Multi-Instance Mining: Discovering Synchronisation in Artifact-Centric Processes

Maikel L. van Eck[1,⋆], Natalia Sidorova[1], and Wil M.P. van der Aalst[2,1]

[1] Eindhoven University of Technology, The Netherlands
{m.l.v.eck,n.sidorova}@tue.nl
[2] RWTH Aachen University, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. In complex systems one can often identify various entities or artifacts. The lifecycles of these artifacts and the loosely coupled interactions between them define the system behavior. The analysis of such artifact system behavior with traditional process discovery techniques is often problematic due to the existence of many-to-many relationships between artifacts, resulting in models that are difficult to understand and statistics that are inaccurate. The aim of this work is to address these issues and enable the calculation of statistics regarding the synchronisation of behaviour between artifact instances. By using a Petri net formalisation with step sequence execution semantics to support true concurrency, we create state-based artifact lifecycle models that support many-to-many relations between artifacts. The approach has been implemented as an interactive visualisation in ProM and evaluated using real-life public data.

1 Introduction

Process discovery is the automated creation of process models that explain the behaviour captured in event data [1]. Over the years, various algorithms and tools have been developed that support process discovery. However, traditional process discovery techniques are not always suitable for every type of process.

In processes where we can identify *artifacts*, key entities whose lifecycles and interactions define the overall process [6], traditional process discovery techniques often fail [4, 7, 8, 11, 12]. Such *artifact-centric processes* form naturally in business environments supported by information systems based on Entity-Relationship models and databases [3], e.g. a procurement process with sales orders, invoices and deliveries. They can also occur in complex environments, e.g. a medical procedure with surgeons, nurses and medical systems. Additionally, software is often developed according to object-oriented programming paradigms, so processes describing the behaviour of such software systems can also be decomposed into artifacts representing the software objects and components.

The existence of *many-to-many* relationships [3,13] between process artifacts is an important reason why traditional process discovery approaches have difficulties

⋆ This research was performed in the context of the IMPULS collaboration project of Eindhoven University of Technology and Philips: "Mine your own body".

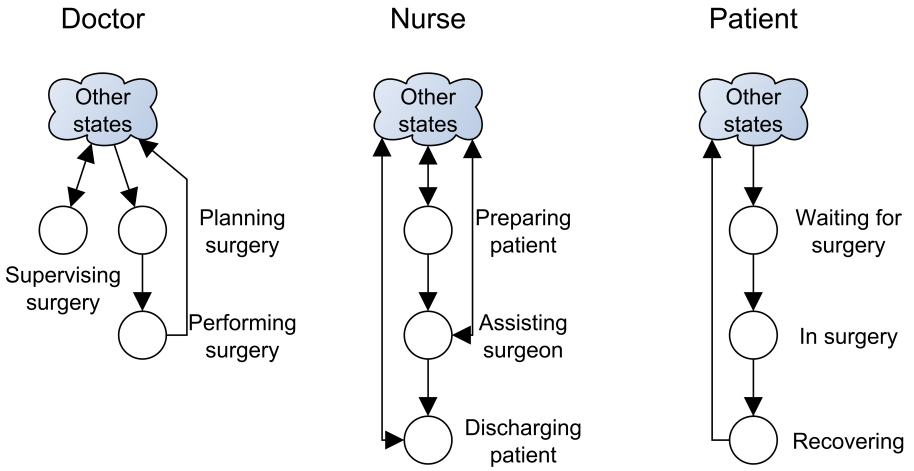


Fig. 1: Partial lifecycle models showing the possible states and transitions between states of three types of artifacts involved in a hospital process: doctors, nurses and patients.

with artifact-centric processes [7,8]. These relationships make it difficult to identify a unique process instance notion to group related events. Enforcing a flat grouping leads to data convergence and divergence problems when calculating statistics related to event occurrences and causal relations between events [8, 13].

Recently, artifact-centric process discovery approaches have been developed that aim to discover models that are not affected by data convergence and divergence issues [7, 8]. However, these approaches have difficulties identifying synchronisation points and flexible interactions between loosely coupled artifacts. Examples of synchronisation points in artifact lifecycles are milestone patterns, e.g. the payment of all invoices before a delivery, and collaborative efforts, e.g. several people meeting to create a project plan.

Consider the example of a simplified artifact-centric hospital process involving doctors, nurses and patients, all modelled as artifacts with their lifecycles shown in Fig. 1. This example process has several possible synchronisation points where people interact: while *preparing patients* two nurses are needed to lift the patient onto the operating table, and the patient can only start *recovering* if the surgeon has finished *performing surgery*. However, the interaction is flexible: the nurses *preparing* a patient are not necessarily the same as those *discharging* the patient, a doctor can be *supervising* multiple surgeries in a single day while doing other things in between, and not every surgery has an additional doctor supervising. Such loosely coupled interactions are difficult to analyse using existing process discovery techniques, e.g. because statistics and relations from the viewpoint of individual doctors are not the same as from the viewpoint of individual patients. Therefore, we have developed an approach to provide accurate statistics and insights in complex artifact-centric processes.

In this paper we describe a process discovery approach suitable for the analysis of synchronous artifact behaviour. This approach builds on ideas presented in [4] for the discovery of state-based models for artifact-centric processes. The state machine formalisation in that work supports the analysis of synchronous artifact behaviour, but only for relations between pairs of artifact instances. We show that by using a Petri net formalisation with step sequence execution semantics to support true concurrency, we can create state-based models that support many-to-many relations between artifacts. This approach has been implemented as a plug-in for the ProM process mining framework and we have evaluated it using the public datasets of the BPI Challenge of 2017.

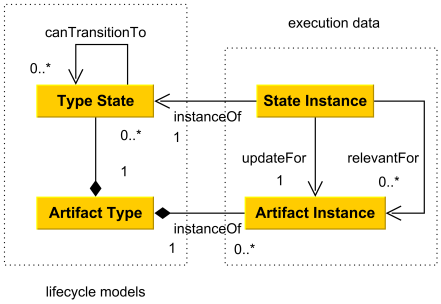
2 Artifact System Modelling

Our goal is to analyse situations where the process of interest involves a number of artifacts interacting. In such an *artifact system* context we distinguish between artifact *types*, e.g. doctors or sales orders, and artifact *instances*, e.g. a specific person working as a doctor. For a given artifact type there is a set of *type states*, i.e. the possible states that artifacts of this type can have. An artifact *lifecycle model* is a graphical representation of an artifact type, its states and the possible transitions between states. The conceptual representation of artifact systems is shown as a class diagram in Fig. 2a.

There exist various modelling languages to describe operational processes [1]. In this work we aim to explicitly analyse the states of artifacts and their interactions. Therefore, we model artifact lifecycles as *state machines*, building on ideas from [4]. State machines are a subclass of Petri nets with the property that each transition has exactly one incoming and one outgoing edge, enabling choice but not concurrency [9]. In a state machine artifact lifecycle model the places represent the type states and a token represents the current state of an artifact instance. To model the beginning and finishing of a lifecycle, we use interface transitions [10] that represent connections to the environment. For simplicity, we represent transitions in lifecycle models using only edges, as in Fig. 1.

Definition 1. *A* state machine *artifact lifecycle model $\mathcal{A}$ is a Petri net tuple (T^c, T^b, T^f, P, E), where T^c is a set of transitions to change states, T^b is a set of input interface transitions, T^f is a set of output interface transitions, P is a finite set of places, $E \subseteq ((T^c \cup T^b) \times P) \cup (P \times (T^c \cup T^f))$ is a set of directed edges between transitions and places, with $\forall t \in (T^c \cup T^b) : \big|\{p \in P | (t,p) \in E\}\big| = 1$, and $\forall t \in (T^c \cup T^f) : \big|\{p \in P | (p,t) \in E\}\big| = 1$.*

We discover artifact lifecycle models based on process execution data containing *state instances*, i.e. moments in time where a specific artifact instance obtains a certain state. In Fig. 2b each row is a state instance updating the state of a specific instance of an artifact in the example hospital process. A mapping of instances to artifact types for this execution data is given in Fig. 2c.



Instance	State	Time	Relevant For
D. Jones	Planning surgery	09:00 - 09:30	S. Hall
D. Jones	Performing surgery	09:30 - 11:30	S. Hall, E. Lee, C. Smith
D. Jones	Writing report	11:30 - 12:30	-
D. Jones	Supervising surgery	12:30 - 13:30	A. James, F. Brown, E. Lee, B. Davis
S. Hall	Waiting for surgery	08:30 - 09:30	-
S. Hall	In surgery	09:30 - 11:30	D. Jones, E. Lee, C. Smith
...	...	...	...

Instance	Type
D. Jones	Doctor
S. Hall	Patient
E. Lee	Nurse
C. Smith	Nurse
A. James	Doctor
F. Brown	Patient
B. Davis	Nurse

(a) (b) (c)

Fig. 2: (a) Class diagram providing a conceptual representation of an artifact system (b) A list of state instances from execution data of the artifact-centric example process (c) A mapping of artifact instances to types

Definition 2. *A log of process* execution data $\mathcal{L}$ *is a tuple* $(\mathbb{S}, \mathcal{I}, \mathbb{A}, S, \mathbb{T}, \mathit{itype})$, *where* $\mathbb{S}$ *is a set of state instances,* $\mathcal{I}$ *is a set of artifact instances,* $\mathbb{A}$ *is a set of artifact types,* S *is a set of type states,* $\mathbb{T}$ *is a time domain, and* $\mathit{itype} : \mathcal{I} \to \mathbb{A}$ *is a mapping from instances to types.*

Each state instance $\varsigma \in \mathbb{S}$ *is a tuple* (ι, s, τ, I), *where* $\iota \in \mathcal{I}$ *is the primary artifact instance that obtains state* $s \in S$ *at time* $\tau \in \mathbb{T}$, *and* $I \subset \mathcal{I}$ *is the set of secondary artifact instances for which it is relevant. The end time of* ς *is an attribute derived by ordering all state instances for* ι *by their timestamp. We assume that* $\forall \iota' \in \mathcal{I}, \tau' \in \mathbb{T} : \left|\{(\iota, s, \tau, I) \in \mathbb{S} | \iota = \iota' \wedge \tau = \tau'\}\right| \leq 1$.

Each state instance can be relevant for a number of secondary artifact instances, which means it can be used to determine synchronisation points or calculate interaction statistics from the perspective of the secondary artifact lifecycles. As shown in Fig. 2, performing surgery involves a patient and nurses, so this state instance can be used to determine per nurse what doctors they work with and for how long. On the other hand, writing reports only concerns one doctor, so it is not relevant from the perspective of other artifact instances. Note that this relation is not necessarily symmetric, e.g. the patient S. Hall is waiting while doctor D. Jones is planning the surgery, but the waiting time is not considered relevant for the doctor's artifact lifecycle.

By modelling artifact lifecycles as state machines there can be no concurrency in the state of a single artifact instance. However, in the context of an artifact system there are multiple interacting artifact instances that each concurrently have a certain state. Therefore, we define a *Composite State Machine* (CSM) Petri net modelling the behaviour of an artifact system as the composition of the lifecycle models of a number of artifact types. Tokens in this Petri net represent artifact instance states, so the combined state of the artifact system is the total marking of the Petri net. The sets of transitions, places, and edges of the CSM are the union of the respective sets of the individual artifact type state machines.

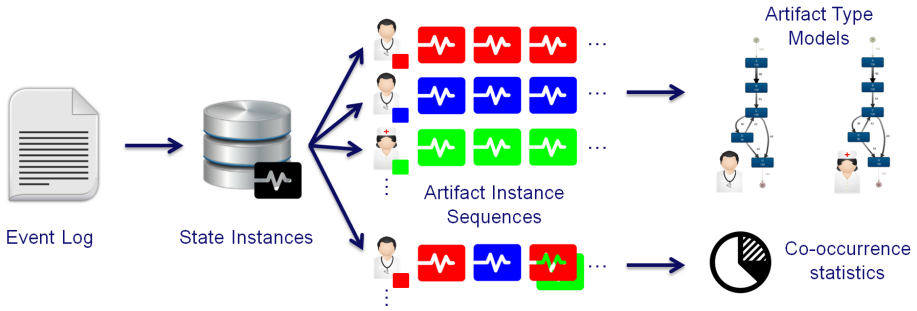


Fig. 3: The overall approach for multi-instance mining of CSMs.

Definition 3. *A* Composite State Machine $\mathcal{M}$ *is a Petri net representing the product of a set of* n *state machines* $\mathcal{A}_1, \ldots, \mathcal{A}_n$. $\mathcal{M} = (T^c_{\mathcal{M}}, T^b_{\mathcal{M}}, T^f_{\mathcal{M}}, P_{\mathcal{M}}, E_{\mathcal{M}})$, *where* $T^c_{\mathcal{M}} = \bigcup_{i\in\{1,\ldots,n\}} T^c_i$ *is the set of transitions representing possible state changes,* $T^b_{\mathcal{M}} = \bigcup_{i\in\{1,\ldots,n\}} T^b_i$ *is the set of transitions where lifecycles begin,* $T^f_{\mathcal{M}} = \bigcup_{i\in\{1,\ldots,n\}} T^f_i$ *is the set of transitions where lifecycles finish,* $P_{\mathcal{M}} = \bigcup_{i\in\{1,\ldots,n\}} P_i$ *is the set of places,* $E_{\mathcal{M}} = \bigcup_{i\in\{1,\ldots,n\}} E_i$ *is the set of edges,*

There is no explicit modelling of synchronisation points in the CSM definition above. Therefore, we model synchronous behaviour by adopting step execution semantics [2,10] instead of the common atomic firing of transitions. Intuitively, a finite multiset of transitions, a *step*, may fire if the artifact instances represented by Petri net tokens can change their state at the same moment in time.

Definition 4. *Given a Petri net with transitions* T*, places* P*, edges* E *and a current state* s *(or marking) as a multiset of tokens* $s \in \mathbb{N}^P$*, then a* step $\mathcal{F} \in \mathbb{N}^T$ *is fireable iff* $\forall p \in P : \sum_{\{t\in T|(p,t)\in E\}} \mathcal{F}(t) \leq s(p)$*, i.e. all input places contain enough tokens. Firing the multiset of transitions in step* $\mathcal{F}$ *results in the state* s' *where* $\forall p \in P : s'(p) = s(p) - \sum_{\{t\in T|(p,t)\in E\}} \mathcal{F}(t) + \sum_{\{t\in T|(t,p)\in E\}} \mathcal{F}(t)$

The use of step execution semantics results in Petri net behaviour with true concurrency, instead of interleaving of transitions [2]. This means that artifact instances can change states independently from each other, but simultaneous state changes can also be analysed to find synchronisation points.

3 Multi-Instance Mining Approach

An overview of the process discovery approach to enable the multi-instance mining of Composite State Machines (CSMs) is shown in Fig. 3.

3.1 State Instance Creation

The executions of many processes are recorded in the form of event logs [1]. To use these logs in our approach they need to be transformed into the format described

in Definition 2. This involves choosing a state representation and determining the set of artifact instances for which a state instance is relevant.

For a given log of process event traces, every position in a trace corresponds to a state of the process [1]. A state representation is a function that, given a sequence of events and a number, produces the process state after the specified number of events in the sequence have occurred. Examples of state representation functions are e.g. the last event executed, the set of events executed, or the full ordered list of events executed. We assume that similar artifact type state representations have been chosen as a preprocessing step, to transform sequences of artifact instance lifecycle events into sequences of state instances.

By linking related artifact instances it is possible to determine the set of relevant secondary artifact instances for a state instance. Such information, relating e.g. doctors to patients or sales orders to deliveries, can be found in relational databases in enterprise settings [3], in ERP systems [8] or in the event log itself [11]. Using domain knowledge, some state instances may get a different set of relevant instances, e.g. they only affect the primary artifact instance.

3.2 Composite State Machine Creation

Given process execution data, we create a time-ordered sequence per artifact instance of all state instances where this artifact is the primary artifact instance. This sequence represents the lifecycle of the given artifact instance.

Individual state machine lifecycle models can be discovered per artifact type by grouping all sequences from instances of the same type and applying the discovery approach described in [4]. The creation of the CSM describing the system is the union of these models, as described in Definition 3.

3.3 Multi-Instance Mining Statistics

We use the notion of a primary instance to group related state instances and calculate accurate sojourn time and co-occurrence statistics in the presence of many-to-many relations. For each artifact instance, we create a time-ordered sequence of state instances where the artifact is either the primary or a secondary instance. That is, given execution data $\mathcal{L} = (\mathbb{S}, \mathcal{I}, \mathbb{A}, S, \mathbb{T}, \mathsf{itype})$ and instance $\iota' \in \mathcal{I}$ we create a sequence from $\{(\iota, s, \tau, I) \in \mathbb{S} | \iota' = \iota \vee \iota' \in I\}$. State instances with the same timestamp τ represent co-occurring transitions, i.e. they are part of a step in the execution of the mined CSM. For example, given the data in Fig. 2b, for artifact instance *S. Hall* there is a transition to *In surgery* co-occurring with a transition from *D. Jones* to Performing surgery. This enables us to calculate conditional co-occurrence statistics similar to those presented in [5], which we map to states and transitions in the lifecycle model of the primary artifact instance.

When the primary instance transitions to a new state, the related instances have a certain state depending on their last known state instance relevant for the primary instance. Together with the related transitions occurring in the same step, this forms a partial view of the system state that we consider as the changing marking of the CSM co-occurring with the primary transition. From

this, we calculate probability estimates conditional on the primary transition: the probability of observing a specific marking or multiset of transitions given the execution of the primary transition, and the probability of having a specific number of instances in a given state when the primary transition is executed.

Similarly, during a system step of related instances not including the primary artifact instance there is a changing partial system marking. From this, we calculate probability estimates for the time spent in certain states co-occurring with the current state of the primary artifact. For example, if the surgery in Fig. 2b had one nurse present for the entire procedure and one nurse present only for the first half hour then based on this the estimated conditional probability for other patients *In surgery* would be that 25% of the time is spent with two nurses in the state *Assisting surgeon* and 75% with one nurse. If the second nurse was e.g. preparing another patient during the later part of the surgery then this is not relevant from the point of view of the current primary patient instance. In this way, the presence of many-to-many relations does not affect the calculation of the conditional co-occurrence statistics.

4 Implementation and Evaluation

In this section we discuss the implementation and evaluation of the Multi-Instance Miner, a plug-in[1] in the open-source process mining framework ProM.

4.1 Multi-Instance Miner

A screenshot of the interactive visualisation of the Multi-Instance Miner is shown in Fig. 4. For each artifact type, a lifecycle model is shown with its type states and transitions. The user can click on states and transitions, which cause the tool to highlight the other states and transitions that can co-occur with the selected element. Moving the mouse to one of the highlighted elements creates a pop-up window that shows how often or for what duration the co-occurrence was observed. Below the main visualisation is a table that provides a detailed list of the statistics mentioned in Sect. 3.3. The user can select what type of co-occurrence relation they want to investigate and filter the results.

4.2 Case Study

The Multi-Instance Miner has been used to analyse the BPI Challenge 2017 data set [14]. This real-life event log concerns a loan application process at a Dutch financial institute.

The event log contains information on the status and the activities performed for all loan applications filed in a single year at the institute. There are 31509 applications, with between 1 and 10 offers related to each application for a

[1] Contained in the *MIMiner* package of the ProM 6 nightly build, available at `http://www.promtools.org/`.

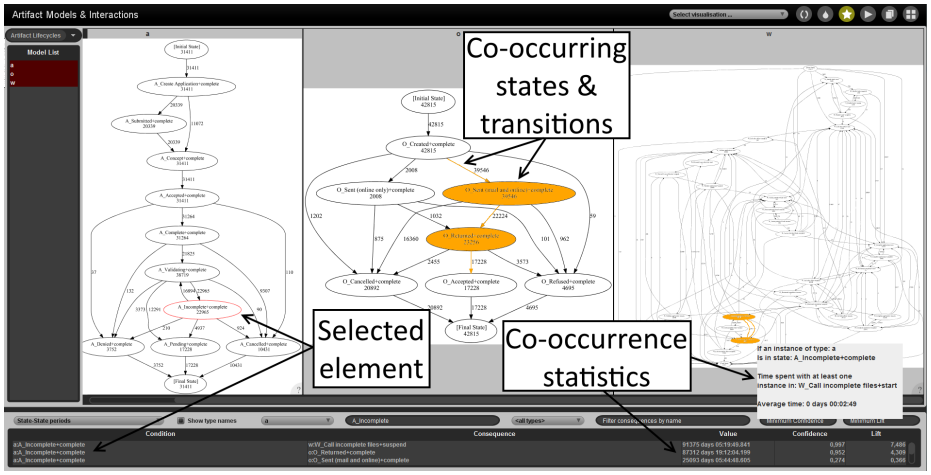


Fig. 4: The interactive visualisation of a discovered CSM. The selected state is circled in red and its co-occurring states and transitions are marked in orange.

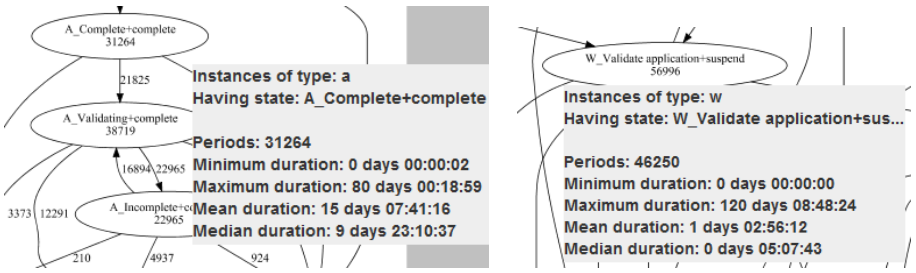
total of 42995 offers in the dataset. We filtered out 98 ongoing applications that did not have a final application status and their related offers and workflow activities. There are 26 distinct activity types, divided into three categories: application state changes (A), offer state changes (O), and workflow activities (W). We consider each category as an artifact type, with application and offer IDs referring to specific instances.

The three lifecycle models are shown in Fig. 4. The application lifecycle includes its creation, complete delivery of the required information, a validation and one of three final results: accepted and pending payment, denied by the institute or cancelled by the customer. The offer lifecycle involves their creation, the sending to and return from the customer, and equivalent final results as for the application. The workflow lifecycle concerns the manual activities performed, so it represents the state of the employee currently working on the application. These activities are changes made to the applications and offers and calls made to contact the customer e.g. regarding missing files. The application and offer models are simple, with the application model containing only a single cycle and the offer model being acyclic. The workflow model is more complex, as workflow activities can be suspended and resumed multiple times before completion.

Explorative analysis revealed that there are many synchronisation points between the different artifacts in this process, which is observable by looking at transitions that are executed in step at the same point in time. For example, Fig. 5 shows the transitions that can co-occur with a transition to the *O_Accepted* state in an offer instance. As expected, if the customer accepts the offer then the application has a simultaneous transition to the *A_Pending* state from either the *A_Incomplete* (28.7%) or the *A_Validating* (71.3%) state. However, there are also 5797 transitions from related offers that are cancelled at the same time. Further

Consequence	Value	Confidence	Lift
a:(A_Incomplete+complete,A_Pending+complete)	4936	0,287	11,217
a:(A_Validating+complete,A_Pending+complete)	12284	0,713	11,217
o:(O_Created+complete,O_Cancelled+complete)	177	0,01	4,855
o:(O_Returned+complete,O_Cancelled+complete)	1409	0,082	8,215
o:(O_Sent (mail and online)+complete,O_Cancelled+complete)	3752	0,218	4,938
o:(O_Sent (online only)+complete,O_Cancelled+complete)	459	0,027	6,802
w:(W_Call incomplete files+resume,W_Call incomplete files+complete)	40	0,002	0,546
w:(W_Call incomplete files+start,W_Call incomplete files+complete)	19	0,001	5,92
w:(W_Validate application+resume,W_Validate application+complete)	3093	0,18	6,907
w:(W_Validate application+start,W_Validate application+complete)	255	0,015	8,829

Fig. 5: The transitions that co-occur with a transition to *O_Accepted*.



(a) (b)

Fig. 6: (a) Sojourn time of applications in state *A_Complete* (b) Sojourn time of the workflow in state *W_Validate application+suspend*

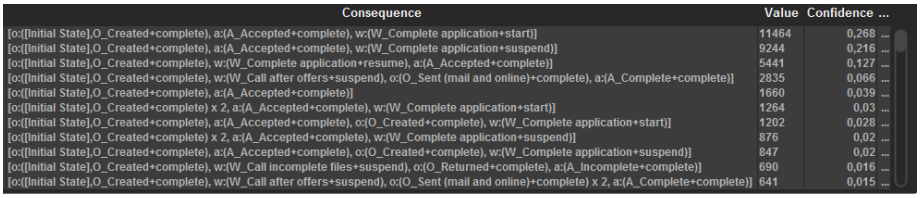
study showed this corresponds to the situation where the customer asked for more than one offer. A logical follow-up analysis would be to check if there are offer characteristics that lead customers to accept one offer over another, in order to potentially make the process more efficient. This shows that explorative analysis can lead to new research questions and possibly ideas for process improvement.

During the challenge, the company was also asking analysts to answer several different business-relevant questions. Two questions we can answer with our tool that they asked were: (1) What are the throughput times per part of the process? and (2) How many customers ask more than one offer, either in a single conversation or subsequently, and how does this affect conversion?

Splitting the log into sequences of state instances for different artifact types separates the events related to employee activities from events related to input by the applicant. As a result, we can determine throughput times that show the difference between the time spent in the company's systems waiting for processing by an employee and the time spent waiting on the applicant. For example, in Fig. 6a there are on average 15 days between the point where an offer is first sent to the customer and the return of a response. By contrast, in Fig. 6b there is on average 1 day and 3 hours between a suspension of the validation due to incomplete files and the first subsequent action e.g. to contact the customer.

Table 1: A count of applications by the number of related offers.

Offers	1	2	3	4	5	6	7	8	9	10
Count	22900	6549	1337	438	125	29	16	12	3	2



Consequence	Value	Confidence ...
[o:([Initial State],O_Created+complete), a:(A_Accepted+complete), w:(W_Complete application+start)]	11464	0,268 ...
[o:([Initial State],O_Created+complete), a:(A_Accepted+complete), w:(W_Complete application+suspend)]	9244	0,216 ...
[o:([Initial State],O_Created+complete), w:(W_Complete application+resume), a:(A_Accepted+complete)]	5441	0,127 ...
[o:([Initial State],O_Created+complete), w:(W_Call after offers+suspend), o:(O_Sent (mail and online)+complete), a:(A_Complete+complete)]	2835	0,066 ...
[o:([Initial State],O_Created+complete), a:(A_Accepted+complete)]	1660	0,039 ...
[o:([Initial State],O_Created+complete) x 2, a:(A_Accepted+complete), w:(W_Complete application+start)]	1264	0,03 ...
[o:([Initial State],O_Created+complete), a:(A_Accepted+complete), o:(O_Created+complete), w:(W_Complete application+start)]	1202	0,028 ...
[o:([Initial State],O_Created+complete) x 2, a:(A_Accepted+complete), w:(W_Complete application+suspend)]	876	0,02 ...
[o:([Initial State],O_Created+complete), a:(A_Accepted+complete), o:(O_Created+complete), w:(W_Complete application+suspend)]	847	0,02 ...
[o:([Initial State],O_Created+complete), w:(W_Call incomplete files+suspend), o:(O_Returned+complete), a:(A_Incomplete+complete)]	690	0,016 ...
[o:([Initial State],O_Created+complete), w:(W_Call after offers+suspend), o:(O_Sent (mail and online)+complete) x 2, a:(A_Complete+complete)]	641	0,015 ...

Fig. 7: A partial view of the state markings that co-occur with a transition to *O_Created*, showing that applications had multiple offers created simultaneously.

Customers can ask for more than one offer, sometimes in the same conversation and sometimes as a result of follow-up calls by the institute. A count of the number of *O_Created* state instances per application is shown in Tab. 1. To determine whether customers ask multiple offers in one conversation or sequentially, we can look at the state marking for all related artifact instances at the time of an offer creation. This view is given in Fig. 7, showing e.g. that there are 11464 offer creations while the related application is in state *A_Accepted* and the related workflow is in state *W_Complete application+start*. From this, we can calculate that given a total of 31411 applications there are at most 3602 applications $(31411 - 11464 - 9244 - 5441 - 1660)$ where the customer initially asked for more than one offer. This information can also be used to determine the conditions under which a subsequent offer is created, e.g. 690 offers were created after the initial offer was returned with insufficient information and 641 offers were created after two initial offers and a follow-up call.

Using the same view it is possible to calculate how the presence of multiple offers and their current state affect the conversion of the application. However, it is difficult to differentiate between offers made simultaneously and those that are created sequentially at later points in the process, as this requires a state abstraction that takes creation history into account. Alternatively, the information on the simultaneous co-occurrences of the offer creation transitions may be used to split the log into applications with simultaneous and applications with sequential offer creations.

5 Related Work

As mentioned in the introduction, several discovery approaches have been developed that can be applied in the context of artifact-centric processes.

Initially, work focussed on enabling the discovery of lifecycles of individual artifact types [12]. These techniques provided lifecycle models and information on which specific artifact instances are related. They did not discover interaction or

synchronisation between artifact instances, which is needed to determine how the lifecycle progression of one instance influences the lifecycle of another instance.

In [11] work was presented to discover synchronisation conditions between artifacts. These synchronisation conditions specify the number of instances of a given type that need to be in a specific state to enable a transition to a specific state for a related instance of another type. As such, the synchronisation points are only discovered for pairs of artifact types, the technique produces no lifecycle models, and the synchronisation conditions do not cover simultaneous state changes in different artifact types.

In [8] an approach was presented to discover causal relations between events for artifact instances related through many-to-many relations from e.g. ERP tables. The resulting lifecylce models show clear causal relations between events from different artifact types, but in settings with loosely coupled artifacts this can result in graphical models with a large number of arcs between the models. This approach does not identify the synchronous execution of activities in different artifacts and unfortunately there is no publicly available implementation.

In [4] we presented a state-based approach for the discovery of artifact interactions. However, the artifact interactions were limited to pairs of artifact instances and only suitable for processes with one-to-one relations between artifact types instead of one-to-many or many-to-many relationships.

In [7] a technique is presented to discover Declare-like constraints with cardinalities on the number of artifact instances involved. This approach supports the calculation of statistics unaffected by convergence or divergence problems and shows many-to-many relations between instances. However, for real-life processes the number of constraints discovered is large and not easy to explore.

6 Conclusion

This paper presented a multi-instance mining approach to discover lifecycle models and their interactions in the context of artifact-centric processes with many-to-many relations between artifact types. In this approach state machine Petri net lifecycle models are discovered from process execution data and combined into a Composite State Machine with true concurrency support through step sequence execution semantics. For the calculation of co-occurrence statistics that identify synchronisation points we used the notion of a primary artifact instance to map state instances onto the lifecycle models.

The developed Multi-Instance Miner supports interactive exploration that allows the user to point at a lifecycle state or transition and see what can happen with a certain estimated probability while an artifact instance is in the selected state or executing the specified transition. It also provides a list of all co-occurrence statistics for more detailed analysis. During the evaluation on the public BPI Challenge 2017 data we were able to answer business-relevant analysis questions and provide starting points for more detailed investigations.

The correlation based statistics are more suitable for loosely coupled artifacts than strict interaction rules, but the list of correlations can grow large. Therefore,

a major challenge remains with visualising the co-occurrence results, e.g. by highlighting the most important results in the models. We also aim to use the approach to analyse the artifact-centric BPI Challenge 2018 data.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Best, E., Devillers, R.R.: Sequential and concurrent behaviour in petri net theory. Theor. Comput. Sci. 55(1), 87–136 (1987)
3. Chen, P.P.: The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. 1(1), 9–36 (1976)
4. van Eck, M.L., Sidorova, N., van der Aalst, W.M.P.: Discovering and exploring state-based models for multi-perspective processes. In: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings. pp. 142–157 (2016)
5. van Eck, M.L., Sidorova, N., van der Aalst, W.M.P.: Guided interaction exploration in artifact-centric process models. In: 19th IEEE Conference on Business Informatics, CBI 2017, Thessaloniki, Greece, July 24-27, 2017, Volume 1: Conference Papers. pp. 109–118 (2017)
6. Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., III, F.F.T.H., Hobson, S., Linehan, M.H., Maradugu, S., Nigam, A., Sukaviriya, P.N., Vaculín, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems, DEBS 2011, New York, NY, USA, July 11-15, 2011. pp. 51–62 (2011)
7. Li, G., de Carvalho, R.M., van der Aalst, W.M.P.: Automatic discovery of object-centric behavioral constraint models. In: Business Information Systems - 20th International Conference, BIS 2017, Poznan, Poland, June 28-30, 2017, Proceedings. pp. 43–58 (2017)
8. Lu, X., Nagelkerke, M., van de Wiel, D., Fahland, D.: Discovering interacting artifacts from ERP systems. IEEE Trans. Services Computing 8(6), 861–873 (2015)
9. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
10. Nielsen, M., Priese, L., Sassone, V.: Characterizing behavioural congruences for petri nets. In: CONCUR '95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings. pp. 175–189 (1995)
11. Popova, V., Dumas, M.: Discovering unbounded synchronization conditions in artifact-centric process models. In: Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. pp. 28–40 (2013)
12. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Cooperative Inf. Syst. 24(1) (2015)
13. Raichelson, L., Soffer, P., Verbeek, E.: Merging event logs: Combining granularity levels for process flow analysis. Inf. Syst. 71, 211–227 (2017)
14. Van Dongen, B.: Bpi challenge 2017 (2017), `https://data.4tu.nl/repository/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`