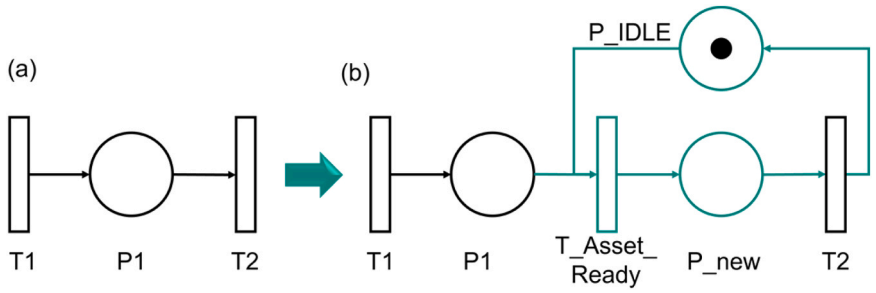

Fig. 7. Incorporating the asset states cycle into the SPN skeleton.

we assign that integer value as the multiplicity on the input arc. Similarly, for output multiplicity, we inspect every trace that contains E_i and count the number of occurrences of E_{i+1} after the first appearance of E_i, and assign the integer value as the multiplicity on the output arc from T_i to P($T_i\_to\_T_{i+1}$).

7. **Control-Flow Patterns Implementation:** To represent system progress precisely, we must implement foundational control-flow patterns for SPN and then refine the resulting places, transitions, and arcs. For instance, to implement XOR-split control-flow, we begin by identifying a transition that serves as the decision point, scanning all traces in *TR* that share an identical prefix but differ at the subsequent event (corresponding to a transition). In the resulting SPN model, we merge the transitions' multiple output places (which we assigned in step 2 of Algorithm 1) into a single place and draw separate arcs from that place to every possible following transition (to each event that appears next in the traces). To implement the AND-join pattern, we identify points where traces with different prefixes meet on the same event in TR. Although the PN generated in step 2 of Algorithm 1 can already encode an AND-join, we test whether the join point is a false AND-join. A true AND-join satisfies two conditions: (i) all tokens generated by the prior events must be present at the same time before the join transition becomes enabled, and (ii) when the transition fires, the tokens form the prior events merge into one, and the join transition creates new case ID(s) equal to the number of output multiplicity of the join transition. If the condition fails (so we have a false AND-join condition, we merge the input places to the transition with the false AND-join condition into one and update the arcs to reflect this refinement.

Algorithm 1. SPN skeleton extraction.

Input: multidimensional event log MEL

Output: *SPN skeleton* $F = (P, T, A)$

1. **Integration of All Collected *MEL* Data**
2. **Trace Extraction and Data Pre-processing**: Sort events in MEL by case ID & timestamp; remove incomplete traces, utilize the cleaned log in PM, store ordered traces in the set TR, define the first events list; apply the same trace-extraction procedure to each asset's MEL.
3. **Petri-net structure construction:** For every trace $t \in TR$, for every pair of consecutive events mapped to transitions (T_i, T_{i+1}): create a place P and add the arcs $T_i \rightarrow P_{i\_to\_i+1} \rightarrow T_{i+1}$.
4. **Asset behaviour implementation:** For each asset K: create an asset-available place $P^K_{\text{avail}}$with one or more tokens in the initial marking m_0(one per unit of the asset); create an Asset_Ready transition T^K_{Ready}that represents the change of state from the asset's current non-active state (such as idle/free) to the active state; insert T^K_{Ready}before the enabler transition that marks the start of the asset's active period by adding a new intermediate place with initial token(s) $P^K_{\text{new}}$and arcs $P^K_{\text{avail}} \rightarrow T^K_{\text{Ready}} \rightarrow P^K_{\text{new}} \rightarrow$(first active transition using asset K); add an output arc from the last active transition using asset Kback to P^K_{avail}, returning the token when the asset completes its activity.
5. **Inhibitor Arcs Detection:**
   a. For every transition T_i and each case j : compute $\Delta_i = \sum_{j=1}^{m} \text{t}_{\text{j,i}} - \min(t_{j,i-1},\ t_{j\_enabled})$.
   b. If $\Delta_i = 0$: no inhibitor check is needed.
   c. Otherwise: remove outliers outside the 15th–85th percentiles and set the median t_{med} ; compute the expected timestamp $t_{j_{expected}} = t_{j_{enabled}} + t_{med}$; If $\Delta_{i,j} > t_{j\_expected}$ and E_{i+1} is enabled: Mark inhibitor arc from the place P($T_i\_to\_T_{i+1}$) to T_i; remove the inhibitor effect on time duration for $E_{i,j}$ and return Δ'_i.
6. **Multiplicity Detection**: In every trace $E_i \rightarrow E_{i+1}$: for input arc multiplicity, assign the number of occurrences E_i before first E_{i+1} ; for output arc multiplicity, assign the number of occurrences E_{i+1} after first E_i.
7. **Control-Flow Patterns Implementation:** Refine the SPN model by adding control-flow patterns, for instance:
   a. XOR-split: locate each transition whose traces share an identical prefix but diverge at the next event; merge its outgoing places into one place, then update arcs.
   b. AND-join: verify join conditions: all incoming tokens must be present simultaneously and merge into the correct number of outputs; if not, integrate its input places into one and update arcs.

3.4.2. Extraction of the multidimensional transition table

Building on the SPN skeleton extracted in Section 3.4.1, we next define each transition's *type* $\in$ {contributing, noncontributing} and its impact in each dimension of interest by developing the MTT. Following, utilizing the MFPM framework and the MTT, we can derive a separate unidimensional model for each dimension. Because the manufacturing process progresses over time, we consider time as a mandatory baseline dimension for the simulation of the final MDSPN model. Furthermore, when the energy dimension is also under study, we extend the SPN by representing assets (K) to enable the model to capture their distinct energy-consumption profile states. In the following, we detail Algorithm 2, which presents a step-by-step procedure for constructing the MTT, using MEL and the SPN skeleton as inputs.

1. **Define Multidimensional Transition Table:** We begin by constructing an MTT, where each row corresponds to a unique event E_i (mapped to a transition T_i), including T^K_{Ready} transitions. To support the automatic MDSPN model extraction, we establish a structured naming for the columns. The first column stores the event identifier (e). For each dimension of interest, we define four additional columns: (i) *dimension_name_Dimension* specifies the dimension (such as *Energy_Dimension*), and the content of this column indicates the type of transition in that dimension (C for contributing, N for non-contributing), (ii) *dimension_transition_impact_type* describes how the transition contributes to the dimension (such as a distribution, rate, or fixed value), and (iii) *dimension_transition_impact_value* provides the corresponding impact value. The impact of transitions with contributing type is quantified using one of the following approaches:
   a. Fixed values: representing the average fixed contribution impact of an activity.
   b. Rate values: estimate the activity-specific rate:

$$r_{i,d} = \frac{1}{m}\sum_{j=1}^{m}\frac{Impact_{i,j}}{\tau_{i,j}}$$

   where $Impact_{i,d}$ is the measured impact (such as energy, cost) of the activity i on dimension d for the included case ID_j, $\tau_{i,j}$ is the corresponding duration, and m is the total number of observed cases. The mean rate can then be multiplied by the duration observed in any future execution of the activity to obtain its total impact on the dimension d.
   c. Dynamic values: calculate the value using MLE or ML techniques to calculate the impact value. We fit distributions using Maximum Likelihood Estimation (MLE) via the SciPy library [100] to a set of candidate distributions{*norm*, *lognorm*, *expon*, …}. or each distribution f, we estimate its parameters $\widehat{\theta}_T(f)$. To evaluate the goodness of fit, we perform a one-sample Kolmogorov–Smirnov (*KS*) test by comparing the empirical cumulative distribution of Δ'_T with the theoretical Cumulative Distribution Function (CDF) $F(x;\widehat{\theta}_T(f))$. We then select the distribution f^*_t with the highest KS p-value with the highest KS p-value as the best-fitting time-feature distribution.

   The impact for all dimensions (except time) can be both positive and negative. For instance, consider a system that uses the waste generated in an activity (positive impact on the waste generation dimension) in another part of the system (negative impact on the waste generation dimension). Furthermore, if a dimension's type column contains more than one type, the dimension is split into separate sub-dimensions. For instance, if a system consumes different types of energy sources, such as grid and oil, each of them would be considered as a separate dimension for that system.
2. **Time Dimension**: We capture the temporal behavior in a dedicated *Time_Dimension* column and identify other related details such as weights (for immediate XOR-splits) or fitted time distributions (for timed transitions) as follows:
   a. **Time dimension definition and transition classification:** In MTT, we add the column *Time_Dimension*, where we assign each transition's type. For this, we check if both Δ_i and Δ'_i are zero, we classify the transition T_i as immediate and set ($T_i \in T_I$), and set *Time_Dimension* to *N*, otherwise it is marked as a timed transition ($T_i \in T_T$), with type *C*. All transitions in the first events list are included in the timed transition list (T_T). In addition, every Asset_Ready transition T^K_{Ready} is treated as an immediate transition ($T^K_{\text{Ready}} \in T_I$) and its Time_Dimension entry is set to N.
   b. **Weight assignment for immediate (XOR-split) transitions:** For each immediate transition ($T_i \in T_I$) involved in XOR-split conditions (identified in step 5 of Algorithm 1), we compute the weight of each of its outgoing branches by counting how often each possible successor transition follows T_i across all traces. We record these values in the *Weight* column (which we add to the multidimensional transitions table), corresponding to each successor transition.
   c. **Distribution fitting for timed transitions:** For each timed transition ($T_i \in T_T$). We further define the *dimension_transition_impact_type* as "distribution", and correspondingly, we compute the *dimension_transition_impact_value* by fitting the adjusted durations Δ'_i using MLE.
3. **Energy Dimension**: The second key dimension in the MDSPN model extraction is the energy dimension, which becomes essential when energy-related behavior is among the system's objectives, particularly in manufacturing systems, where energy consumption is coupled with asset operation. To extract the energy dimension, we proceed as follows:
   a. **Energy dimension definition and transition classification:** In the multidimensional transitions table, we define *dimension_name_Dimension* as *Energy_Dimension*. A transition is marked as contributing (*C*) under *dimension_transition_type* if its corresponding event's entry in the *Energy Stamp* column of the MEL is not missing (not NaN); otherwise, it is labeled as *N*. For contributing transitions, we specify *dimension_transition_impact_type* as *rate*, since energy consumption depends on the time the asset spends in its energy-relevant states during an activity (including processing, idle, standby, or other non-off states), where all such states contribute to the overall Energy Consumption Profile (ECP).
   b. **Energy consumption profile table construction:** To determine the *dimension_transition_impact_value*, we construct an ECP table for each energy-consuming asset K in the system. The ECP table is built from the asset-specific MEL and captures the energy consumption rates for both active and idle states. For each asset, we identify the time intervals during which it is active or idle and associate these with corresponding power consumption values. We then calculate the instantaneous energy consumption rate as defined in Step 1. The active ECP rate for each asset is added to the *dimension_transition_impact_value* for every corresponding event in the MEL where that asset is involved. Next, we add a column to the ECP table to represent the asset's state changes caused by an activity. Using this, we identify the first event in the asset's MEL where the asset becomes active (as the enabler for the active state) and the last event before the asset returns to its available state (as the enabler for the asset-ready state).
4. **Other Dimensions**: For all additional dimensions beyond time and energy, we define *dimension_name_Dimension* based on the presence of a corresponding *Dimension_stamp* in the MEL. If the value in the *Dimension_Stamp* column is not NaN for a given event, we classify the transition as contributing to that dimension by setting *dimension_name_Dimension* value to C; otherwise, it is marked as N. For contributing transitions, we specify the

dimension_transition_impact_type as Fixed, Rate, or Dynamic, depending on the semantics of the dimension and as defined in Step 1 of Algorithm 2.

Algorithm 2. Multidimensional transition table development.

Input: *multidimensional event log MEL, SPN skeleton* $F = (P, T, A)$

Output: *Multidimensional Transition Table (MTT)*

1. **Define MTT**: Initialize one MTT with one row per unique event E_i → transition T_i; set the first column as event identifier e; for every dimension D of interest: add columns $D\_Dimension, D\_transition\_impact\_type$, and $D\_transition\_impact\_value$.
2. **Time Dimension:**
   a. For each T_i: if Δ_i and Δ'_i =0; then $Time\_Dimension = N$ and $T_i \in T_I$; else $Time\_Dimension = C$ and $T_i \in T_T$; Mark all transitions in the first events list as $T_i \in T_T$. Additionally, each Asset_Ready transition T^K_{Ready} is treated as an immediate transition ($T^K_{\text{Ready}} \in T_I$) with Time_Dimension = N.
   b. For every $T_i \in T_I$ in an XOR-split: count the successors that follow T_i and store in their *Weight* column.
   c. For every $T_i \in T_T$: set *Time_transition_impact_type* = distribution and fit Δ'_i via MLE and store the parameters in *Time_transition_impact_value.*
3. **Energy Dimension**:
   a. For each T_i:set *Energy_Dimension* = *C* if *Energy Stamp* in MEL ≠ NaN, else *N*;
   b. For *Energy_Dimension* = *C*, set *Energy_transition_impact_type* = rate;
   c. Build an Energy-consumption-profile (ECP) table;
   d. For each unique asset *K* in *MEL*: from the asset-specific MEL, extract the intervals and associated power values for all energy-relevant states (including processing, idle, standby); compute the corresponding rates: $r_{i,Energy} = \frac{1}{m}\sum_{j=1}^{m} \frac{Impact_{i,j}}{\tau_{i,j}}$ and store it in *Energy_transition_impact_value* for the transitions that use asset K. Add columns in the ECP that flag (i) the first event where the asset enters an active state and (ii) the last event before it returns to its available state.
4. **Other Dimensions:** For each T_i: if D_{Stamp} = NaN: set *D_Dimension = N;* else *C* and choose *D_transition_impact_type* ∈ *{Fixed, Rate, Dynamic}* (from Step 1) *and compute D_transition_impact_value* accordingly.

3.4.3. MDSPN model extraction and MDSPN simulation code development

Following the last subsection, we extract the MTT of the system, which enables us to extract individual, unidimensional SPN models for each system objective and then integrate them into a single MDSPN. Because the MTT centralizes every transition's contribution type and impact values, we can readily update the model whenever the objectives of the system change.

In the MDSPN model development process, each entry E_i in the MTT is assigned to a transition MT_E in the MDSPN. To visually and semantically encode each transition's contributions across all dimensions *d*, we split MT_E into *d* segments. Next, we set the color of each segment to white if the transition's type for the assigned dimension to this segment is "*C*" (*type = contributing*) or black if "*N*" (*type = noncontributing*). The resulting MDSPN can be exported in a script-ready format, either as a tool-specific input script or as a general interchange file such as Petri Net Markup Language (PNML) [101] or Multidimensional Petri Net Markup Language (MDPNML) [102], so that the extracted MDSPN model can be directly loaded into any compatible simulation environment.

In this work, the automatically discovered output is an instantiated MDSPN (explicit places, transitions, arcs, plus estimated timed/stochastic parameters). If a generic (or template) representation is preferred for reuse or large-model organization, the extracted per-asset and repeated subnet structures can be packaged as templates without changing the executable net semantics and exported via MDPNML for tool interoperability [102].

3.5. MDSPNs simulation tool: MDPySPN

MDPySPN is an open-source Python library designed for modeling, simulation, and MEL generation of MSPNs. MDPySPN is the extended version of PySPN [68] and is utilized as the core simulation library for this paper's case study. The MDPySPN library supports multiple distribution functions for timed transitions (for instance, exponential, normal, and Weibull), as well as fixed and rate-based values for transitions contributing to dimensions other than the time dimension, idle places for assets, inhibitor arcs, and guard functions. MDPySPN is able to simulate the multidimensional behavior of complex discrete-event systems, such as manufacturing, logistics, and healthcare. A key feature of MDPySPN is its ability to generate synthetic MELs, which are essential for process discovery, validation, and further refinement of CDT models. In cases where access to real-world data is restricted, such as during early-stage system design or in sensitive operational environments, the synthetic MEL generated by MDPySPN can be used as a substitute and enables model analysis in the absence of actual system data. MDPySPN can be employed to develop data-driven CDTs in (near) real-time, which enable system monitoring, support what-if scenario analysis, and assist in multi-objective decision-making in complex systems such as SMSs. Building on the MDSPN model extracted automatically in Section 3.4, we generate the corresponding MDPySPN simulation script so that the MDSPN model can be executed.

3.6. Multidimensional simulation model validation

We conduct the validation process to evaluate whether the extracted model accurately represents the real-world system (or original model), both structurally and operationally. The structural validation [103] is possible when the process model of the original system is available (such as a ground-truth model). In this case, we evaluate the correctness of the network flow of entities and their causal relations, such as counting the number of places, transitions, and arcs, between the extracted model (which can be obtained from the MDPySPN simulation) and the original model. For operational validity (output validation), we compare the

KPIs defined for the original model with those obtained from the simulation of the extracted model, where data in both is generated over the same time duration. The selection of KPIs for validation can be based on the objectives of the system under analysis. Finally, we assess whether the 95 % confidence intervals (CIs) of the defined KPIs overlap [104].

Once the extracted MDSPN is validated against the real-world system, it becomes a CDT of the system and serves as a basis for what-if analyses to support multi-objective decision-making in the system. For example, scenarios that target non-value-adding behaviors, such as minimizing the waiting times or asset idle state periods, and then assessing the improvement on throughput, energy consumption, and other KPIs defined by the real-world system. Furthermore, ML techniques can help refine the models by identifying patterns in simulation outcomes and proposing parameter adjustments automatically, such as optimized scheduling policies. In this paper, as we mostly concentrate on the automated generation of the MDSPN model, we outline a case study related scenario.

4. Illustrative case study

To demonstrate the fully automated, data-driven MDSPN model extraction workflow, we designed an illustrative case study of hot forging process based on an extended version of the study presented in [105]. Within our proposed CDT framework, the hot-forging line represents the real-world entity (observable manufacturing process), while the extracted MDSPN serves as its DT simulation model, constructed from the generated MEL data. We employ our case-study model, which we use as a ground truth reference to generate data, which is subsequently utilized to rediscover the underlying MDSPN model for the dimensions of time, energy consumption, and waste generation. We verify the proposed method by comparing the simulated behavior of the extracted MDSPN against the reference behavior using the relevant KPI measures. Following validation, we employ the CDT simulation model in what-if scenarios to support multi-objective optimization in the case study system. We first describe the case-study system and, following the methodology in Section 3, generate the MEL data for MDSPN extraction and simulation. After validating the extracted model against the ground truth model, we employ the CDT simulation model in an example what-if scenario to support multi-objective optimization in the case study system.

4.1. Case study description

The hot forging system includes a multi-product manufacturing line. Fig. 8 shows the SPN model of the described case study (ground-truth) model. The process begins with the arrival of raw material units (bars), such as steel or aluminum, at the system from a source, following an exponential distribution with $\lambda = 0.1$ (T1). Operators manually transport these bars, taking a triangular-distributed duration of between 3 and 4 min (with a mode of 3.5 min) (T2), to an automatic feeding machine that requires 2 min per operation (T3). Once loaded, the bars are cut into coils by a separation machine, with a processing time following a triangular distribution (minimum 5, mode 5.5, maximum 6 min) (T4). After separation, each billet is routed with equal probability to one of three conveyor lines (cold (T5.3), warm (T5.2), or hot conveyors (T5.1)) that prepare the material for forging in accordance with product requirements. The cold forging conveyor takes 5 min per transfer and consumes relatively low electricity. The hot forging conveyor takes 10 min and consumes a high level of electricity, while the warm conveyor takes 7 min, with electricity consumption between the cold and hot conveyors. The forging machine itself operates with high electricity consumption and processes the billets with a triangular-distributed duration (7–8 min, mode of 7.5 min) (T7). Once forged, the products are transported via another conveyor, with a fixed duration of 5 min, to a manual quality assessment station (T8). The manual quality check takes a triangular-distributed duration of 2–3 min (with a mode of 2.5 min) (T9), after which each product is classified as qualified (T20.2) or unqualified (T10.1), with an acceptance rate of 90 %. Furthermore, in our study, in addition to energy considerations, we incorporate waste material metrics into some processes to incorporate the dimensions of interest for our case study on CDT. Specifically, both the separation machine and the forging process produce a fixed amount of material waste per operation.

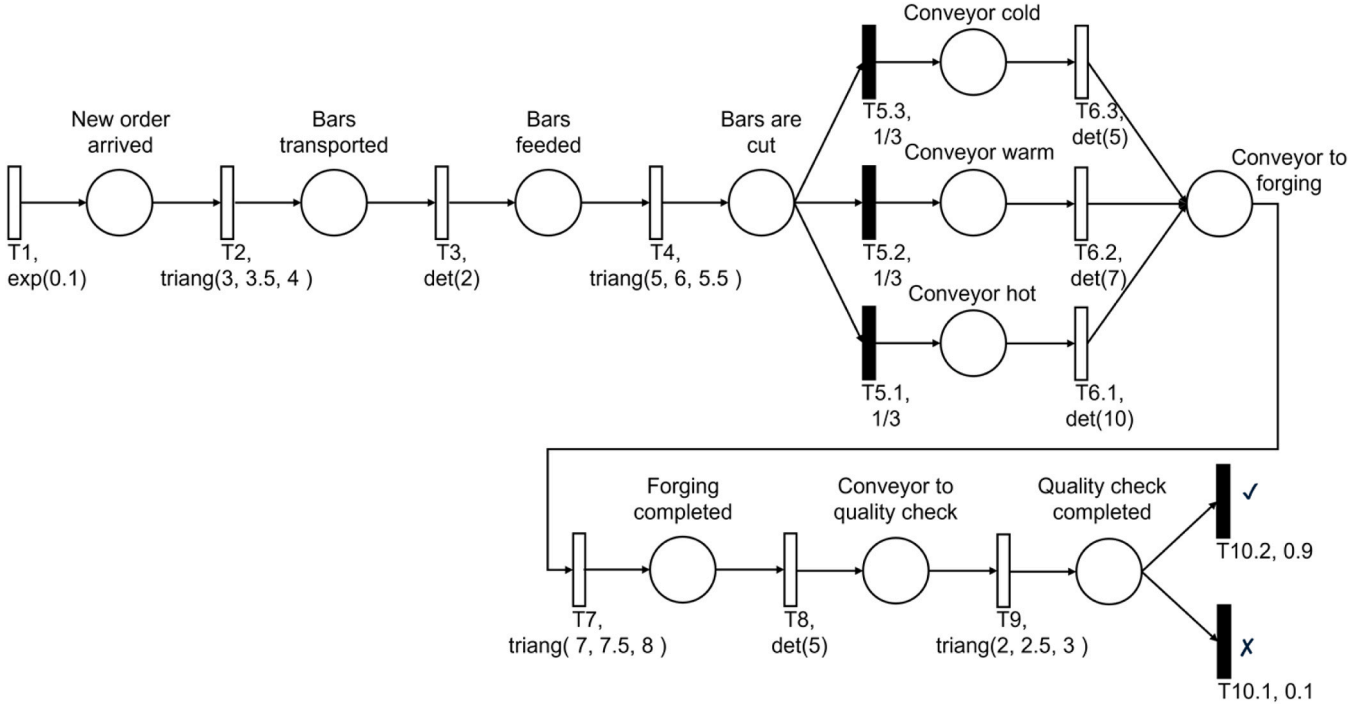

Fig. 8. SPN model of the case study system.

Table 4
Excerpt of the case study MEL.

#	Time Stamp	Case ID	Asset	Energy Stamp	Energy Type	Waste Stamp	Waste Type	Event
…	…	…	…	…	…	…	…	…
24	00:32:29	5775	Conveyor Cold	0.11	Electricity	0.0	NA	Conveyor Cold End
25	00:33:19	5775	Forging Machine	37.34	Electricity	0.05	Iron	Forging Begin
26	00:37:29	5774	Conveyor to Forging	0.10	Electricity	0.0	NA	Conveyor End
27	00:37:29	5774	NA	0.0	NA	0.0	NA	Quality Check Begin
28	00:40:22	5774	NA	0.0	NA	0.0	NA	Quality Check End
…	…	…	…	…	…	…	…	…

Table 5
Except for the case study ECP table.

#	Asset	State	ECP	Enabling Transition
1	Feeding Machine	Active	0.0167	Feeding B2C Begin (T3)
2	Feeding Machine	Idle	0.083	Feeding B2C End (T3)
3	Separation Machine	Active	0.05	Coils to Bullets Begin (T4)
4	Separation Machine	Idle	0.25	Coils to Bullets End (T4)
…	…	…	…	…

4.2. Data collection and preprocessing

Next, we outline the data required to construct the CDT of the case-study system. We first identify the KPIs that drive our data-collection strategy: (i) the number of inputs, (ii) the electricity consumed by each asset, (iii) the material waste produced by each asset, and (iv) the total energy use and waste generation over a 10-hour production day. Using the methodology described in Section 3.2, we then collect a multidimensional MEL for 100 production days. In Table 4, we present an excerpt of the MEL, generated from the ground-truth model of the hot-forging line.

4.3. MDSPN model extraction and simulation

Using the collected MEL from the ground-truth model, we automatically extracted the case-study MDSPN with the procedure described in Section 3.4, integrating the time, energy, and waste dimensions into a single multidimensional model. In Table 5, we present the excerpt for the case study ECP table, where we define each asset's different state, its ECP rate, and the enabling events for its different states. For instance, the feeding machine state turns to active once the activity feeding bars begin and returns to its idle state once the activity feeding bars end.

In Table 6, we present the extracted MTT, where each transition with its behavior in different dimensions is defined. To save space in the table, we abbreviate the dimension-specific column headers as follows: TD for Time Dimension, TDIT for Time-Dimension Impact Type, TDIV for Time-Dimension Impact Value; ED for Energy Dimension, EDIT for Energy-Dimension Impact Type, EDIV for Energy-Dimension Impact Value; and WD for Waste Dimension, WDIT for Waste-Dimension Impact Type, WDIV for Waste-Dimension Impact Value. For example, transition T1 (new-order arrival) contributes only to the time dimension as a timed transition and is non-contributing to all other dimensions. Transition T3 (bar feeding) contributes to both the time and energy dimensions, whereas transition T4 (cutting bars into coils) contributes to all three dimensions: time, energy, and waste generation.

Using the knowledge extracted from the MEL, such as the ECP table and the MTT, we automatically construct the system's MDSPN digital simulation model. In Fig. 9, we present the resulting MDSPN (redrawn for clarity). Each transition icon is divided into three horizontal segments that encode dimensional contributions: the upper segment represents the time dimension, the middle segment the energy dimension, and the lower segment the waste-generation dimension. For example, transition MT7 (forging) contributes to all three dimensions, whereas the conveyor-selection transitions (MT51, MT52, MT53) are non-contributing to every dimension. Finally, we run the extracted MDSPN in the extended version of the MDPySPN, which records all target KPIs directly from the model's execution traces.

4.4. MDSPN simulation model validation

The validation of the case study comprises two components: structural validation and output validation. Structural validity was established by comparing the discovered simulation model with the ground-truth process model, confirming full face validity and consistency of the control-flow structure.

Output validation was then conducted by comparing the predefined KPIs over 100 replications of both the ground-truth and the extracted MDSPN simulation models. The corresponding 95 % CIs are reported in Table 7 and visualized in Fig. 10. Each block of rows in the table refers to one dimension of system performance, time-related KPIs, energy consumption, and waste generation, respectively. Units of Raw Material are denoted by URM, and the Number of Products by NP.

Some of the observed deviations in energy-related KPIs can be explained by differences in the number of input units processed at each stage. For example, in the extracted model, the 95 % CI for the number of inputs is between 58.55 and 58.94 URM, whereas in the ground-truth model, the CI ranges from 57.42 to 60.47 URM. Although this discrepancy in throughput volume and the associated processing times is relatively small, it accumulates across stages and propagates into the energy dimension. This is visible, for instance, at the separation machine: the CI for energy consumption in the extracted model is 93.80–94.45 kWh, while the real system exhibits slightly higher values of 94.99–98.85 kWh. Furthermore, for stages with routing choices, such as the cold, warm, and hot conveyors, the selections made across the 100 runs can lead to substantial variation, especially for the most energy-intensive path (the hot conveyor). Nevertheless, the 95 % CIs for total energy consumption show a large degree of overlap between the extracted and ground-truth models, indicating that the simulation reproduces the aggregate energy behavior of the real system with good

Table 6
Excerpt of the case study MTT.

#	Transition	TD	TDIT	TDIV	Weight	ED	EDIT	EDIV	WD	WDIT	WDIV
1	T1	C	Distribution	weibull(2.83, 6.38, 4.27)	NA	N	NA	NA	N	NA	NA
2	T2	C	Distribution	norm (3.47, 0.20)	NA	N	NA	NA	N	NA	NA
3	T23	N	NA	NA	NA	C	Rate	0.0167	N	NA	NA
4	T3	C	Distribution	lognorm (1.33, 2.0, 0.0)	NA	C	Rate	0.0833	N	NA	NA
5	T4	C	Distribution	lognorm (1.44, 5.10, 0.79)	NA	C	Rate	0.25	C	Fixed	0.02
.	…	…	…	…	…	…	…	…	…	…	…

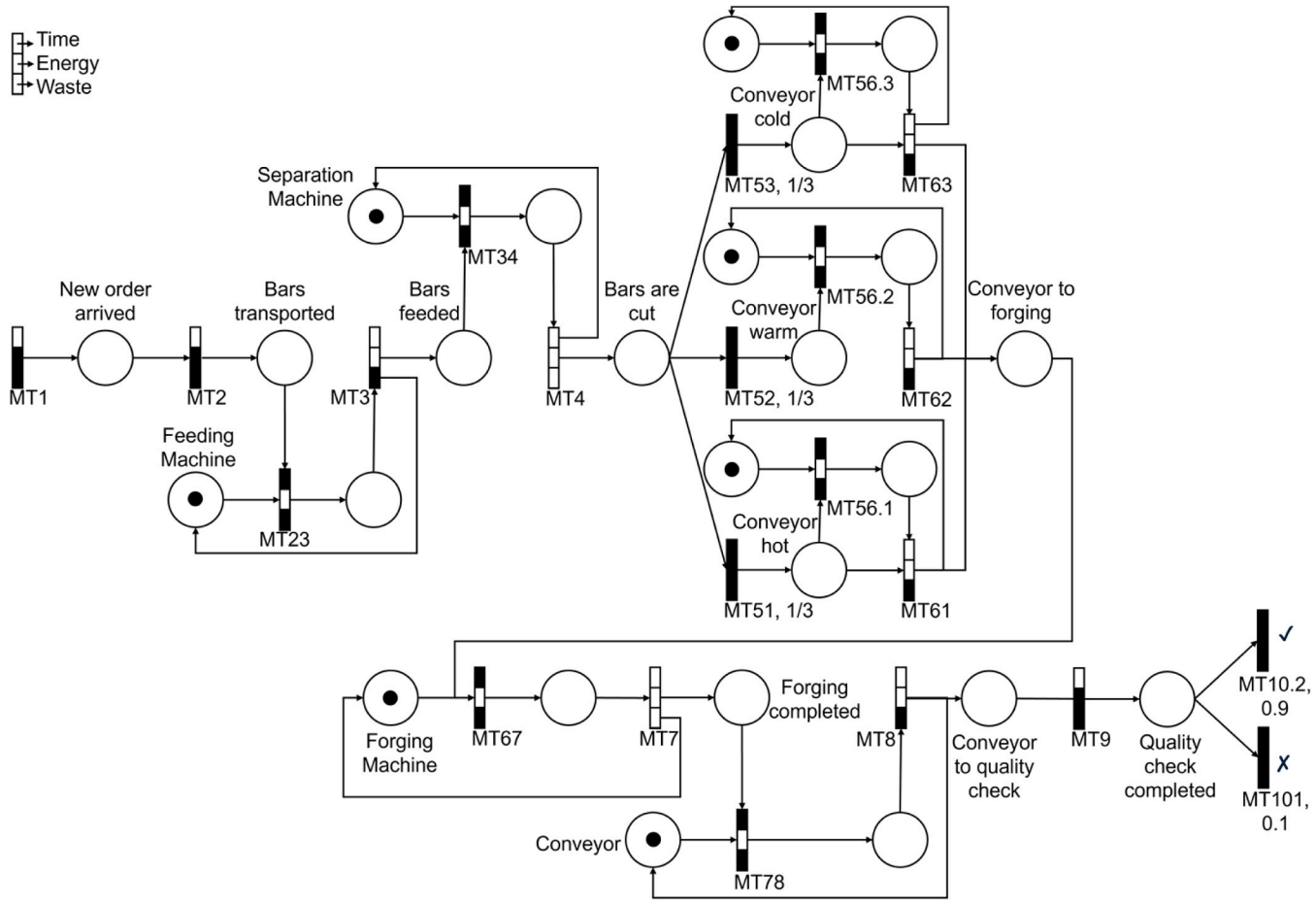

Fig. 9. The extracted MDSPN model of the case study.

Table 7
95 % confidence intervals for KPIs of 100 replications of the ground-truth and the extracted model.

#	KPI	Extracted Model CI Low	Extracted Model CI High	Ground Truth CI Low	Ground Truth CI High
1	Input	58.55 (URM)	58.94 (URM)	57.42 (URM)	60.47 (URM)
2	Output	55.01 (NP)	55.40 (NP)	53.62 (NP)	56.25 (NP)
3	Throughput	0.93	0.94	0.93	0.94
4	Feeding M EC	17.66 (kWh)	17.72 (kWh)	17.41 (kWh)	17.85 (kWh)
5	Separation M EC	93.80 (kWh)	94.45 (kWh)	94.99 (kWh)	98.85 (kWh)
6	Conveyor EC	10.25 (kWh)	10.31 (kWh)	10.02 (kWh)	10.43 (kWh)
7	Cold Conveyor EC	7.11 (kWh)	7.47 (kWh)	7.01 (kWh)	7.41 (kWh)
8	Warm Conveyor EC	67.01 (kWh)	68.50 (kWh)	62.43 (kWh)	63.95 (kWh)
9	Hot Conveyor EC	138.60 (kWh)	142.33 (kWh)	129.05 (kWh)	132.43 (kWh)
10	Forging M EC	856.38 (kWh)	858.78 (kWh)	864.97 (kWh)	879.42 (kWh)
11	Total EC	1193.11 (kWh)	1197.31 (kWh)	1185.17 (kWh)	1206.90 (kWh)
12	Separation M WG	1.14 (kg)	1.15 (kg)	1.12 (kg)	1.18 (kg)
13	Forging M WG	2.78 (kg)	2.80 (kg)	2.71 (kg)	2.84 (kg)
14	Total WG	3.93 (kg)	3.96 (kg)	3.83 (kg)	4.02 (kg)

accuracy.

4.5. Multi-objective decision-making scenarios and results

To demonstrate the application of our proposed CDT to obtain multi-objective decision-making, we simulate our validated CDT model in an example what-if scenario to reduce total energy consumption without degrading other KPIs. For this, we identify non–value-adding activities, such as idle periods for each machinery asset. We then introduced a new standby state for all assets, except for the forging

machine, where any asset that remains idle for more than two minutes automatically transitions to standby [106]. In the standby state, the asset consumes significantly less power (at least 10 times less than the idle state ECP) but requires approximately 30 s to resume operation, doing so at a higher energy rate than when idle. As presented in Table 8, we applied this scenario to our ground-truth model, which resulted in a mean reduction in total energy consumption for 100 runs by 8.38 %, while other KPIs remained unchanged. These results are based on 100 runs for the standby scenario and the

historical data of the base model (before the change) for 100 runs, with 95 % confidence intervals. This scenario illustrates one of many potential scenarios we could deploy in our developed CDT to identify optimized system features. In Fig. 11, we compare the normalized KPIs, including the number of outputs, total energy use, and waste generated for the ground truth model (original model) versus the new scenario. Each metric is scaled between 0 and 1 based on the observed range across both scenarios. The new scenario exhibits a clear improvement in energy efficiency and almost the same results for other KPIs (number of outputs and amount of waste generated).

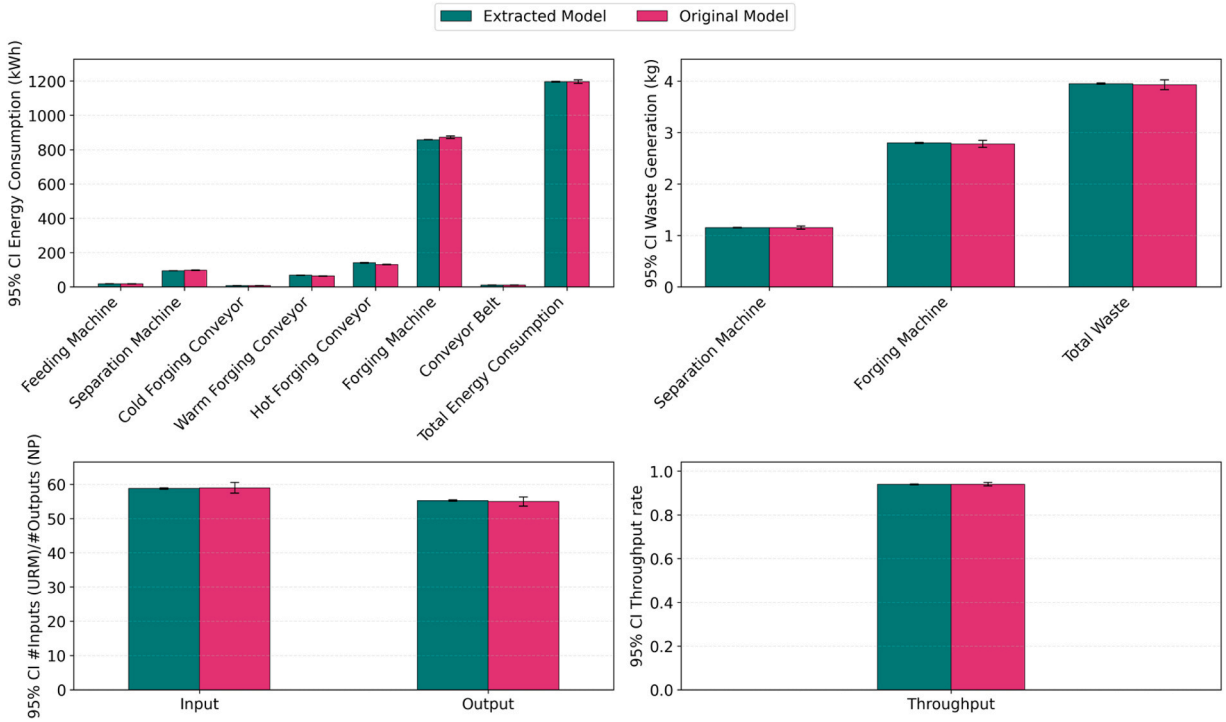

Fig. 10. Graphical comparison of 95 % confidence intervals for KPIs of 100 replications of the ground-truth and the extracted model.

Table 8
Total energy consumption: baseline vs. standby scenario (95 % CI).

Model	Total Energy Consumption CI Low	Total Energy Consumption CI High
Base Model	1187.01 (kWh)	1207.91 (kWh)
New Scenario Model	1084.89 (kWh)	1109.35 (kWh)

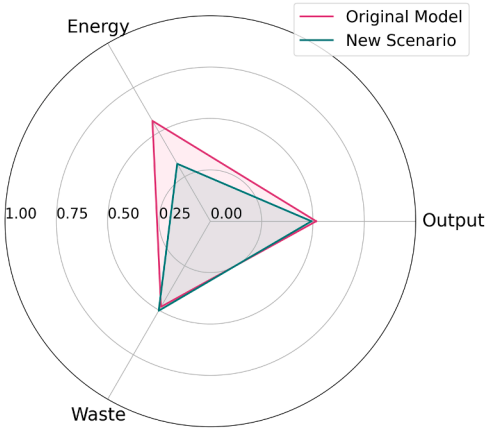

Fig. 11. Radar of normalized KPIs from the new scenario and the original model.

5. Challenges in developing comprehensive digital twins

The introduction of the CDT has presented several challenges, detailed as follows:

- **Data integration and latency**: CDTs require an accurate compiled event log that is both coherent and time-synchronized. Delays in data collection, specifically common in real-world systems, can skew activity durations and also distort event correlations, which result in incorrect model extraction.
- **Domain-Specific Dimensional Behavior**: Different dimensions might exhibit distinct behaviors that necessitate a thorough understanding of the system. For example, the battery-powered Automated Guided Vehicles (AGVs) introduce distinct characteristics, including charge/discharge thresholds and minimum dwell times, and nonlinear charging curves.
- **Human Interaction with the System**: Given the significant role of human participation in manufacturing systems, human actions add stochasticity in task duration, decision-making, and operational efficiency, thereby complicating the modeling and simulation processes.

6. Summary and outlook

Complex systems, such as manufacturing systems, involve multiple, and often conflicting, objectives, including increasing throughput, improving energy efficiency, and reducing CO_2 emissions and material waste. These objectives correspond to distinct dimensions of the system that must be considered simultaneously. To obtain an optimized multi-objective decision in such systems, stakeholders require a comprehensive understanding of the system's behavior in each of these dimensions. Recent advances in industrial technology have established the Digital Twin as a virtual replica of the physical system, which enables system analysis and supports decision-making by utilizing real-time system data. The current state of the art does not yet offer an automated approach to extract system models that coherently and intuitively represent the system's behavior across multiple dimensions.

In this paper, we address this gap by presenting a methodology for comprehensive Digital Twins and detailing our automated, data-driven approaches for their development. We specify the required data and their structured representation (the multidimensional event log), describe our Multi-Flow Process Mining (MFPM) approach, and detail the steps to extract, simulate, and validate systems' multidimensional

stochastic Petri Nets (MDSPNs) as underlying simulation models of comprehensive Digital Twins. Through an illustrative case study of a hot forging packing process, we demonstrated our proposed methodology for developing comprehensive Digital Twins that encompass the dimensions of time, energy consumption, and waste generation. We employed our validated comprehensive Digital Twin model to simulate a what-if scenario aimed at enhancing overall energy efficiency. The scenario resulted in an average 8.38 % reduction in total energy consumption without impacting other key performance indicators, such as total production output and waste generation. This result highlights the capability of the comprehensive Digital Twin to support multi-objective decision-making and explore optimal system configurations. Several opportunities remain to extend our proposed comprehensive Digital Twins capabilities as follows:

- Multi-objective decision support: Implement algorithms to automatically recommend/support optimal multi-objective decisions.
- Establish an interchange format: Extend existing standardized interchange formats, such as Petri Net Markup Language, to support multidimensional stochastic Petri-net annotations, and seamless model exchange between simulation tools.

CRediT authorship contribution statement

Atieh Khodadadi: Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Sanja Lazarova-Molnar:** Writing – review & editing, Supervision, Project administration, Methodology, Formal analysis, 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.

Acknowledgment

The authors extend their thanks for the funding received from the ONE4ALL project funded by the European Commission, Horizon Europe Programme under Grant Agreement No. 101091877.

References

[1] G. Erbach, "European climate law," in "Regulation (EU)," European Parliamentary Research Service (EPRS), European Parliament, PE 649.385, 2021, vol. 1119. [Online]. Available: ⟨https://www.univiu.org/images/aauniviu2017/GP/co-curr/European_climate_law.pdf⟩.
[2] EU, "Regulation (EU) 2021/1119 of the European Parliament and of the Council of 30 June 2021 establishing the framework for achieving climate neutrality and amending Regulations (EC) No 401/2009 and (EU) 2018/1999 ('European Climate Law')," *Official Journal of the European Union L* 243, vol. 1, p. 64, 2021.
[3] J. Cresko *et al.*, "US department of energy's industrial decarbonization roadmap," U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), DOE/EE-2635, 2022. [Online]. Available: ⟨https://www.osti.gov/servlets/purl/1961393⟩.
[4] Chinchorkar S. Data-Driven Paradigm for Smart Manufacturing in the Context of Big Data Analytics. Big Data Analytics in Smart Manufacturing. Chapman and Hall/CRC; 2022. p. 21–34.
[5] Sharma R, Villányi B. Evaluation of corporate requirements for smart manufacturing systems using predictive analytics. Internet Things 2022;19:100554.
[6] Soori M, Arezoo B, Dastres R. Internet of things for smart factories in industry 4.0, a review. Internet Things CyberPhys Syst 2023.
[7] Kagermann H, Anderl R, Gausemeier J, Schuh G, Wahlster W. Industrie 4.0 in a Global Context: strategies for cooperating with international partners. Herbert Utz Verlag; 2016.
[8] M. Breque, L.De Nul, and A. Petridis, Industry 5.0: towards a sustainable, human-centric and resilient European industry, Directorate General for Research and Innovation (DG RTD) of the European …, 2021.
[9] Zhang W, Van Luttervelt C. Toward a resilient manufacturing system. CIRP Ann 2011;60(1):469–72.
[10] Liu S, Zheng P, Bao J. Digital Twin-based manufacturing system: a survey based on a novel reference model. J Intell Manuf 2023:1–30.
[11] Friederich J, Francis DP, Lazarova-Molnar S, Mohamed N. A framework for data-driven digital twins of smart manufacturing systems. Comput Ind 2022;136:103586.
[12] Lugaresi G, Zanotti M, Tarasconi D, Matta A. Manufacturing systems mining: Generation of real-time discrete event simulation models. 2019 IEEE international conference on systems, man and cybernetics (SMC). IEEE; 2019. p. 415–20.
[13] van der Aalst WM, Leemans SJ. Learning generalized stochastic petri nets from event data. Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III. Springer; 2024. p. 3–17.
[14] Khodadadi A, Lazarova-Molnar S. Multi-flow process mining for comprehensive simulation model discovery. Presente ICICM 2024.
[15] Khodadadi A, Lazarova-Molnar S. Multi-Flow Process Mining as an Enabler for Comprehensive Digital Twins of Manufacturing Systems. Winter Simulation Conference, Seattle. Washington: IEEE; 2025.
[16] Dreher S, Reimann P, Gröger C. Application fields and research gaps of process mining in manufacturing companies. INFORMATIK 2020. Bonn: Gesellschaft für Informatik; 2021. p. 621–34.
[17] Yu H, Ogbeyemi A, Lin W, He J, Sun W, Zhang W-J. A semantic model for enterprise application integration in the era of data explosion and globalisation. Enterprise Information Systems, 17; 2023, 1989495.
[18] Van der Aalst W, Weijters T, Maruster L. Workflow mining: discovering process models from event logs. IEEE Trans Knowl data Eng 2004;16(9):1128–42.
[19] *Automation systems and integration — Digital twin framework for manufacturing — Part 1: Overview and general principles*, International Standard I. O. f. Standardization, Geneva, Switzerland, 2021. [Online]. Available: ⟨https://www.iso.org/standard/75066.html⟩.
[20] Kritzinger W, Karner M, Traar G, Henjes J, Sihn W. Digital Twin in manufacturing: A categorical literature review and classification. IfacPap 2018;51(11):1016–22.
[21] Camargo M, Dumas M, González-Rojas O. Automated discovery of business process simulation models from event logs. Decis Support Syst 2020;134:113284.
[22] Uhlemann TH-J, Schock C, Lehmann C, Freiberger S, Steinhilper R. The digital twin: demonstrating the potential of real time data acquisition in production systems. Procedia Manuf 2017;9:113–20.
[23] Al Faruque MA, Muthirayan D, Yu S-Y, Khargonekar PP. Cognitive digital twin for manufacturing systems. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE; 2021. p. 440–5.
[24] Zheng P, et al. Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives. Front Mech Eng 2018;13:137–50.
[25] Li L, Lei B, Mao C. Digital twin in smart manufacturing. J Ind Inf Integr 2022;26:100289.
[26] Grieves M. Digital twin: manufacturing excellence through virtual factory replication. "White paper,", 1; 2014. ⟨https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf⟩.
[27] Batty M. Digital twins, 45. London, England: Sage Publications Sage UK; 2018. p. 817–20.
[28] Bi Z, Zhang CW, Wu C, Li L. New digital triad (DT-II) concept for lifecycle information integration of sustainable manufacturing systems. J Ind Inf Integr 2022;26:100316.
[29] Zhang W, Van der Werff K. Guidelines for product data model formulation using database technology. Proc. of Int. Conf. on Engineering Design (ICED'93), 3; 1993. p. 1618–26.
[30] Ma Z, Zhang W, Ma W, Chen G. Extending EXPRESS-G to model fuzzy information in product data model. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 35111. American Society of Mechanical Engineers; 2000. p. 285–91.
[31] Li Q, Zhang W, Tso S. Generalization of strategies for product data modeling with special reference to Instance-As-Type problem. Comput Ind 2000;41(1):25–34.
[32] Lazarova-Molnar S. A Vision for Advancing Digital Twins Intelligence: Key Insights and Lessons from Decades of Research and Experience with Simulation. 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications. SCITEPRESS Digital Library; 2024. p. 5–10.
[33] Gudivada V, Apon A, Ding J. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. Int J Adv Softw 2017;10(1):1–20.
[34] Duarte L, Neto P. Classification of primitive manufacturing tasks from filtered event data. J Manuf Syst 2023;68:12–24.
[35] Shu X, Ye Y. Knowledge discovery: methods from data mining and machine learning. Soc Sci Res 2023;110:102817.
[36] Van Der Aalst WM, Van Dongen BF. Discovering petri nets from event logs. Transactions on Petri nets and other models of concurrency vii. Springer; 2013. p. 372–422.
[37] Banks J, Carson JS. Introduction to discrete-event simulation. Proceedings of the 18th conference on Winter simulation. 1986. p. 17–23.
[38] Rebs T, Brandenburg M, Seuring S. System dynamics modeling for sustainable supply chain management: a literature review and systems thinking approach. J Clean Prod 2019;208:1265–80.
[39] Agalianos K, Ponis S, Aretoulaki E, Plakas G, Efthymiou O. Discrete event simulation and digital twins: review and challenges for logistics. Procedia Manuf 2020;51:1636–41.

[40] Overbeck L, Graves SC, Lanza G. Development and analysis of digital twins of production systems. Int J Prod Res 2024;62(10):3544–58.
[41] Antelmi A, Cordasco G, D'Ambrosio G, De Vinco D, Spagnuolo C. Experimenting with agent-based model simulation tools. Appl Sci 2022;13(1):13.
[42] Khan AA, Abonyi J. Simulation of sustainable manufacturing solutions: tools for enabling circular economy. Sustainability 2022;14(15):9796.
[43] Kampa A, Gołda G, Paprocka I. Discrete event simulation method as a tool for improvement of manufacturing systems. Computers 2017;6(1):10.
[44] Li W, Huynh BH, Akhtar H, Myo KS. Discrete event simulation as a robust supporting tool for smart manufacturing. Implement Ind 4 0 Model Fact Key Enabler Future Manuf 2021:287–312.
[45] Magnanini MC, et al. A digital twin-based approach for multi-objective optimization of short-term production planning. IFACPap 2021;54(1):140–5.
[46] Tsinarakis G, Sarantinoudis N, Arampatzis G. A discrete process modelling and simulation methodology for industrial systems within the concept of digital twins. Appl Sci 2022;12(2):870.
[47] van Cruchten R, Weigand H. Towards event log management for process mining-vision and research challenges. International Conference on Research Challenges in Information Science. Springer; 2022. p. 197–213.
[48] Bozkaya M, Gabriels J, Van der Werf JM. Process diagnostics: a method based on process mining. 2009 International Conference on Information, Process, and Knowledge Management. IEEE; 2009. p. 22–7.
[49] Brockhoff T, et al. Process prediction with digital twins. 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE; 2021. p. 182–7.
[50] Naderifar V, Sahran S, Shukur Z. A review on conformance checking technique for the evaluation of process mining algorithms. TEM J 2019;8(4):1232.
[51] de Leoni M. Foundations of process enhancement. Process Mining Handbook. Springer; 2022. p. 243–73.
[52] Friederich J, Lazarova-Molnar S. Data-driven reliability modeling of smart manufacturing systems using process mining. 2022 Winter Simulation Conference (WSC). IEEE; 2022. p. 2534–45.
[53] Bemthuis RH, Lazarova-Molnar S. Discovering agent models using process mining: Initial approach and a case study. 2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). IEEE; 2022. p. 163–72.
[54] Flammini F. Data-Driven Anomaly Detection in Smart-Railways through Self-Adaptation, Process Mining, and Digital Twins. 2024 IEEE International Conference on Big Data (BigData). IEEE; 2024. p. 8653–7.
[55] Kumbhar M, Ng AH, Bandaru S. A digital twin based framework for detection, diagnosis, and improvement of throughput bottlenecks. J Manuf Syst 2023;66:92–106.
[56] Vogelgesang T, Appelrath H-J. Multidimensional process mining: a flexible analysis approach for health services research. Proc Jt EDBT/ICDT 2013 Workshops 2013:17–22.
[57] Vogelgesang T, Appelrath H-J. Multidimensional ProCess Mining with Pmcube Explorer. BPM 2015:90–4.
[58] T. Vogelgesang, G. Kaes, S. Rinderle-Ma, and H.-J. Appelrath, "Multidimensional process mining: Questions, requirements, and limitations," 2016.
[59] Erdogan TG, Tarhan AK. Multi-perspective process mining for emergency process. Health Inform J 2022;28(1):14604582221077195.
[60] Xu H, Pang J, Yang X, Yu J, Li X, Zhao D. Modeling clinical activities based on multi-perspective declarative process mining with openEHR's characteristic. BMC Med Inform Decis Mak 2020;20:1–11.
[61] F. Mannhardt, "Multi-perspective process mining," 2018.
[62] Bolt A, van der Aalst WM. Multidimensional process mining using process cubes. International Workshop on Business Process Modeling, Development and Support. Springer; 2015. p. 102–16.
[63] Knoll D, Reinhart G, Prüglmeier M. Enabling value stream mapping for internal logistics using multidimensional process mining. Expert Syst Appl 2019;124:130–42.
[64] Kroeger S, Rafles A, Jordan P, Soellner C, Zaeh MF. Data model to enable multidimensional process mining for data farming based value stream planning in production networks. Prod Eng 2024:1–21.
[65] Guzzo A, Joaristi M, Rullo A, Serra E. A multi-perspective approach for the analysis of complex business processes behavior. Expert Syst Appl 2021;177:114934.
[66] S. Sim, L. Liu, and H. Bae, "Automatic Discovery of Multi-perspective Process Model using Reinforcement Learning," *arXiv preprint arXiv:2211.16687*, 2022.
[67] Aalst W v d. Business alignment: using process mining as a tool for delta analysis and conformance testing. Requir Eng 2005;10:198–211.
[68] Friederich J, Khodadadi A, Lazarova-Molnar S. PySPN: a python library for modeling, simulation, and event log generation of stochastic petri nets. Trans Soc Model Simul Int 2025;2025.
[69] Brzychczy E, Szpyrka M, Korski J, Nalepa GJ. Imperative vs. declarative modeling of industrial process. The case study of the longwall shearer operation. IEEE Access 2023;11:54495–508.
[70] S.A. White, "Introduction to BPMN," in "Ibm Cooperation," Object Management Group (OMG), 2004. [Online]. Available: ⟨https://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf⟩.
[71] Cecconi A, De Giacomo G, Di Ciccio C, Maggi FM, Mendling J. Measuring the interestingness of temporal logic behavioral specifications in process mining. Inf Syst 2022;107:101920.
[72] Maggi FM, Bose RJC, van der Aalst WM. Efficient discovery of understandable declarative process models from event logs. Advanced Information Systems Engineering: 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings 24. Springer; 2012. p. 270–85.
[73] Kalenkova AA, van der Aalst WM, Lomazova IA, Rubin VA. Process mining using BPMN: relating event logs and process models. Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. 2016. p. 123.
[74] Schröder C, van Detten JN, Leemans SJ. Locally optimized process tree discovery. International Conference on Process Mining. Springer; 2024. p. 389–401.
[75] Kalenkova AA, Lomazova IA, van der Aalst WM. Process model discovery: A method based on transition system decomposition. International Conference on Applications and Theory of Petri Nets and Concurrency. Springer; 2014. p. 71–90.
[76] Davidrajuh R. Petri nets for modeling of large discrete systems. Singapore: Springer; 2021.
[77] Wu N, Zhou M, Hu G. On Petri net modeling of automated manufacturing systems. 2007 IEEE International Conference on Networking, Sensing and Control. IEEE; 2007. p. 228–33.
[78] Gutowska K, Kogut D, Kardynska M, Formanowicz P, Smieja J, Puszynski K. Petri nets and ODEs as complementary methods for comprehensive analysis on an example of the ATM–p53–NF-κ B signaling pathways. Sci Rep 2022;12(1):1135.
[79] Kahraman C, Tüysüz F. Manufacturing system modeling using petri nets. Production Engineering and Management under Fuzziness. Springer; 2010. p. 95–124.
[80] Kaid H, Al-Ahmari A, El-Tamimi AM, Nasr EAbouel, Li Z. Design and implementation of deadlock control for automated manufacturing systems. South Afr J Ind Eng 2019;30(1):1–23.
[81] Pommereau F. SNAKES: A flexible high-level petri nets library (tool paper). Application and Theory of Petri Nets and Concurrency: 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings 36. Springer; 2015. p. 254–65.
[82] Westergaard M, Kristensen LM. The access/cpn framework: A tool for interacting with the cpn tools simulator. International conference on applications and theory of Petri nets. Springer; 2009. p. 313–22.
[83] Looney CG. Fuzzy Petri nets for rule-based decisionmaking. IEEE Trans Syst Man Cybern 1988;18(1):178–83.
[84] Zimmermann A. Colored petri nets. Stoch Discret Event Syst Model Eval Appl 2008:99–124.
[85] F. Bause and P.S. Kritzinger, *Stochastic petri nets*. Vieweg Wiesbaden, 2002.
[86] Zhang W, Bi Z, Zha X. A generic petri net model for flexible manufacturing systems and its use for FMS control software testing. Int J Prod Res 2000;38(5):1109–31.
[87] Wang J, Ip W, Muddada RR, Huang J, Zhang W. On Petri net implementation of proactive resilient holistic supply chain networks. Int J Adv Manuf Technol 2013;69(1):427–37.
[88] S. Lazarova-Molnar, "The proxel-based method: Formalisation, analysis and applications," PhD dissertation, Otto-von-Guericke-Universität Magdeburg, 2005.
[89] A. Khodadadi, A. Zare, M. Jungmann, M. Götz, and S. Lazarova-Molnar, "A Tutorial on Data-Driven Petri Net Model Extraction and Simulation for Digital Twins in Smart Manufacturing," presented at the Winter Simulation Conference, Seattle, Washington, 2025, 2025.
[90] Zeigler BP, Praehofer H, Kim TG. Theory of modeling and simulation. Second ed. Academic press; 2000.
[91] Van Der Aalst WM. Learning Colored Petri Nets Using Object-Centric Event Data (OCED2CPN). 2023 7th IEEE Congress on Information Science and Technology (CiSt). IEEE; 2023. p. 1–6.
[92] Y. Wang, G. Zacharewicz, D. Chen, and M.K. Traoré, "Integrating dependency with DEVS in the process mining," presented at the New Information Communication Sciences and Technology for Sustainable Development (NICST 2015), Bordeaux, France, 2015. [Online]. Available: ⟨https://hal.science/hal-01551430v1⟩.
[93] De Giacomo G, Oriol X, Estanol M, Teniente E. Linking data and BPMN processes to achieve executable models. International Conference on Advanced Information Systems Engineering. Springer; 2017. p. 612–28.
[94] AnyLogic. "AnyLogic Simulation Software." ⟨https://www.anylogic.com/⟩ (accessed 13rd March 2025.
[95] "ExtendSim." ⟨https://extendsim.com/⟩ (accessed 2024).
[96] "Simulink." ⟨https://www.mathworks.com/products/simulink.html⟩ (accessed 2024).
[97] A. Khodadadi and S. Lazarova-Molnar, "Multidimensional Stochastic Petri Nets: A Novel Approach to Modeling and Simulation of Stochastic Discrete-Event Systems," presented at the IEEE SMC, 2025.
[98] A. Khodadadi and S. Lazarova-Molnar, "Essential Data Requirements for Industrial Energy Efficiency with Digital Twins: A Case Study Analysis," presented at the ED-I40, 2023.
[99] Fink L, Matyas G, Zink F, Wallner B, Bleicher F, Trautner T. Severity of failure: resilience assessment on the shop floor. Procedia CIRP 2025;134:85–90.
[100] Virtanen P, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 2020;17(3):261–72.
[101] Billington J, et al. The petri net markup language: Concepts, technology, and tools. International Conference on Application and Theory of Petri Nets. Springer; 2003. p. 483–505.
[102] A. Khodadadi and S. Lazarova-Molnar, "MDPNML: A Multidimensional Petri Net Markup Language Enabling Construction and Simulation of Comprehensive Digital Twin Models," presented at the The International Conference on Model-Based Software and Systems Engineering (MODELSWARD), 2026.
[103] J. Martens and F. Put, "A theory of structural model validity in simulation," KU Leuven – Departement Toegepaste Economische Wetenschappen (DTEW),

Leuven, Belgium, 9936, 1999. [Online]. Available: ⟨https://lirias.kuleuven.be/1834098?limo=0⟩.
[104] Sargent RG. A tutorial on verification and validation of simulation models. Proceedings of the 16th conference on Winter simulation. 1984. p. 114–21.
[105] Wenzel S, Rabe M, Strassburger S, von Viebahn C. Energy-Related Material Flow Simulation in Production and Logistics. Springer; 2024.
[106] Simunic T, Benini L, Glynn P, De Micheli G. Event-driven power management. IEEE Trans ComputAided Des Integr Circuits Syst 2001;20(7):840–57.