Concurrency and Objects Matter! Disentangling the Fabric of Real Operational Processes to Create Digital Twins

Wil M.P. van der Aalst

[1] Process and Data Science (Informatik 9), RWTH Aachen University, Aachen, Germany
[2] Fraunhofer-Institut für Angewandte Informationstechnik (FIT), Sankt Augustin, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. Process mining dramatically changed the way we look at process models and operational processes. Even seemingly simple processes like Purchase-to-Pay (P2P) and Order-to-Cash (O2C) are often amazingly complex, and traditional hand-made process models fail to capture the true fabric of such processes. Many processes are inherently concurrent and involve interaction between different objects (customers, suppliers, orders, items, shipments, payments, machines, workers, etc.). Process mining uses event data to construct process models that can be used to diagnose performance and compliance problems. If such models reflect reality well, they can be used for forward-looking forms of process mining, including predictive analytics, evidence-based automation, and what-if simulation. The ultimate goal is to create a "digital twin of an organization" that can be used to explore different improvement actions. This paper provides a high-level overview of the different process mining tasks followed by a more detailed discussion on concurrency and object-centricity in process mining.

Keywords: Process Mining · Event Data · Concurrency · Digital Twins

1 Towards a Digital Twin of an Organization

The desire to adequately describe operational processes has been around since the 1890-ties when the field of *scientific management* emerged. Scientific management is also known as Taylorism, named after its pioneer Frederick Winslow Taylor (1856-1915) who tried to systematically improve economic efficiency, especially labor productivity. Taylor systematically observed how people work and can be seen as the "first process miner" using pen and paper (see Figure 1). In 1950 computers started to influence business processes. However, the systematic use of data about operational processes is much more recent [1].

The desire to build computer models that mimic organizations and processes is also not that new. Since the 1960-ties so-called *discrete event simulation* tools have been available with SIMULA [11] as one of the first influential examples. In discrete event simulation it is common to estimate parameters and distributions based on observed data (e.g., service times and arrival rates). However, one still needs to model the process by hand. The first comprehensive approaches to automatically learn complete simulation

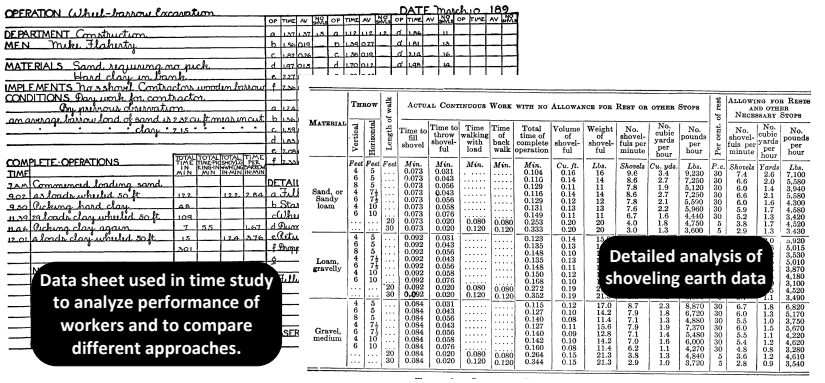

Fig. 1. Analyzing event data to improve operational processes is not that new. This is illustrated by some of the tables in [33]. Frederick Winslow Taylor can be seen as the first "process miner" using manually collected event data.

models from event data became available around 2008 [30, 31]. Based on event logs, it is possible to learn a control-flow model (transition system, Petri net, of BPMN model) that is enriched with information about resources, data, and time using replay or alignment techniques [30, 31].

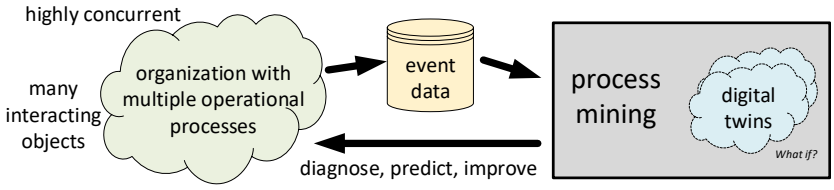

Fig. 2. Process mining provides a concrete approach to create a digital twin of an organization and its operational processes. A key element is the creation of a model based on event data that is able to mimic reality as well as possible. Such a model needs to be able to capture concurrency and interacting objects (customers, workers, products, orders, payments, shipments, etc.).

The notion of a *digital twin* is part of the Industry 4.0 development facilitated through advanced data analytics (machine learning, process mining, etc.) and the Internet of Things (IoT) connectivity [20, 15]. The notion can be described as an effortless integration of the "real reality" and a "modeled reality" in both directions. The "modeled reality" is based on the "real reality", but may also influence the "real reality". This is one of the key concepts in the *Internet of Production* (IoP) developed at RWTH Aachen University [6]. In IoP, *process mining* plays a key role. Gartner coined the term *digital twin of an organization* to indicate that the desire to create a digital twin is not limited to specific Industry 4.0 applications [19]. The goal is to create a virtual representation of an organization and its operational processes (including assets such as

architectures, infrastructures, roles, responsibilities, products, etc.) to assess the impact of change in a controlled environment. Note that this is only a vision that is still far away from reality. However, it illustrates the role that models will need to play in the future.

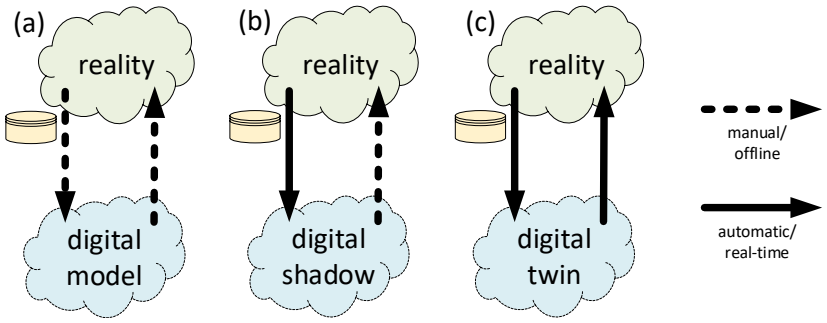

Fig. 3. The difference between a digital model, a digital shadow, and a digital twin.

Figure 3 illustrates the difference between (a) a digital model, (b) a digital shadow, and (c) a digital twin. Building a discrete event simulation model (using e.g. Arena, AnyLogic, CPN Tools, FlexSim, Vensim, or Simul8) in a classical way corresponds to the *digital model* notion in Figure 3(a). The dashed lines show that the model is created by hand. There is no automated connection between reality and the digital model. Moreover, insights generated by the simulation model do not automatically lead to concrete actions. The *digital shadow* notion in Figure 3(b) uses a model, driven by data automatically extracted from reality. If such a connection is automated, it is often possible and desirable to update the model continuously. If reality changes, also the model changes. However, insights and diagnostics still need to be translated into actions manually. The *digital twin* notion in Figure 3(c) shows that there is an automated and real-time connection between reality and the model(s) in both directions. As a result, the digital twin directly influences reality, possibly without human intervention.

It is important to note that many of these ideas have been realized in the context of process mining, albeit with a focus on individual processes in isolation [1]. Most process mining techniques aim to create a digital shadow, as indicated in Figure 3(b). This ranges from control-flow discovery (from mining directly follows graphs [2] to scalable inductive mining [23]) to automatically creating simulation models (e.g., [30, 31]). However, under the umbrella term of "operational support" [1], process mining also aims to impact the process automatically in real-time. An early example is the work presented in [32], where workflow technology is connected to process mining. In [32] YAWL is used as a workflow management system, ProM as a process mining system, and CPN Tools as the simulation engine. ProM is used to learn a faithful simulation model from the event data of YAWL and/or the models in YAWL. At any point in time, the current state of the YAWL workflow system can be loaded into the simulation model and simulated using CPN Tools. This concept is termed *short-term simulation* because

rather than focusing on the steady-state behavior, the focus is on transient behaviors and answering what-if questions. Commercial process mining tools increasingly support what we call "action-oriented process mining". This means that diagnostics are turned into actions. The recent release of the Celonis EMS (Execution Management System), which embeds a low-code workflow management system, illustrates this trend.

The above shows that the idea of creating a digital twin was already realized in the field of process mining long before the term became "in vogue". However, existing approaches typically focus on well-defined processes that are considered in isolation. We are still far away from creating a realistic "digital twin of an organization". In this paper, we focus on two of the many challenges to create such digital twins:

- *Concurrency.* Organizations are like distributed systems or social systems. The different parts operate autonomously but need to synchronize at selected points in time. Although most organizations and systems are highly concurrent, the dominant paradigm is still the highly sequential Turing machine model created in 1936 which does not allow for concurrency. The von Neumann architecture defined in 1945 is based on the Turing machine and also views computation as a sequential process. Moreover, automata, transition systems, Markov chains, and many other representations of behavior do not support concurrency. If concurrency is supported, it is often added as an afterthought. Representations that start from concurrency, like Petri nets, are still the exception. Consider for example a Petri net without places and just transitions. Even people familiar with Petri nets have difficulties to accept that such a Petri net allows for any behavior (and that Petri nets are much more declarative than commonly assumed). Although organizations are highly concurrent, event logs are viewed as sequential (i.e., events are assumed to be totally ordered). This complicates the creation of a digital twin from event data.
- *Object-centricity.* Most of the modeling notations used (e.g., BPMN, Workflow Nets, UML activity diagrams, etc.) assume a single case notion. However, events may involve a variety of objects. Consider for example batching where in one event many objects are affected or an assembly step where a collection of objects is transformed into a new composite object. When drawing for example a BPMN model one needs to pick one case notion (the process instance). In many applications this is not so easy. Consider for example the hiring process of new employees. Is the vacancy the case or the application? One can also consider the classical example of ordering books from Amazon. One order may include multiple books, a shipment may contain books of different orders, and an order may involved multiple shipments. Possible case notions are order, book, and shipment. It is impossible to create a digital twin of an organization without being able to represent the different objects and their interactions.

For example, imagine a car factory producing hundreds of cars per day with each car assembled from thousands of components. Process models that do not allow for concurrency and object-centricity are clearly unable to describe such a factory as a digital twin.

The remainder of this paper is organized as follows. Section 2 present a short high-level introduction to process mining. Section 3 discusses event logs and the importance of concurrency and object-centricity. Section 4 concludes this short paper.

2 Process Mining: A Top-Down View

In recent years, we could witness an uptake in process mining. There used to be a gap between *process science* (i.e., tools and techniques to improve operational processes) and *data science* (i.e., tools and techniques to extract value from data). Mainstream machine learning and data mining techniques do not consider operational processes. Business Process Management (BPM) and Operations Research (OR) tend to start from models rather than data. Process mining bridges this gap [1].

Currently, there are over 35 commercial process mining vendors (ABBYY Timeline, ARIS Process Mining, BusinessOptix, Celonis Process Mining, Disco/Fluxicon, Everflow, Lana, Mavim, MPM, Minit, PAFnow, QPR, etc.) and process mining is applied in most of the larger organizations in countries such as Germany and The Netherlands. Example application domains include: finance (Rabobank, Hypovereinsbank, etc.), telecom (Deutsche Telekom, Vodafone, etc.), logistics (Vanderlande, etc.), production (BMW, Siemens, Fiat, Bosch, etc.), food (Edeka, etc.), fashion (Zalando, etc.), energy (E-on, etc.), transport (Uber, DB, Lufthansa, etc.), healthcare (AstraZenica, Medtronic, etc.), consulting (Deloitte, EY, KPMG, etc.), and IT systems (Dell, IBM, ServiceNow, etc.).

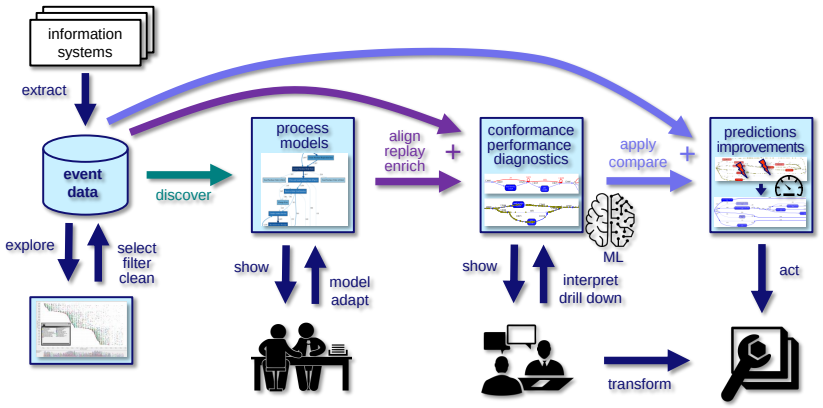

Fig. 4. Overview of the process mining pipeline.

Figure 4 shows a high-level overview of process mining. Event data need to be extracted from information systems. Such data can be explored, filtered, and cleaned. Process discovery tools transform event data into process models (e.g., BPMN, Petri nets, and UML activity diagrams). There are simple approaches like creating so-called Directly-Follows-Graphs (DFGs) that do not discover concurrency thus having obvious problems [2]. The Alpha algorithm was the first to discover concurrent processes [7]. This approach provides some guarantees, but most processes do not satisfy the assumptions described in [7]. After the Alpha algorithm, dozens of more sophisticated algorithms were proposed [1, 9, 21–23, 34]. Using replay and alignment techniques it

is possible to relate process models (hand-made or discovered) with event data. This can be used to discover differences between reality and model [1, 10, 29]. Moreover, the model can be extended with additional perspectives, e.g., organizational aspects, decisions, and temporal aspects. This way, detailed performance analyses are possible. Root-cause analysis can be performed for both conformance and performance problems. It is always possible to relate observations to the original event data. Such evidence-based diagnostics aid discussions about root-causes and possible improvements. The right-hand side of Figure 4 refers to forward-looking techniques aimed at improving the processes. Process models extended with additional perspectives (organizational aspects, decisions, and temporal aspects) can be used to predict conformance and performance problems. As described in [1], predictions can be used to generate recommendations. Figure 4 shows that Machine Learning (ML) techniques can be used in this step. These may range from novel deep learning approaches (e.g., artificial recurrent neural networks like LSTM) to more traditional approaches like logistic regression and decision-tree learning.

It should be noted that process mining techniques are different from mainstream Machine Learning (ML) techniques. However, as Figure 4 shows, process mining can be used to generate ML problems. The current trend is to make process mining techniques more action-oriented, e.g., automatically trigger a corrective workflow when a problem emerges.

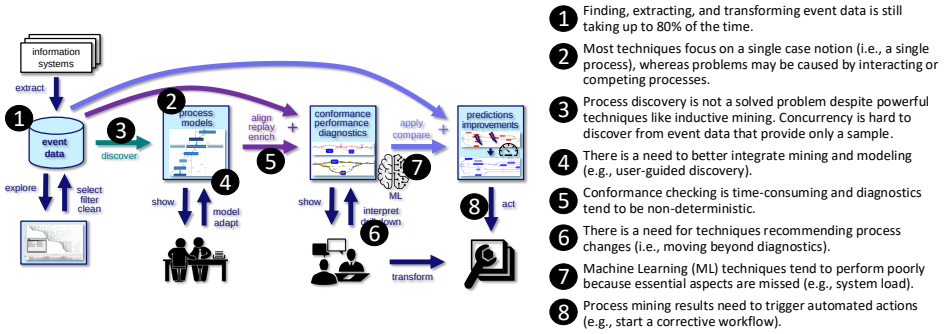

Fig. 5. Some of the challenges encountered when applying process mining.

The process mining manifesto [17] published a decade ago lists 11 challenges. Most of these challenges still exist and are still relevant. Figure 5 maps eight challenges onto the overview used before (Figure 4). These are partly overlapping with the challenges listed in [17], e.g., basic tasks like data extraction and process discovery remain challenging. The reader interested in applications of process mining is recommended to read [26] with experience reports from Siemens, BMW, Uber, ABB, Bayer, and several other organizations.

The top-left corner and bottom-right corner show the interface between the real systems and organization on the one hand and process mining technology on the other

hand. These correspond to the solid arrows in Figure 3(c) used to explain the notion of a digital twin. Using state-of-the-art process mining tools it is possible to create a digital twin with limited scope (e.g., a single process). Process mining is probably the most concrete technology available to create digital twins. Most of the proposals are merely visions or application specific.

3 Process Mining: A Bottom-Up View

After providing a high-level view on process mining, we focus on concurrency and object-centricity. These are essential to create digital twins that properly reflect real organizations. To illustrate these concepts, we use Petri nets. However, it is good to note that the ideas are generic and not notation-specific.

3.1 Petri Nets

Figure 6 shows an accepting labeled Petri net eight places ($p1, p2, \ldots, p8$) and seven transitions ($t1, t2, \ldots, t7$) with initial marking $[p1]$ and final marking $[p8]$. We assume that the reader is familiar with the semantics of Petri nets [5, 13, 25, 27, 28]. However, to make the paper more self-contained, we informally explain the behavior of an accepting labeled Petri net. A transition t is *enabled* in a marking if each of its input places contains at least one token. An enabled transition t may *fire*, i.e., one token is removed from each of the input places $\bullet t$ and one token is produced for each of the output places $t\bullet$. This way the Petri net can move from one marking to the next. For example, in the marking shown in Figure 6 (with a token in $p1$) only $t1$ is enabled. Firing $t1$ means the consumption of one token and the production of three tokens. The resulting marking is $[p2, p3, p4]$. In this marking, four transitions are enabled: There is a choice between $t5$ or $t6$ because both compete for the token in $p4$. The ordering of $t2$, $t4$, and $t5$ or $t6$ is not fixed. The transitions in Figure 6 are labeled, e.g., executing $t6$ correspond to taking an X-ray. Moreover, next to the initial marking indicated by the black token in place $p1$, there is also a final target marking with just a token in $p8$. We are interested in firing sequences leading from $[p1]$ to $[p8]$. Three examples are: $\sigma_1 = \langle t1, t2, t4, t5, t7 \rangle$, $\sigma_2 = \langle t1, t2, t4, t6, t7 \rangle$, and $\sigma_3 = \langle t1, t2, t3, t2, t3, t2, t4, t6, t7 \rangle$. There are infinitely many firing sequences due to the loop. If we block the loop and do not fire transition $t3$, there are 12 possible firing sequences.

Figure 7 shows three example *runs* of the accepting labeled Petri net. Places in Figure 7 correspond to tokens in Figure 6, and transitions in Figure 7 correspond to transition firings in Figure 6. A run of a Petri net corresponds to a partial order. For example, $r1$ in Figure 7 does not impose an ordering on the three middle activities. The transition labels in Figure 7 refer to the transitions in Figure 6, e.g., $t21$, $t22$, and $t23$ in run $r3$ refer to transition $t2$ (administer medicine). For a formal definition of the runs of a Petri net, we again refer to standard literature [12, 14, 27]. Typically, the number of runs is much smaller than the number of firing sequences. For example, if we block the loop and do not fire transition $t3$, then there are only two runs ($r1$ and $r2$) whereas there where 12 possible firing sequences (e.g., σ_1 is one of the six firing sequences corresponding to run $r1$). Run $r3$ corresponds $7 * 6 = 42$ firing sequences.

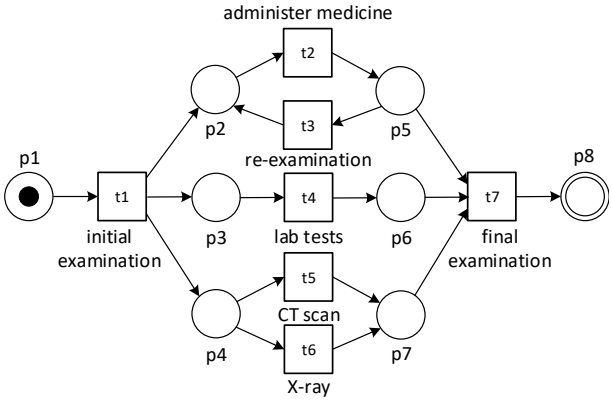

Fig. 6. An accepting labeled Petri net eight places $(p1, p2, \ldots, p8)$ and seven transitions $(t1, t2, \ldots, t7)$.

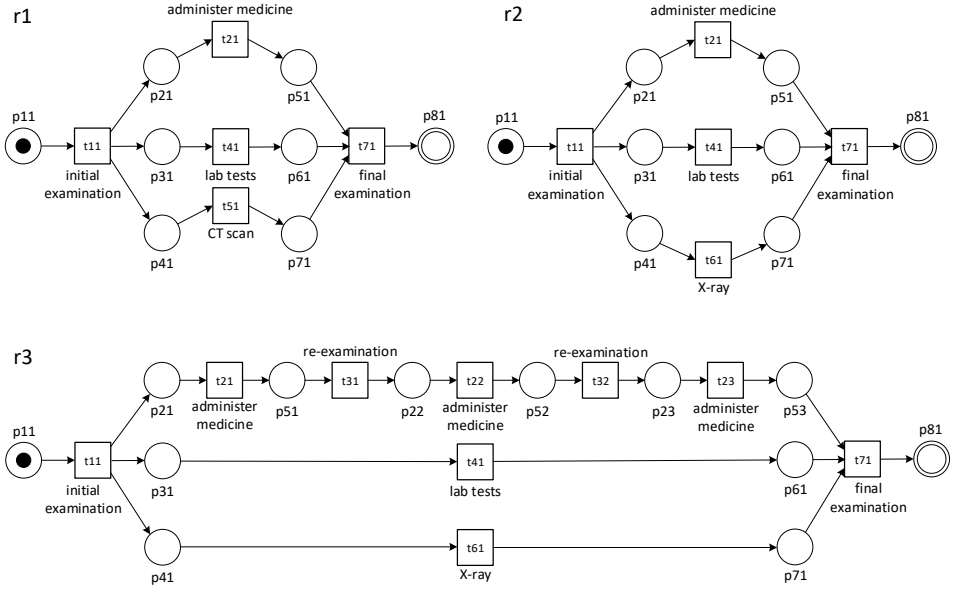

Fig. 7. Three example runs of the accepting Petri net: $r1$, $r2$, and $r3$. Run $r3$ corresponds to 42 firing sequences.

The fact that run $r3$ corresponds to 42 firing sequences illustrates the challenge of discovering concurrency. If we assume that $t3$ is executed at most 5 times, then there are $2(1+1+1+1+1+1) = 12$ runs and $2(13*12+11*10+9*8+7*6+5*4+3*2) = 812$ firing sequences. Even when our event log has information about thousands of traces, it is extremely unlikely that one can witness all 812 variants (especially when not all

variants have an equal probability). This illustrates that one cannot ignore concurrency, because it will lead to an explosion of possible interleavings of which just a fraction will be witnessed.

3.2 Object-Centric Partially-Ordered Event Logs

Next to the problem of concurrency, we also need to deal with events referring to collections of objects. This is analogous to moving from a classical Petri net to a Colored Petri Net (CPN) [5, 18]. In a CPN, tokens have values and can present different objects. In a classical Petri net, tokens are indistinguishable and transitions cannot consumer or produce a variable number of tokens.

Techniques to discover Petri nets from event data assume precisely one case identifier per event [3, 4]. These case identifiers are used to correlate events, and the resulting discovered Petri net aims to describe the life-cycle of individual cases. In reality, there are often multiple intertwined case notions, and it is impossible to pick a single case notion to describe the whole process. For example, events may refer to mixtures of orders, items, packages, customers, and products. A package may refer to multiple items, multiple products, one order, and one customer. Therefore, we need to assume that each event refers to a collection of objects, each having a type (instead of a single case identifier). Such *object-centric event logs* are closer to data in real-life information systems (e.g., SAP, Salesforce, Oracle, etc.). From an object-centric event log, we want to discover an *object-centric Petri net* with places that correspond to object types and transitions that may consume and produce collections of objects of different types. Such object-centric Petri nets visualize the complex relationships among objects of different types.

In the remainder, we present object-centric event logs as defined in [3, 4]. Note that this is a simplified version of the later *OCEL standard* (see `ocel-standard.org`) which also adds attributes to objects [16]. OCEL also provides JSON/XML serializations of object-centric event logs and intends to overcome the limitations of the XES standard [8]. Recall that is the official IEEE standard for storing and exchanging event data assuming a single case notion.

Definition 1 (Universes). *We define the following universes (based on [3, 4]):*

- $\mathbb{U}_{ei}$ *is the universe of event identifiers,*
- $\mathbb{U}_{act}$ *is the universe of activity names (also used to label transitions in an accepting Petri net),*
- $\mathbb{U}_{time}$ *is the universe of timestamps,*
- $\mathbb{U}_{ot}$ *is the universe of object types (also called classes),*
- $\mathbb{U}_{oi}$ *is the universe of object identifiers (also called entities),*
- $type \in \mathbb{U}_{oi} \rightarrow \mathbb{U}_{ot}$ *assigns precisely one type to each object identifier,*
- $\mathbb{U}_{omap} = \{omap \in \mathbb{U}_{ot} \not\rightarrow \mathcal{P}(\mathbb{U}_{oi}) \mid \forall_{ot \in dom(omap)} \forall_{oi \in omap(ot)} type(oi) = ot\}$ *is the universe of all object mappings indicating which object identifiers are included per type,*[1]

[1] $\mathcal{P}(\mathbb{U}_{oi})$ is the powerset of the universe of object identifiers, i.e., object types are mapped onto sets of object identifiers. $omap \in \mathbb{U}_{ot} \not\rightarrow \mathcal{P}(\mathbb{U}_{oi})$ is a partial function. If $ot \notin dom(omap)$, then we assume that $omap(ot) = \emptyset$.

- $\mathbb{U}_{att}$ *is the universe of attribute names,*
- $\mathbb{U}_{val}$ *is the universe of attribute values,*
- $\mathbb{U}_{vmap} = \mathbb{U}_{att} \not\rightarrow \mathbb{U}_{val}$ *is the universe of value assignments,*[2] *and*
- $\mathbb{U}_{event} = \mathbb{U}_{ei} \times \mathbb{U}_{act} \times \mathbb{U}_{time} \times \mathbb{U}_{omap} \times \mathbb{U}_{vmap}$ *is the universe of events.*

An event $e = (ei, act, time, omap, vmap) \in \mathbb{U}_{event}$ is characterized by a unique event identifier ei, the corresponding activity act, the event's timestamp $time$, and two mappings $omap$ and $vmap$ for respectively object references and attribute values.

Definition 2 (Event Projection). *Given* $e = (ei, act, time, omap, vmap) \in \mathbb{U}_{event}$, $\pi_{ei}(e) = ei$, $\pi_{act}(e) = act$, $\pi_{time}(e) = time$, $\pi_{omap}(e) = omap$, *and* $\pi_{vmap}(e) = vmap$.

$\pi_{omap}(e) \in \mathbb{U}_{ot} \not\rightarrow \mathcal{P}(\mathbb{U}_{oi})$ maps a subset of object types onto sets of object identifiers for an event e. An *object-centric event log* is a collection of *partially ordered events*. Event identifiers are unique, i.e., two events cannot have the same event identifier.

Definition 3 (Object-Centric Event Log). $L = (E, \preceq_E)$ *is an event log with* $E \subseteq \mathbb{U}_{event}$ *and* $\preceq_E \subseteq E \times E$ *such that:*

- $\preceq_E$ *defines a partial order (reflexive, antisymmetric, and transitive),*
- $\forall_{e_1,e_2 \in E}\ \pi_{ei}(e_1) = \pi_{ei}(e_2) \ \Rightarrow\ e_1 = e_2$, *and*
- $\forall_{e_1,e_2 \in E}\ e_1 \preceq_E e_2 \ \Rightarrow\ \pi_{time}(e_1) \leq \pi_{time}(e_2)$.

Definition 3 allows for partially ordered event logs. Many process mining techniques require a total order, e.g., events are ordered based on timestamps and when two events have the same timestamp we assume some order. However, there are process discovery techniques that take into account causalities [3, 24]. These can exploit such partial orders. There may be many reasons to use partially ordered event logs: efficiency, imprecise timing information, uncertainty, and explicit partial order information (e.g., based on data flow analysis). As argued before, it is unreasonable to assume that all possible interleavings will indeed be present in the event log. Instead of a partial order one can also use the stricter notion of a weak order. This is particularly suitable when one has imprecise timestamps (e.g., events on the same day cannot be ordered).

3.3 Object-Centric Petri Nets

In this paper, we argued that concurrency and objects matter. To progress the field of process mining, we cannot assume that events are totally ordered and can be correlated using a single case notion. Hence, we need process mining techniques and process model representations handling concurrency and object-centricity as first-class citizens. In [4], we presented an approach to automatically learn a so-called *Object-Centric Petri Net* (OCPN) given an object-centric event log (e.g., in OCEL format [16]). A detailed explanation of the approach to discover OCPNs is beyond the scope of this short paper. Therefore, we only show the example depicted in Figure 8.

[2] $\mathbb{U}_{att} \not\rightarrow \mathbb{U}_{val}$ is the set of all partial functions mapping a subset of attribute names onto the corresponding values.

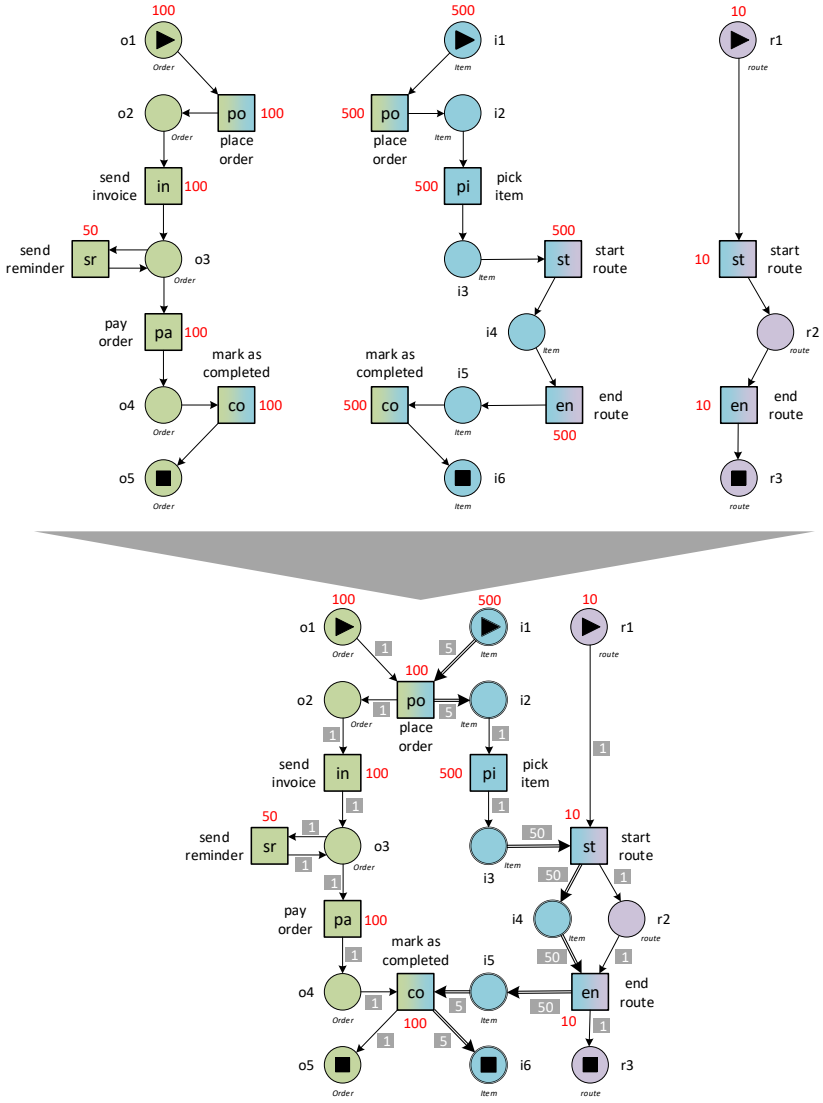

Fig. 8. An Object-Centric Petri Net (OCPN) can be learned by first learning a classical Petri net per object type and then merging the nets while correcting the multiplicities using variable arcs (a detailed derivation of the process model was presented in [4]).

Let $L = (E, \preceq_E)$ be an event log. The events in E refer to objects. Therefore, given a specific object o of type ot, it is possible to create a partial order of all events that refer to o. $(E_o, \preceq_{E_o})$, with $E_o = \{e \in E \mid o \in \pi_{omap}(e)(ot)\}$ and $\preceq_{E_o} = \preceq_E \cap (E_o \times E_o)$, defines the corresponding partial order. Hence, we can group all partial orders of events corresponding to objects of a given type ot to get the required input for

a standard process discovery algorithm. Note that the same event may appear in multiple partial orders. Next, we can learn a process model per object type. For simplicity, we assume that we discover a labeled accepting Petri net per object type satisfying the constraint that labels of visible transition are unique. There may be silent transitions (i.e., transitions that do not refer to an activity). However, there cannot be two transitions referring to the same activity.

The top part of Figure 8 shows three labeled accepting Petri nets discovered for 100 orders, 500 items, and 10 routes. These three models happen to be sequential, but could have been concurrent. The initial and final markings are denoted by the places with the play and stop symbol. Next, the three labeled accepting Petri nets are merged into an OCPN. Since the visible transitions are unique, merging is trivial. However, the annotations need to be modified. In an OCPN there is a one-to-one correspondence between transition firings and events. A single event (i.e., transition occurrence) may involve a *variable number of objects* (e.g., one order may have any number of items). This is indicated by the double arcs in the lower part of Figure 8. For example, on average one execution of *place order* corresponds to five items and one order. On average one execution of *start route* corresponds to 50 items and one route. For more details, we refer to [4].

The discovery Object-Centric Petri Nets (OCPNs) from object-centric event logs in OCEL format is still in its infancy. However, the topic is important because in most applications of process mining one faces the problem of one-to-many and many-to-many relations between different types of objects relevant for an organization. Processes are intertwined and difficult to separate. Figure 8 shows that it is possible to create one, more holistic, process model that is showing the interactions between the different types of objects. Actually, the term "process model" may be misleading in the context of OCPNs that may represent collections of interacting processes.

4 Conclusion

To create a "digital twin of an organization" we need to disentangle the fabric of real operational processes. Process mining provides many of the ingredients to make such a step. In this paper, we provided a high-level overview of process mining and linked it to historical developments in the field of scientific management and simulation. As shown, there have been early examples of digital twins (or at least digital shadows) in the field of process mining. We mentioned, for example, the work combining the process mining framework ProM, the workflow management system YAWL, and CPN Tools as the simulation engine [32]. This enabled new forms of "short-term simulation" that can be used to see the effects of decisions given the current state and historic information.

However, we are far away from fully capturing the fabric of real operational processes in a single model. An important prerequisite is the proper handling of concurrency and entangled objects. One event may refer to many objects and organizations are highly concurrent. It is unrealistic to assume that one can witness all interleavings of highly concurrent processes. Therefore, we elaborated on Object-Centric Petri Nets (OCPNs) and OCEL as a format for exchanging object-centric event logs [16].

Future research needs to address the challenges described in this paper. Compared to the pen-and-paper analyses done by Frederick Winslow Taylor and colleagues more than a century ago, we have booked tremendous progress. The detailed event data available today provide unprecedented opportunities to create digital twins (provided we are able to concurrency and object-centricity properly).

Acknowledgments The author thanks the Alexander von Humboldt (AvH) Stiftung for supporting his research.

References

1. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.
2. W.M.P. van der Aalst. A Practitioner's Guide to Process Mining: Limitations of the Directly-Follows Graph. In *International Conference on Enterprise Information Systems (Centeris 2019)*, volume 164 of *Procedia Computer Science*, pages 321–328. Elsevier, 2019.
3. W.M.P. van der Aalst. Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data. In P.C. Ölveczky and G. Salaün, editors, *Software Engineering and Formal Methods (SEFM 2019)*, volume 11724 of *Lecture Notes in Computer Science*, pages 3–25. Springer-Verlag, Berlin, 2019.
4. W.M.P. van der Aalst and A. Berti. Discovering Object-Centric Petri Nets. *Fundamenta Informaticae*, 175(1-4):1–40, 2020.
5. W.M.P. van der Aalst and C. Stahl. *Modeling Business Processes: A Petri Net Oriented Approach*. MIT Press, Cambridge, MA, 2011.
6. W.M.P. van der Aalst, T.Brockhoff, A. Farhang, M. Pourbafrani, M.S. Uysal, and S. J. van Zelst. Removing Operational Friction Using Process Mining: Challenges Provided by the Internet of Production (IoP). In C. Quix, editor, *Selected Papers of the International Conference on Data Science, Technology and Applications (DATA 2020)*, Communications in Computer and Information Science. Springer-Verlag, Berlin, 2021.
7. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
8. G. Acampora, A. Vitiello, B. Di Stefano, W. van der Aalst, C. Günther, and E. Verbeek. IEEE 1849: The XES Standard - The Second IEEE Standard Sponsored by IEEE Computational Intelligence Society. *IEEE Computational Intelligence Magazine*, 12(2):4–8, 2017.
9. A. Augusto, R. Conforti, M. Marlon, M. La Rosa, and A. Polyvyanyy. Split Miner: Automated Discovery of Accurate and Simple Business Process Models from Event Logs. *Knowledge Information Systems*, 59(2):251–284, May 2019.
10. J. Carmona, B. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking: Relating Processes and Models*. Springer-Verlag, Berlin, 2018.
11. O.J. Dahl and K. Nygaard. SIMULA: An ALGOL Based Simulation Language. *Communications of the ACM*, 1:671–678, Sept 1966.
12. J. Desel. Validation of Process Models by Construction of Process Nets. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 110–128. Springer-Verlag, Berlin, 2000.
13. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

14. B.F. van Dongen, J. Desel, and W.M.P. van der Aalst. Aggregating Causal Runs into Workflow Nets. In K. Jensen, W.M.P. van der Aalst, M. Ajmone Marsan, G. Franceschinis, J. Kleijn, and L.M. Kristensen, editors, *Transactions on Petri Nets and Other Models of Concurrency (ToPNoC VI)*, volume 7400 of *Lecture Notes in Computer Science*, pages 334–363. Springer-Verlag, Berlin, 2012.
15. A. Fuller, Z. Fan, C. Day, and C. Barlow. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, 8:108952–108971, 2020.
16. A.F. Ghahfarokhi, G. Park, A. Berti, and W.M.P. van der Aalst. OCEL Standard. www.ocel-standard.org, 2021.
17. IEEE Task Force on Process Mining. Process Mining Manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer-Verlag, Berlin, 2012.
18. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1*. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
19. M. Kerremans and J. Kopcho. Create a Digital Twin of Your Organization to Optimize Your Digital Transformation Program, Research Note G00379226. `www.gartner.com`, 2019.
20. W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. Digital Twin in Manufacturing: A Categorical Literature Review and Classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.
21. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer-Verlag, Berlin, 2013.
22. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer-Verlag, Berlin, 2014.
23. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, 17(2):599–631, 2018.
24. X. Lu, D. Fahland, and W.M.P. van der Aalst. Conformance Checking Based on Partially Ordered Event Data. In F. Fournier and J. Mendling, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2014)*, volume 202 of *Lecture Notes in Business Information Processing*, pages 75–88. Springer-Verlag, Berlin, 2015.
25. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
26. L. Reinkemeyer. *Process Mining in Action: Principles, Use Cases and Outlook*. Springer-Verlag, Berlin, 2020.
27. W. Reisig. *Petri Nets: Modeling Techniques, Analysis, Methods, Case Studies*. Springer-Verlag, Berlin, 2013.
28. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
29. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
30. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Colored Petri Nets From Event Logs. *International Journal on Software Tools for Technology Transfer*, 10(1):57–74, 2008.

31. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Simulation Models. *Information Systems*, 34(3):305–327, 2009.
32. A. Rozinat, M. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. Fidge. Workflow Simulation for Operational Decision Support. *Data and Knowledge Engineering*, 68(9):834–850, 2009.
33. F.W. Taylor. *The Principles of Scientific Management*. Harper and Bothers Publishers, New York, 1919.
34. S.J. van Zelst, B.F. van Dongen, W.M.P. van der Aalst, and H.M.W Verbeek. Discovering Workflow Nets Using Integer Linear Programming. *Computing*, 100(5):529–556, 2018.