Discovering and Exploring State-based Models for Multi-perspective Processes

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

Eindhoven University of Technology, The Netherlands
{m.l.v.eck,n.sidorova,w.m.p.v.d.aalst}@tue.nl

Abstract. Process mining provides fact-based insights into process behaviour captured in event data. In this work we aim to discover models for processes where different facets, or perspectives, of the process can be identified. Instead of focussing on the events or activities that are executed in the context of a particular process, we concentrate on the states of the different perspectives and discover how they are related. We present a formalisation of these relations and an approach to discover state-based models highlighting them. The approach has been implemented using the process mining framework ProM and provides a highly interactive visualisation of the multi-perspective state-based models. This tool has been evaluated on the BPI Challenge 2012 data of a loan application process and on product user behaviour data gathered by Philips during the development of a smart baby bottle equipped with various sensors.

1 Introduction

The aim of process mining is to provide fact-based insights into the execution of processes [1,11,13]. An important aspect of this is the discovery of process models based on behaviour captured in event data. These models generally show the activities that can be executed during the process and how they are ordered [15].

One of the most important aspects of process discovery is to deduce the states of the operational process in the log [1]. Many mining algorithms only have an implicit notion of state, i.e. the focus is on learning the ordering of activities [11, 16]. However, process state information may actually be present explicitly in information systems. Examples of such explicit state information are the diagnosis of a patient in a healthcare process or the status of an order in a purchasing process. In this paper we focus on analysing such state information instead of merely focussing on the activities that are executed during a process.

A single process can have different facets, or *perspectives*, each with their own state space. For example, consider the homeostatic process in a person, parts of which regulate sleep and nutrition. From the perspective of sleep the state of a person can be e.g. awake or asleep, while the state of the nutrition perspective

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

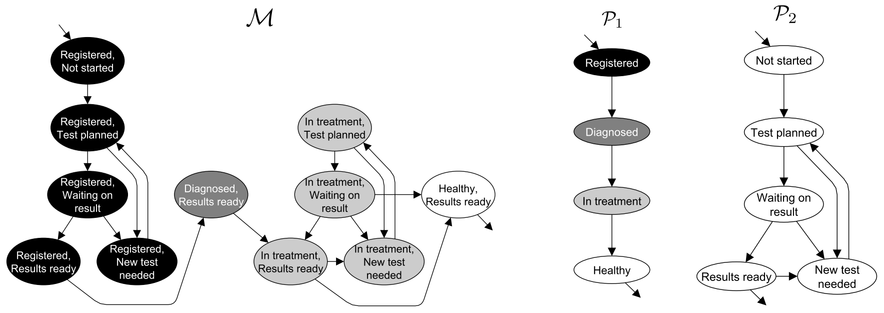


Fig. 1: A model of a simple healthcare process $\mathcal{M}$ and its two perspectives $\mathcal{P}_1$ and $\mathcal{P}_2$. Each state in the process is a combination of a state from each perspective.

can be e.g. eating or sated. These perspectives have individual process cycles, but there are interdependencies between states from different perspectives, e.g. people are awake while eating. The state of a person is the composition of the state of both perspectives, and we aim to study such multi-perspective processes.

In Fig. 1 we present a simple healthcare process, which we use as a running example. This composite process $\mathcal{M}$ has two distinct perspectives: $\mathcal{P}_1$, related to the status of the patient being treated, and $\mathcal{P}_2$, related to the status of lab tests of the patient. The initial states are marked with an incoming arrow and the final states are marked with an outgoing arrow.

The healthcare process starts when the patient is registered, after which a lab test is planned to diagnose the patient. If the patient misses their appointment or if the results are inconclusive then a new test is planned, but if the test results are ready then the treatment can proceed. During the treatment additional tests may be required, until the patient is healthy again and the process ends. Note that the composite process is smaller than the cartesian product of the perspectives ($4 \times 5 = 20$ states) because not all state combinations can be observed due to interdependencies. For example, once the patient is healthy no extra lab tests are needed. Such dependencies between perspectives can be interesting to analyse.

In this paper we present an approach to provide insights into processes that can be considered from multiple state-based perspectives, like the ones described above. The models of these processes quickly become complex as the number of perspectives or the number of states per perspective increases, and it is difficult to interpret the relations between states from different perspectives. Therefore, our approach focusses on visualising and quantifying these relations and empowering the user through interactive exploration of the discovered process models.

The structure of the paper is as follows. In Sect. 2 we formally define state-based models for multi-perspective processes. In Sect. 3 we discuss operations that simplify these models. In Sect. 4 we introduce metrics to quantify the relations between perspectives and we show how they can be visualised. In Sect. 5 we discuss an evaluation of the approach on two real-life data sets. Finally, we present the related work in Sect. 6, and conclusions and future work in Sect. 7.

2 Composite State Machines

We model state-based processes such as the one in Fig. 1 as *Composite State Machines* (CSMs). In this section we first formally describe CSMs and their *perspectives*, based on the finite-state machine formalism [4, 10]. We then define how the state information of a process can be captured in a system execution log and present a discovery algorithm to construct a CSM from such a log. Finally, we define several behavioural relations between process perspectives.

Regarding notation, we write σ_i for the i-th element of a sequence $\sigma \in S^*$ of elements from some set S, and $|\sigma|$ denotes the length of σ. We write $s \in \sigma$ if $s = \sigma_i$ for some i. Additionally, for an element s of a cartesian product $S_1 \times \ldots \times S_n$ we write $s(i)$ for the value of the i-th component of s ($i \in \{1, \ldots, n\}$).

2.1 State Machines and Perspectives

We define *State Machines* (SMs) as follows:

Definition 1. *A* State Machine $\mathcal{M}$ *is a tuple* (S, T, S_0, S_F) *where* S *is the set of states,* $T \subseteq S \times S$ *is the set of transitions,* $S_0 \subseteq S$ *is the set of initial states, and* $S_F \subseteq S$ *is the set of final states.* $(s, s') \in T$ *is also denoted as* $(s \rightarrow s')$.

An *execution sequence* of a state machine is a sequence of states starting from an initial state and ending in a final state such that every state change is allowed by the transitions of the SM. The set of all valid execution sequences of an SM represents the possible behaviour of the process modelled by that SM.

Definition 2. *An* execution sequence $\sigma \in S^*$ *of an SM* $\mathcal{M} = (S, T, S_0, S_F)$ *is a sequence of states such that* $\sigma_1 \in S_0$, $\sigma_{|\sigma|} \in S_F$, *and* $(\sigma_i, \sigma_{i+1}) \in T$ *for* $i \in \{1, \ldots, |\sigma| - 1\}$. *The set* $\Sigma_{\mathcal{M}}$ *is the set of all the execution sequences of* $\mathcal{M}$.

A CSM describes a process with a number of *perspectives*. A state of a CSM is defined as the composition of the states of its perspectives, i.e. it is a vector of states. The set S of all possible states of a CSM is a subset of the cartesian product $S_1 \times \ldots \times S_n$ of the sets of states of its perspectives, as not all combinations of perspective states are necessarily present. Each transition in a CSM represents a change in the state of at least one perspective; therefore we do not consider self loops. Formally:

Definition 3. *A* Composite State Machine $\mathcal{M} = (S, T, S_0, S_F)$ *is a state machine where* $S \subseteq (S_1 \times \ldots \times S_n)$, *with* $S_1, \ldots, S_n$ *being sets of perspective states, and for all* $(s, s') \in T$ *it holds that* $s \neq s'$.

Perspectives of CSMs can be interpreted as projections of a CSM, as seen in Fig. 1. Two states s_i, s'_i of a perspective P_i are connected by a transition iff there is a transition from some state s to a state s' in the CSM that changes the value of the i-th state component from s_i to s'_i. Again, self loops are not considered because transitions represent state changes. Formally:

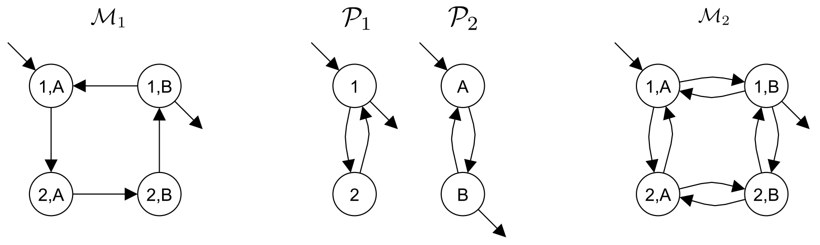


Fig. 2: Two CSMs $\mathcal{M}_1$ and $\mathcal{M}_2$ with $S = \{1,2\} \times \{A, B\}$, $S_0 = \{(1, A)\}$, $S_F = \{(1, B)\}$. Both have the same two perspectives $\mathcal{P}_1$ and $\mathcal{P}_2$.

Definition 4. Perspective $\mathcal{P}_i$ $(i \in \{1, \ldots, n\})$ *of a CSM* $\mathcal{M} = (S, T, S_0, S_F)$ *with* $S \subseteq (S_1 \times \ldots \times S_n)$ *is a state machine* $\mathcal{P}_i = (S_i, T_i, S_{i0}, S_{iF})$ *with* $S_{i0} = \{s(i) | s \in S_0\}$, $S_{iF} = \{s(i) | s \in S_F\}$, *and* $T_i \subseteq S_i \times S_i$ *such that:* $(s_i, s'_i) \in T_i$ *iff* $s_i \neq s'_i \wedge \exists (s, s') \in T : s(i) = s_i \wedge s'(i) = s'_i$.

Figure 2 shows two different CSMs, $\mathcal{M}_1$ and $\mathcal{M}_2$, both having identical perspectives. However, the possibility of changes of the states w.r.t. one perspective of $\mathcal{M}_1$ depends on the state of the other perspective, while in $\mathcal{M}_2$ these changes only depend on the state of a single perspective. E.g. the transition from state 1 to state 2 in perspective $\mathcal{P}_1$ is only possible when $\mathcal{M}_1$ is in state A in perspective $\mathcal{P}_2$, while this transition is independent of perspective $\mathcal{P}_2$ in $\mathcal{M}_2$. This type of dependency shows a relation between the states of different perspectives.

2.2 State Logs and CSM Discovery

The executions of the behaviour of a process can be recorded in a log [1,3–5]. A *state log* describes a collection of sequences with each sequence consisting of the points in time where a process entered a new state. Hence, a *state entry* in the log indicates that the given process has a specific state from the corresponding point in time onwards, until the next *different* state is entered.

Definition 5. *A* trace $\in (S \times \mathbb{T})^*$ *over the state set* S *is a timed sequence of state entries with time domain* $\mathbb{T}$ *such that subsequent state entries differ in their states. A* state log $\mathcal{L} \in \mathbb{N}^{(S \times \mathbb{T})^*}$ *is a multiset of traces over* S.

Given a state log of a process, with every state entry being the product of the states of all perspectives, i.e. entries $(s_1 \times \ldots \times s_n \times t)$, we can discover a CSM describing the process behaviour. We interpret the sequence of state entries in each trace in the log as an execution sequence of the SM being discovered. Time is not used in the discovery of the model, only in the calculation of statistics for the visualisation later. The discovery algorithm is defined as follows:

Definition 6. Discovery algorithm $\mathcal{D}(\mathcal{L})$ *takes a state log* $\mathcal{L} \in \mathbb{N}^{(S \times \mathbb{T})^*}$ *over the set of states* $S = S_1 \times \ldots \times S_n$ *and produces a CSM* $\mathcal{M} = (\widehat{S}, T, S_0, S_F)$ *such that:*

- $\widehat{S} = \{\mathsf{trace}_i(1) | \mathsf{trace} \in \mathcal{L} \wedge (i \in \{1, \ldots, |\mathsf{trace}|\})\}$,
- $T = \{(\mathsf{trace}_i(1), \mathsf{trace}_{i+1}(1)) | \mathsf{trace} \in \mathcal{L} \wedge (i \in \{1, \ldots, |\mathsf{trace}| - 1\})\}$,
- $S_0 = \{\mathsf{trace}_1(1) | \mathsf{trace} \in \mathcal{L}\}$,
- $S_F = \{\mathsf{trace}_{|\mathsf{trace}|}(1) | \mathsf{trace} \in \mathcal{L}\}$,

with $\mathsf{trace}_i(1)$ *denoting the i-th state entry of* trace*.*

This means that each trace in the log is parsed and every state unseen before is added to $\widehat{S}$, and every new pair of consecutive states are added to T. The first state entry in every trace is added to S_0 and the last state entry is added to S_F. The perspectives of the CSM discovered in this way are obtained by projecting the sets of states and transitions, as defined above.

This discovery algorithm corresponds to the first step of the approach presented in [1]. The algorithm in [1] takes an *event* log as input and constructs a transition system. Several different possible abstractions are described that can be used to infer *implicit* states from the events recorded in the log. However, we defined our state log to contain the *explicit* state information of our process, so we do not need to use these abstractions. In fact, mining the log with a horizon limited to single transitions produces a CSM like the algorithm above.

2.3 Behavioural Relations Between Perspectives

Once we have obtained a CSM, we can consider several types of behavioural relations to analyse. Traditional process discovery primarily aims to discover causal relations, i.e. which activity (eventually) follows another [11, 16]. In that context it is more difficult to analyse relations like the expected waiting time between two activity occurrences because of the implicit state notion. However, with an explicit state notion the calculation of time statistics is much easier, while the causal relations are still expressed as transitions between states.

In addition, there are also specific insights that can be of interest related to the interdependencies between perspectives in a multi-perspective process. E.g. for the healthcare process in Fig. 1 one can compare the time required to obtain a result when the patient is not yet diagnosed versus the time required for that when the patient is already in treatment. To enable this, it is necessary to know which states and transitions from different perspectives can be observed to *co-occur*, for which additional statistics can then be calculated.

For a given state of a perspective we consider three relations defining with which states and transitions of another perspective it can co-occur:

Definition 7. *Let* $\mathcal{M} = (S, T, S_0, S_F)$ *be a CSM with* $S \subseteq S_1 \times \ldots \times S_n$ *and* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *its perspectives. For a state* $s_i \in S_i$ *of perspective* $\mathcal{P}_i$ $(i \in \{1, \ldots, n\})$,

- *the co-occurring CSM states are* $CMS_i(s_i) = \{s \in S | s(i) = s_i\}$
- *the co-occurring states of perspective* $\mathcal{P}_j$, $j \neq i$ *are* $CPS_{ij}(s_i) = \{s_j \in S_j | \exists s \in S : s(i) = s_i \wedge s(j) = s_j\}$
- *and the state's co-occurring transitions of perspective* $\mathcal{P}_j$, $j \neq i$ *are* $SCPT_{ij}(s_i) = \{(s_j, s'_j) \in T_j | \exists (s, s') \in T : s(i) = s_i \wedge s'(i) = s_i \wedge s(j) = s_j \wedge s'(j) = s'_j\}$.

The CMS and CPS relations show which combinations of states from different perspectives can be observed in a CSM. For the CSM $\mathcal{M}$ in Fig. 1, e.g. CMS_1(*Diagnosed*) = {(*Diagnosed, Results ready*)}, while the other $\mathcal{P}_2$ states do not occur together with the *Diagnosed* state. The SCPT relation similarly shows which transitions in a specific perspective can be observed when in a given state of another perspective. E.g. SCPT_{21}(*Results ready*) = {(*Registered*→*Diagnosed*), (*Diagnosed*→*In treatment*), (*In treatment*→*Healthy*)}, so all transitions of perspective $\mathcal{P}_1$ are possible when perspective $\mathcal{P}_2$ is in the *Results ready* state.

For a given transition of a perspective we consider three relations linking the transitions of a perspective to the transitions of the CSM and to the states and transitions of other perspectives:

Definition 8. *Let* $\mathcal{M} = (S, T, S_0, S_F)$ *be a CSM with* $S \subseteq S_1 \times \ldots \times S_n$ *and* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *its perspectives. For a transition* $(s_i, s'_i) \in T_i$ *of perspective* $\mathcal{P}_i$ $(i \in \{1, \ldots, n\})$,

– *the co-occurring CSM transitions are* $CMT_i(s_i, s'_i) = \{(s, s') \in T | s(i) = s_i \wedge s'(i) = s'_i\}$,
– *the co-occurring transitions of perspective* $\mathcal{P}_j$, $j \neq i$ *are* $CPT_{ij}(s_i, s'_i) = \{(s_j, s'_j) \in T_j | \exists (s, s') \in T : s(i) = s_i \wedge s'(i) = s'_i \wedge s(j) = s_j \wedge s'(j) = s'_j\}$
– *and the transition's co-occurring states of perspective* $\mathcal{P}_j$, $j \neq i$ *are* $TCPS_{ij}(s_i, s'_i) = \{s_j \in S_j | \exists (s, s') \in T : s(i) = s_i \wedge s'(i) = s'_i \wedge s(j) = s_j \wedge s'(j) = s_j\}$.

The CMT relation gives the set of transitions that contain a specific state change in a given perspective. E.g. in Fig. 1, CMT_2((*Test planned*→*New test needed*)) = {((*Registered, Test planned*)→(*Registered, New test needed*)), ((*In treatment, Test planned*)→(*In treatment, New test needed*))}, so the transition from *Test planned* to *New test needed* is possible at two points in $\mathcal{M}$. The CPT relation gives the transitions that can be observed simultaneously. So, CPT_{12}((*In treatment*→*Healthy*)) = {(*Waiting on result*→*Results ready*)}. Finally, the TCPS relation shows all the states in a perspective where it is possible to observe a specific transition in another perspective. For example, TCPS_{21}((*Test planned*→*New test needed*)) = { *Registered, In treatment* }.

3 Creating Simplified Views for CSMs

The CSMs that are discovered on real life process can be quite complex. Thus it can be desirable to simplify the model in order to focus the analysis on the parts of interest. Therefore, we consider three different operations that create a simplified view on a state machine, i.e. the CSM as a whole or one of the perspectives. These operations take an SM and create a new SM, so multiple operations can be applied in sequence to create a final view.

The first operation removes a given transition from a state machine. This simplifies the model in the sense that the number of arcs is decreased. When creating a view for a perspective of a CSM it is assumed that a similar view is

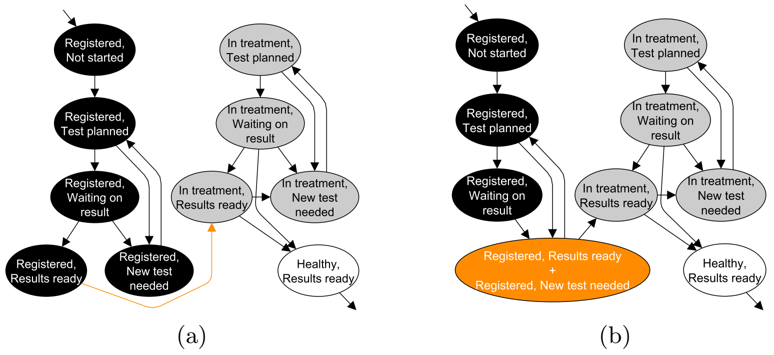

Registered, Not started
Registered, Test planned
Registered, Waiting on result
Registered, Results ready
Registered, New test needed
In treatment, Test planned
In treatment, Waiting on result
In treatment, Results ready
In treatment, New test needed
Healthy, Results ready
(a)
Registered, Not started
Registered, Test planned
Registered, Waiting on result
Registered, Results ready + Registered, New test needed
In treatment, Test planned
In treatment, Waiting on result
In treatment, Results ready
In treatment, New test needed
Healthy, Results ready
(b)

Fig. 3: Two views of $\mathcal{M}$ from Fig. 1. In view (a) the (*Diagnosed, Results ready*) state is abstracted from and in view (b) (*Registered, Results ready*) and (*Registered, New test needed*) are aggregated in addition.

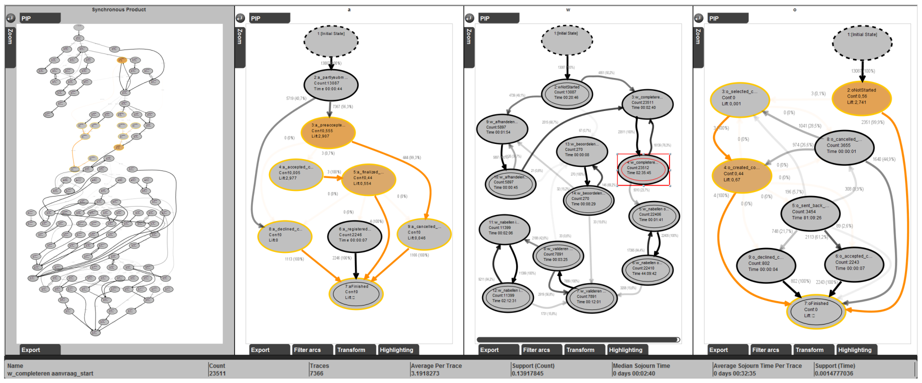
also created for the CSM as a whole. So, if a transition (s_i, s'_i) is removed from perspective $\mathcal{P}_i$ then all transitions from $\mathrm{CMT}_i(s_i, s'_i)$ are also removed.

Removing transitions from an SM affects its behaviour, i.e. the set of allowed execution sequences is reduced. For example, removing the transition (*Waiting on result*$\rightarrow$*New test needed*) from perspective $\mathcal{P}_2$ in Fig. 1 implies that the result of the test is never inconclusive. Note that the transitions ((*Registered, Waiting on result*)$\rightarrow$(*Registered, New test needed*)) and ((*In treatment, Waiting on result*)$\rightarrow$(*In treatment, New test needed*)) should also be removed from $\mathcal{M}$ in Fig. 1 to keep $\mathcal{P}_2$ consistent with $\mathcal{M}$.

The second operation abstracts from a given state in a state machine, simplifying the model by decreasing the number of states. This means that the state is removed, but other states that were connected by transitions through this state should remain connected. In addition, if the abstracted state was an initial or final state then the states that could directly follow or precede this state respectively become initial or final states as well. As an example, Fig. 3a shows the abstraction of the (*Diagnosed, Results ready*) state from $\mathcal{M}$ in Fig. 1. In this view there is now a new transition from (*Registered, Results ready*) to (*In treatment, Results ready*).

Abstracting from states is not guaranteed to simplify the model. If a state is highly connected with many incoming and outgoing transitions then abstracting from this state can result in the addition of many new transitions that make the model more complex.

The third operation aggregates two given states into a single new state, simplifying the model by decreasing both the number of states and transitions. The two old states are removed from the model and a new state is added representing the combination of the two, so all the transitions to and from the old states are also added to the new state (omitting self-loops). If either of the old states was an initial or final state, then the new state is also an initial or final state,



Name	Count	Traces	Average Per Trace	Support (Count)	Median Sojourn Time	Average Sojourn Time Per Trace	Support (Time)
w_completeren aanvraag_start	23511	7366	3.1918273	0.13917845	0 days 00:02:40	0 days 00:32:35	0.0014777036

Fig. 4: The interactive visualisation of a discovered CSM. The selected state is denoted with a red box and its co-occurring states and transitions are highlighted in the other perspectives and the overall view based on their *confidence*. Additional statistics are displayed for the highlighted states and transitions.

respectively. In Fig. 3b the aggregation of the states (*Registered, Results ready*) and (*Registered, New test needed*) from the CSM of Fig. 3a is shown.

Although no behaviour is removed during aggregation due to the preservation of transitions, new behaviour may be added. For example, in the CSM in Fig. 3b it is now possible to go from the state *Registered* to the state *In treatment* without going through the state (*Registered, Waiting on result*), while this was not possible before.

4 Exploring Composite State Machines

In this section we introduce the metrics of *support*, *confidence* and *lift* to quantify the behavioural relations between perspectives. These metrics come from the field of association rule learning [9] and they enable us to highlight relations of potential interest. We also discuss the metric visualisation in the implementation of our approach as a plug-in[1] in the process mining framework ProM.

The visualisation of the discovered CSMs is shown in Fig. 4. The CSM is shown on the left and its perspectives are displayed next to it. Initial states are marked with a dashed border and final states with a double border. Statistics such as the number of observations, are displayed at the bottom for the selected state or transition. The operations from Sect. 3 can be applied to simplify the discovered models. For example, the user can filter arcs based on the amount of observations and iteratively select the states to abstract from or to aggregate.

The behavioural relations introduced in Sect. 2.3 are highlighted and quantified for the selected state or transition. These statistics are calculated from the

[1] Contained in the *CSMMiner* package of the ProM 6 nightly build and the ProM 6.6 release, available at `http://www.promtools.org/`.

observations in the state log that was used to discover the CSM. The reason for the use of highlighting and an interactive display of statistics based on the selected state or transition is to prevent an overload of information and to facilitate the exploration of scenario's. For example, the user can select a specific state in a perspective and evaluate what the occurrence of this state means for the state of the rest of the process and which transitions may be enabled.

The states with the highest *support* are the most frequently observed states. This metric is calculated as the number of observations of a state or transition in the log divided by the total number of observations of states or transitions [9]. That is, if a state s has been observed 40 times and in total there were 100 state entries in the state log then $\mathit{Supp}(s) = \frac{40}{100}$. Similarly, support can also be calculated as the time that was spent in a given state divided by the total time covered by the log. For transitions the support is only defined over the number of observations, as a transition is assumed to happen instantaneously.

The *confidence* metric is defined over pairs of state or transitions, expressing the estimated conditional probability of the occurrence of one, given the occurrence of the other [9]. E.g. if a state s_i from perspective $\mathcal{P}_i$ co-occurs with two states $CPS_{ij}(s_i) = \{s_j, s'_j\}$ from perspective $\mathcal{P}_j$, and if the co-occurrence of s_i with s_j is observed 30 times and the co-occurrence of s_i with s'_j is observed 10 times, then $\mathit{Conf}(s_i, s_j) = \frac{30}{40}$ and $\mathit{Conf}(s_i, s'_j) = \frac{10}{40}$. Confidence for co-occurring states can also be calculated based on the amount of time that was spent in the related states. For pairs of transitions the computation is comparable.

We define the confidence metric slightly differently for the co-occurrence of a state with a transition. For a given state's co-occurring transitions the confidence expresses the expected conditional probability of observing the transition, given that the CSM is in this specific state *and* a transition occurs. For example, a state s_i from perspective $\mathcal{P}_i$ co-occurs with two transitions $SCPT_{ij}(s_i) = \{(s_j, s'_j), (s_j, s''_j\}$ from perspective $\mathcal{P}_j$. Then the confidence $\mathit{Conf}(s_i, (s_j, s'_j))$ is the estimated conditional probability of observing transition (s_j, s'_j), given that the CSM is in s_i and s_j and a transition occurs. That is, if (s_j, s'_j) has been observed 8 times while in state s_i and (s_j, s''_j) has been observed 2 times while in state s_i then $\mathit{Conf}(s_i, (s_j, s'_j)) = \frac{8}{10}$ and $\mathit{Conf}(s_i, (s_j, s''_j)) = \frac{2}{10}$. The confidence of observing a transition's co-occurring state is the expected conditional probability of being in that specific state, given that the transition is observed.

The *lift* metric is also defined over pairs of states or transitions and it expresses how much the confidence differs from the expected confidence [9]. For the co-occurrence of two states s_i and s_j, given that the CSM is in state s_i in perspective $\mathcal{P}_i$, the lift is computed as the ratio of the confidence $\mathit{Conf}(s_i, s_j)$ over the unconditional probability of being in s_j in perspective $\mathcal{P}_j$. E.g. if $\mathit{Conf}(s_i, s_j) = \frac{30}{40} = 0.75$ and the probability of being in s_j in perspective $\mathcal{P}_j$ (i.e. its support) is $\frac{40}{100} = 0.4$, then $\mathit{Lift}(s_i, s_j) = \frac{0.75}{0.4} = 1.875$. This indicates that the probability of being in state s_j in perspective $\mathcal{P}_j$ is 1.875 times higher than expected when in s_i in perspective $\mathcal{P}_i$. In other words, the lift quantifies whether being in s_i provides information on the likelihood of being in s_j and expresses whether the relation is unexpected and hence potentially interesting.

5 Evaluation

The tool introduced in Sect. 4 has been used to analyse two data sets recorded for two real-life processes. One is the BPI Challange 2012 data of a loan application process [6] and the other is product user behaviour data for a smart baby bottle equipped with various sensors that was developed by Philips.

5.1 BPI Challenge 2012

The BPI Challenge 2012 data set (BPI2012) is a real-life event log that was obtained from a Dutch financial institute [6]. The log contains 262.200 events distributed over 13.087 process instances. The process described in this log concerns applications for a personal loan or overdraft at the financial institute. The events recorded in this log are related to three interrelated sub-processes, which we take as our perspectives. Artificial initial and final states were added to each process instance in the log[2] to ensure correct calculations of state sojourn times for all three perspectives.

The first perspective concerns the state of the application (A-events), the second relates to the work-items performed by the bank's employees (W-events), and the third concerns the state of the institute's offers to the applicant (O-events). Although the BPI2012 log is presented as an event log, the A and O-events actually specify changes in the state of the application or an offer. This means that they can be interpreted as state entries in a state log as defined in Sect. 2.2. On the other hand, the W-events are clearly identifiable as activities. These activities are enabled at some point in the process, indicated with a single *schedule* event, after which the *start* and *completion* of each instance of this activity is recorded whenever it is performed. At most one activity is performed per application at a time, so we study the process from the viewpoint that the states of the work-item perspective indicate either the type of activity currently being executed (i.e. indicated by a start event) or the type of activity that was most recently completed (i.e. indicated by a complete event).

The interrelation of these three perspectives introduces complex behaviour that makes it difficult for traditional process discovery algorithms to discover informative models. Fig. 5 shows a process model discovered by the Inductive visual Miner (IvM) [11]. This flower model provides very little insights into the application process and no relations between the three perspectives. On the other hand, models such as the one shown in Fig. 6, discovered with the Flexible Heuristics Miner (FHM) [16], do show these relations, but they provide little structure and they are difficult to interpret.

Applying the CSM Miner on the BPI2012 results in the models shown in Fig. 4. The discovered CSM is shown as the leftmost model and, like the result from the FHM, it is very difficult to interpret. However, the three models for the individual perspectives are well structured and easy to comprehend. Mining such structured models is also possible with traditional process discovery algorithms if

[2] Available at `http://svn.win.tue.nl/repos/prom/Packages/CSMMiner/Logs/`

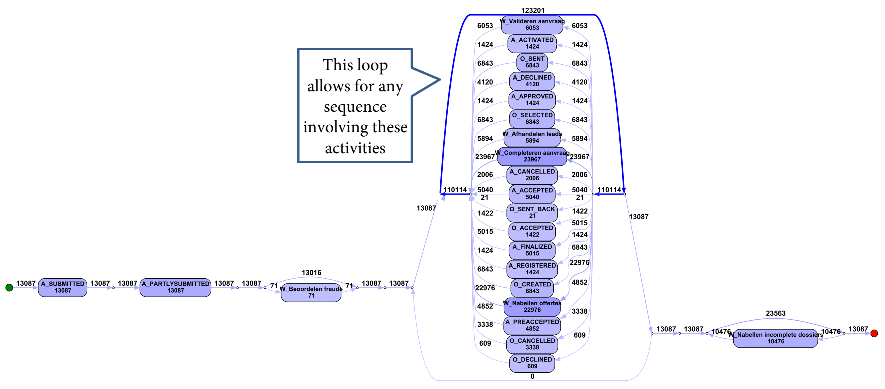


Fig. 5: The result of applying the IvM [11] on the BPI2012 data. The relations between the events from different perspectives are not visible in this model.

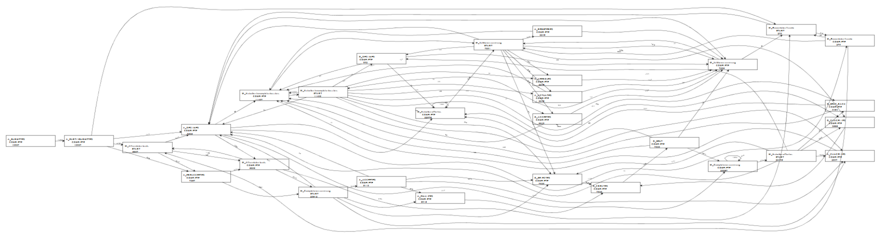


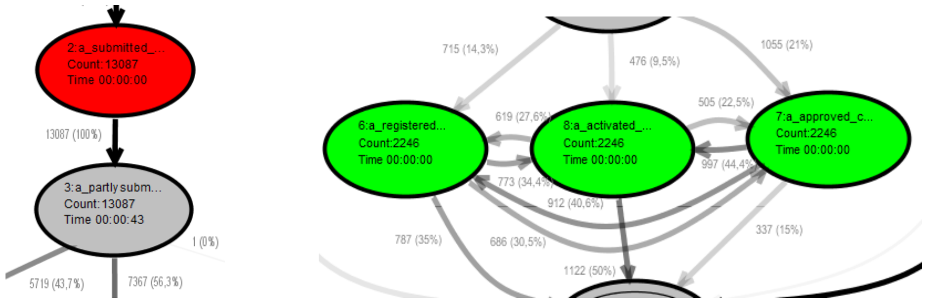
Fig. 6: The result of applying the FHM [16] on the BPI2012 data. The general flow of the process cannot easily be inferred from this model.

the log is filtered for a specific perspective. *However, the resulting models would not show any of the interrelations between the perspectives.* The CSM Miner does show these relations when the user explores the models interactively.

Another reason why the CSM Miner results are easier to interpret is that they can be simplified using the transformation operations from Sect. 3. E.g. the BPI2012 process starts with the submission of the application, shown in Fig. 7a. This is always immediately followed by a state indicating that the application is not completed yet, i.e. partly submitted. Based on the fact that the time spent in the first state is negligible, this state can be abstracted from.

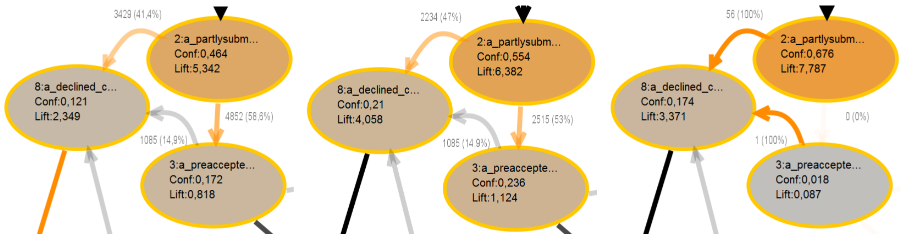
Fig. 7b also shows a transformation simplifying the application perspective. It contains the end of the application process for successfully accepted applications, which are approved, registered and the loan is activated. The model structure and state statistics indicate that these states occur in arbitrary order and that the process immediately ends afterwards (i.e. the state sojourn time is 0). Hence, these states can be merged into a single one representing a successful application.

Exploring the discovered CSM provides several interesting insights. For example, when inspecting the declined applications that co-occur with automatic processing (Fig. 8a) and comparing them to the declined applications that co-



(a) (b)

Fig. 7: Part of the *application* perspective with *a_submitted* marked for abstraction, and *a_registered*, *a_activated* and *a_approved* marked for aggregation.



(a) (b) (c)

Fig. 8: The decline of applications by the institution, highlighted to show statistics for the co-occurrence with automatic application submission (a), manual handling of leads (b), and fraud detection (c).

occur with manual handling of the leads (Fig. 8b). These statistics show that the rate of declined applications is very similar for both type of applications. So, this suggests that perhaps the guidelines for declining applications are uniform, but some applications may come in through a channel where it is not possible to automatically evaluate them on the application.

Exploration of the CSM also shows that some applications are declined while the institution is investigating potential fraud, shown in Fig. 8c. In the 75 cases where fraud is investigated for applications that have not been validated, 57 were declined (76%). However, after application validation there were also 33 cases where potential fraud was investigated, but none of these were declined. Therefore, the validation appears to be successful at filtering out fraud and it suggests that they only investigate applications that have not been validated for which they already have a suspicion of possible fraud.

Finally, it is also possible that an application is declined after it has been validated and the offer sent to the applicant. This appears to be related to unresponsive applicants or incomplete applications. In Fig. 9a the co-occurrence statistics are shown for the state where the client is called because of an incom-

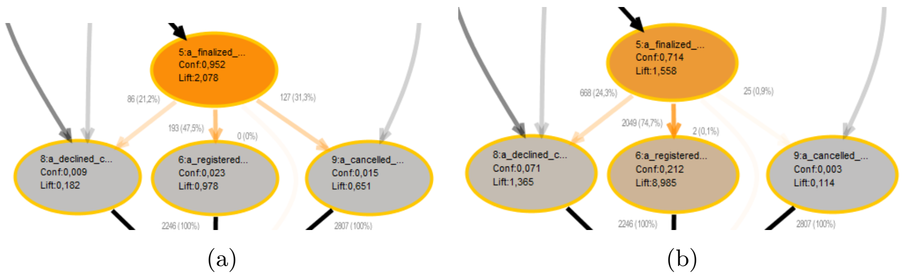


Fig. 9: Status changes of the application after it is *finalized*, while the client is called due to an incomplete application (a), or during application validation (b).

plete application. While on the phone, only $\frac{86+193+127}{1647} = 25\%$ cases changed state, so most people cannot provide the required information right away. The number of successfully registered applications at this point is also lower than for the other applications that reach this point in the process, as shown in Fig. 9b.

Interestingly, exploration also revealed that on average 7 calls are made for the cases where information is incomplete, suggesting that these clients are not taking the effort to complete the application even after being contacted. Therefore, the institution could investigate the trade-off between the value of additional successful applications and the required effort for these incomplete applications. To get more insights into this, it would also be useful to see the acceptance rate for the applications that are incomplete. However, this information cannot currently be obtained as the relations between the different perspectives are limited to co-occurrence and do not show (long term) dependencies.

5.2 Smart Product User Behaviour

This data set was obtained from Philips during a study where Philips worked on the design of a smart baby bottle equipped with various sensors. The goal of the study was to investigate the characteristics of the data obtained during the use of the bottle, and to explore potential product improvements or ideas for related services based on analysis of this data.

The data set used during this evaluation concerns 358 instances of baby feedings that resulted in 8369 state entries in a state log. There are two perspectives used in the analysis: a temperature sensor and an accelerometer measuring bottle movement. The states in this log correspond to the state of the sensor signals of these two sensors and their product-specific interpretation. They were obtained by clustering the sensor measurement values and labelling the cluster centroids. The resulting CSM is shown in Fig. 10 The main challenge of analysing this sensor data is the recognition of user behaviour and its effects on the measurements.

One of the basic assumptions on user behaviour for this smart bottle is that a feeding is started soon after the bottle has been heated. Figure 11a shows the bottle movement states, highlighted to indicate their co-occurrence with the

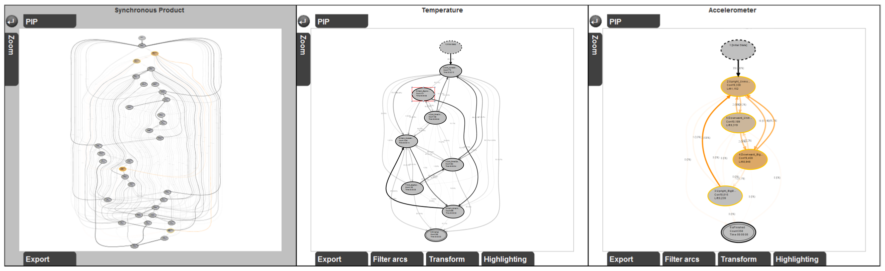
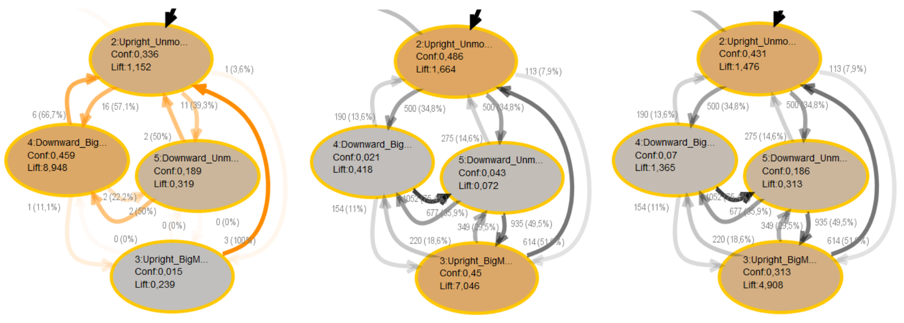


Fig. 10: The result of applying the CSM Miner on the Philips data set.



(a) (b) (c)

Fig. 11: States in the *accelerometer* perspective, highlighted for the co-occurrence with the bottle heating (a), cooling (b), and small temperature increases (c).

state of the bottle having just been heated. The high lift of the state *Downward_BigMove* shows that the feedings are indeed generally started soon after heating, as this state is an important indicator for the start of a feeding.

Similarly, Fig. 11b shows the co-occurrence of bottle movement states with the transition from a warm bottle to a cold bottle through a big decrease in temperature. Here the lift of the state *Upright_BigMove* is high, indicating a relation with this indicator of the feeding having ended. This shows that the bottle is usually cleaned soon after a feeding has ended.

Interestingly, there is also a strong relation between a small temperature increase and the *Upright_BigMove* state, as shown in Fig. 11c. This occurs because during the feeding the warm food was further away from the sensor than when the bottle was in a stationary position, resulting in fluctuating temperatures. The product designers inferred from this that the temperature sensor was not in the correct position to measure the temperature accurately during the feeding.

6 Related Work

The discovery of state-based models from logs of behavioural data is not a recent idea [4,5]. Finite state machines have been found to be convenient to model historical patterns of behaviour in different contexts, e.g. to understand software behaviour [5, 12] or to find successful proof strategies for interactive theorem provers [8]. These approaches are similar to traditional process discovery approaches that produce a single model describing the observed behaviour [1,11,16].

More recently, processes have been studied from the point of view of the *business objects* or *artifacts* involved in a process [13–15],e.g. orders or invoices. In this context, state machines have been traditionally used to model the individual lifecycles of artifacts, although more specific formalisms have also been developed [14]. While artifacts are generally defined to include both an information model with all data related to the artifact and a lifecycle model describing how events and activities affect the state of the artifact, a perspective is only a collection of related states and the transitions that are possible between these states in the context of a state-based process. To the best of our knowledge, there is currently no publicly available implementation of an artifact-centric approach that can discover the interactions between objects or artifacts [13, 14].

Systems composed of multiple state machines have also been studied in the areas of model checking and software analysis [3, 7]. The individual state machines are generally assumed to either operate independently or interact using messages [2, 10]. Few approaches can discover models for such systems and they model interaction through message passing instead of the relations we study [3].

The formalisms we introduce for CSMs are similar to notions from automata theory [2], but while automata theory is centered on actions or activities, we deviate from the dominating activity view in process mining and utilize available state information explicitly. In automata theory notions of e.g. product automata can be used to build up a system of multiple automata based on synchronised transitions [2], which can be reduced to create a minimal automata, i.e. bottom-up construction. However, a CSM cannot be built up from its perspectives as there is no information in the data of individual perspectives to synchronise on. This information is only available in the log of the entire process, which is mined to directly create a composite model that is minimal by definition.

7 Conclusion

This paper presented an approach to discover state-based process models that can be interactively explored. We first formally defined the notion of a Composite State Machine as a way to model multi-perspective processes that can be learned from event logs. As the resulting models can be quite complex, we provided three different operations that can be used to create simplified views on state machines.

To explore the discovered models we have developed an interactive visualisation tool that is available as a plug-in for ProM. The tool highlights interesting relations between states and transitions graphically and quantifies them in terms of *support*, *confidence* and *lift*. This tool has been evaluated on two real-life data sets, demonstrating that valuable and novel insights can be obtained.

Future work we plan to do in this area aims at improving practical usability. For example, the view creation operations could be automatically evaluated to provide the user with feedback on the changes in process model quality when creating a new view. Based on the existing metrics or on concurrency detection there could also be automatic suggestions for candidate transitions and states for removal or aggregation. Finally, the approach should be extended to support more types of behavioural relations between the perspectives. In addition to co-occurrence, it is also interesting to look at the dependencies between perspectives that occur before or after reaching a given state.

References

1. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and System Modeling 9(1), 87–111 (2010)
2. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
3. Beschastnikh, I., Brun, Y., Ernst, M.D., Krishnamurthy, A.: Inferring models of concurrent systems from logs of their behavior with csight. In: 36th International Conference on Software Engineering, ICSE '14. pp. 468–479 (2014)
4. Biermann, A.W., Feldman, J.A.: On the synthesis of finite-state machines from samples of their behavior. IEEE transactions on Computers 21(6), 592–597 (1972)
5. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. 7(3), 215–249 (1998)
6. van Dongen, B.F.: BPI Challenge 2012 (2012), `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`
7. Fisler, K., Krishnamurthi, S.: Modular verification of collaboration-based software designs. In: Proceedings of the 8th European Software Engineering Conference 2001. pp. 152–163 (2001)
8. Gransden, T., Walkinshaw, N., Raman, R.: Mining state-based models from proof corpora. In: Intelligent Computer Mathematics - International Conference, CICM 2014 Proceedings. pp. 282–297 (2014)
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer (2001)
10. Kam, T., Villa, T., Brayton, R.K., Sangiovanni-Vincentelli, A.: Synthesis of finite state machines: functional optimization. Springer Science & Business Media (2013)
11. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Exploring processes and deviations. In: Business Process Management Workshops - BPM 2014 International Workshops, Revised Papers. pp. 304–316 (2014)
12. Lorenzoli, D., Mariani, L., Pezzè, M.: Automatic generation of software behavioral models. In: 30th International Conference on Software Engineering (ICSE 2008). pp. 501–510 (2008)
13. 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)
14. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Cooperative Inf. Syst. 24(1) (2015)
15. Ryndina, K., Küster, J.M., Gall, H.C.: Consistency of business process models and object life cycles. In: Models in Software Engineering, Workshops and Symposia at MoDELS 2006, Reports and Revised Selected Papers. pp. 80–90 (2006)
16. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011. pp. 310–317 (2011)