Process Mining for Semantic Business Process Modeling

Florian Lautenbacher, Bernhard Bauer and Sebastian Förg
Programming Distributed Systems Lab
University of Augsburg
Augsburg, Germany
{*lautenbacher, bauer, foerg*}*@ds-lab.org*

***Abstract*—Business processes are captured by models that serve as a basis for communication and training purposes, but this modeling is still a time consuming manual job. Semantic annotation of process models in combination with AI planning approaches can contribute to solve this drawback enabling an automatic creation of process models. But the semantic annotated process fragments necessary for starting the planning are often missing at all or not up-to-date anymore. Therefore, this work describes an approach for the semantic annotation and semantic-based planning of process models and introduces `Cystid`, an integration of Process Mining algorithms and semantic-based planning.**

***Index Terms*—Process Mining; SBPM; Process Models**

I. Introduction

Our competitive world requires companies to adapt their processes frequently in order to develop new products or advanced services and to react to changing laws and regulations. These processes are often captured and described by models that serve, for instance, as a basis for communication and training inside and across companies or are used as a starting point for implementing processes in a service-oriented way. But process modeling still has one major drawback: It is a time-consuming manual job. A semantic annotation of process models as part of the research area Semantic Business Process Management (SBPM) [1] in combination with AI planning approaches can contribute to solve this drawback enabling an automatic creation of process models through semantic-based planning [2].

The requirement for this planning is a library (lib_A) which contains process models, actions and fragments that are semantically annotated in terms of (at least) their input and output parameters. At the moment this process library lib_A is manually filled. However, there are already process models in a company (mostly we do not have a greenfield approach!) - but are these models still up-to-date or does the design and execution differ in practice? Have all processes been covered by models or are some processes missing?

Luckily, companies often already have executed a huge amount of processes using their information systems (such as ERP, CRM or workflow systems). A lot of these typically support event logging in order to store which tasks have been executed for a given process instance, the order in which these tasks have been performed, etc. As a logical consequence, if a company does not want to model their processes or review all process models again, the data from these event logs needs to be utilized. Process Mining is the research area that is concerned about discovering new knowledge based on information in event logs (and more specifically process logs).

On the other side also Process Mining still has obstacles: The analysis of process logs is purely syntactic as the mining algorithms are unable to reason over the concepts behind the labels in the log. Utilizing ontologies [3] the relations between the concepts in the ontologies can be used to aggregate tasks and compute hierarchical process models including several abstraction levels. Additionally, if there are missing or corrupt log files (a problem generally known as *noise*, cf. [4]), then the mining algorithms can not reconstruct a complete process model, but only fragments of it. The same problem would occur with *hidden tasks* where the log files are all existent and not corrupt, but the tasks simply were not recorded in the log. Hence, manual work is necessary again.

If a company would execute their processes with a predefined process definition format on a process engine, then the deployed files could be used for this purpose, of course. But not all IT systems use process engines, instead they often execute the processes directly in their systems. Therefore, the planning of processes is necessary in order to come to the original processes again.

Henceforth, the contributions of our work are the following: we combine the ideas of Process Mining on the one hand and semantic-based process planning on the other hand. With Process Mining we reconstruct as many process actions and fragments as possible, annotate them afterwards with semantic information and start then an automatic planning algorithm in order to compute a correct process model for one or more predefined goals. By this, the drawbacks of both areas can be minimized. Therefore, we introduce our approach of annotating process models which is the basis for a semantic-based planning of process models.

The remainder of this paper is organized as follows: After a short introduction about (Semantic) Business Process Management and Process Mining in section II, we give an overview about the semantic annotation and planning of process models. Afterwards, we introduce our algorithm `Cystid` which combines both areas in Section IV. We show the advantages and limitations of our approach by a case study (Section V), before we describe some related work and conclude this paper.

II. Basics

A. Business Process Management

A *Business Process* consists of a set of activities that are performed in coordination with an organizational and technical environment. These activities jointly realize a business goal. A business process is typically enacted by a single organization, but it may interact with processes performed by other organizations [5]. Additionally, there also exist cross-organizational business processes (CBPs) that span over several companies.

Business processes are mostly captured in process models. A process modeling language is called *declarative* when it explicitly takes into account the business concerns that govern business processes and leaves as much freedom as is permissible at execution time for determining a valid and suitable execution scenario [6]. Declarative process models are mostly goal-driven whereas procedural modeling languages are state-driven.

A *Workflow* on the other side is concerned with the automation of a business process where information and tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal. Whilst a workflow may be manually organized, in practice most workflows are normally organized within the context of an IT system to provide computerized support for the procedural automation [7].

A workflow can formally be denoted as a directed graph $\mathcal{G}$ (compare also [8]). $\mathcal{G}$ is denoted by ($\mathcal{N}$,$\mathcal{E}$), where $\mathcal{E}$ is the set of edges and $\mathcal{N}$ the set of nodes. $\mathcal{N}$ consists of the disjoint subsets: $\mathcal{N}_{start}$, $\mathcal{N}_{stop}$, $\mathcal{N}_{action}$, $\mathcal{N}_{fork}$, $\mathcal{N}_{join}$, $\mathcal{N}_{decision}$, $\mathcal{N}_{merge}$. The terms $action, decision$, etc. are thereby used in analogy to UML activity diagrams [9].

Each node $n \in \mathcal{N}$ has a set of incoming and outgoing edges $\mathcal{E}_{in}(n)$ and $\mathcal{E}_{out}(n)$ respectively. The source of an edge $\mathcal{E}(n)$ is described via $\mathcal{N}_{source}(\mathcal{E}(n))$, the target as $\mathcal{N}_{target}(\mathcal{E}(n))$. Further the graph $\mathcal{G}$ has to satisfy the following conditions:

- $\mathcal{N}_{start}$ and $\mathcal{N}_{stop}$ have exactly one element (n_{start}, n_{stop}), such that $|\mathcal{E}_{in}(n_{start})| = 0 \wedge |\mathcal{E}_{out}(n_{start})| = 1$ (called *entry edge* e_{entry}) and $|\mathcal{E}_{out}(n_{stop})| = 0 \wedge |\mathcal{E}_{in}(n_{stop})| = 1$ (called *exit edge* e_{exit})).
- $\forall n \in (\mathcal{N}_{fork} \cup \mathcal{N}_{decision})$: $|\mathcal{E}_{in}(n)| = 1 \wedge |\mathcal{E}_{out}(n)| >= 2$, $\forall n \in (\mathcal{N}_{join} \cup \mathcal{N}_{merge})$: $|\mathcal{E}_{in}(n)| >= 2 \wedge |\mathcal{E}_{out}(n)| = 1$ and $\forall n \in \mathcal{N}_{action}$: $|\mathcal{E}_{in}(n)| = 1 \wedge |\mathcal{E}_{out}(n)| = 1$.
- $\forall n \in \mathcal{N} : \exists$ path p in $\mathcal{G}$: $n \in p$ (all nodes are reachable from the start).

B. Process Mining

The goal of Process Mining [10] is to discover, monitor and improve executed processes (workflows) by extracting knowledge from event logs. It assumes that already happened events have been recorded and that these records include the following: each event refers to an action, a specific instance of a process, can have a performer and should have a timestamp. In the best case all events that happened have been stored in the correct order.

Process Mining has been developed in the Data Mining area with the purpose to extract knowledge from unstructured data. Already ten years ago Cook and Wolf [11] developed methods to discover software processes in event logs. Agrawal, Gunopulos and Leymann [12] were the first to use process mining techniques for workflow management. During the last years buzzwords such as Business Activity Monitoring (BAM) or Business Process Intelligence (BPI) have been developed which essentially mean the same as Process Mining [13]. Nowadays three different categories of Process Mining can be distinguished: *discovery* of models, testing the *conformance* of models and logs and the *extension* of existing models [14].

For the discovery and creation of models several algorithms have been developed, the most prominent one being the α-Algorithm [15]. It can be used for the modeling of workflow nets (an extension of Petri nets with concepts and notations that ease the representation of business processes [5]). Therefore, the algorithm computes transitions for the workflow net by working through the process logs. It calculates whether existing actions have been executed in sequence, in parallel or alternative paths according to the following definitions:

- $n_a >_W n_b$, iff there is a path $p = \{n_1, n_2, \dots, n_m\}$ and $i = \{1, \dots, m-1\}$ such that $p \in \mathcal{G}, n_i = n_a$ and $n_{i+1} = n_b$ (n_b follows n_a).
- $n_a \rightarrow_W n_b$, iff $n_a >_W n_b$ and $n_b \not>_W n_a$ (causal relationship: n_a always appears before n_b).
- $n_a ||_W n_b$, iff $n_a >_W n_b$ and $n_b >_W n_a$ (n_a and n_b are executed in parallel).
- $n_a \#_W n_b$, iff $n_a \not>_W n_b$ and $n_b \not>_W n_a$ (n_a and n_b are not related at all).

Based on these definitions, the α-Algorithm creates a workflow net in eight steps (described in more detail in [15]):

1) Compute all actions from the process log
2) All process actions that start a process instance are stored in an array
3) All process actions that define the end of a process instance are stored in another array
4) A transition matrix for all elements that do not invoke themselves is created
5) Refinement of the matrix in removing unnecessary actions. A tree is created with root showing the start element
6) Every existing relation between elements in the tree defines a place in the Petri net and is created
7) All places get connected with transitions
8) Returning the finalized Petri net

ProM [13] has been one of the first frameworks that has been developed for Process Mining. Already in version 5.0 this framework developed by the Process Mining Group and the Technical University of Eindhoven includes several algorithms (such as the α-Algorithm) and provides an open plattform for which already more than 230 plugins have been developed. It includes plugins for the actual mining, several import and export plugins, conversion plugins as well as analysis plugins.

C. Semantic Business Process Management

Business Process Management (BPM) has been one of the main topics in commercial information technology for many years and is becoming even more important now. The graphical modeling of business processes and their processing into software products requires so much human work that the production cycles can not comply with the fast changing demands of today's global markeets. To improve the degree of automatic processing in BPM, techniques from the Semantic Web such as ontologies and reasoners have been transferred to the business process world. A semantic annotation of process management (e.g. process models) as envisioned in the research area Semantic Business Process Management (SBPM) [1] aims to reduce the current drawbacks in the creation of process models, execution, monitoring and maintenance of processes.

As pointed out by Thomas and Fellmann [16], the semantics of meta-model elements for process modeling and their relations are already defined by well established approached for process and enterprise modeling like ARIS. However, the terms used to specify individual model elements (e.g. the name of a particular function or an input parameter) and their semantics are still left to the modeler. Problems in comprehension and ambiguities are the consequence of inconsistently used terms. By means of ontologies, terms in process models can be conceptualized and their relations are technically defined. This allows an advanced and automatic processing of semantically annotated process models and their elements.

III. SEMANTIC ANNOTATION AND PLANNING

A. Semantic Annotation of Process Models

A *Semantic Business Process Model* describes a set of activities including their functional, behavioral, organizational, operational as well as non-functional aspects. These aspects are not only machine-readable, but also "machine-understandable" which means that they are either semantically annotated or already in a form which allows a computer to infer new facts using the underlying ontology.

We define *annotation* formally as a function that returns a set of concepts from the ontology for each node and edge in the graph, $SemAn : \mathcal{N} \cup \mathcal{E} \rightarrow C_{Onts}$. $SemAn$ describes all kinds of semantic annotations which can be input, output, metamodel annotation, etc. The semantic annotation can either be done manually or computed automatically considering word similarities, see e.g. [17]. We can now define a *semantic annotated graph* $\mathcal{G}_{sem} = (\mathcal{N}_{sem}, \mathcal{E}_{sem}, Onts)$ with $\mathcal{N}_{sem} = \{(n, SemAn(n)) | n \in \mathcal{N}\}$ and $\mathcal{E}_{sem} = \{(n_{sem}, n'_{sem}) | n_{sem} = (n, SemAn(n)) \wedge n'_{sem} = (n', SemAn(n')) \wedge (n, n') \in \mathcal{E}\}$. C_{Onts} is a set of concepts of (possibly different) ontologies of the set of ontologies $Onts$ ($C_{Onts} \subseteq Onts$).

An *ontology* $Ont \in Onts$ is a "(formal) explicit specification of a (shared) conceptualization" [3] and in our context defined as a quadruple $Ont := (C, R, I, A)$ which consists of different classes C and relations R between them. A relation connects a class either with another class or with a fixed literal. It can define subsumption hierarchies between classes or other relationships. Additionally, classes can be instantiated with a set of individuals I. An ontology might also contain a set of axioms A which state facts (what is true) in a domain. Please note that [3] actually speaks of "classes, relations, functions and other objects", whereby current languages of the semantic web such as OWL [18] also include individuals and axioms.

For semantic annotation to be usable an approach is required that is independent of a modeling language and modeling tool, but which can be easily integrated in existing tools. It shall be usable, i.e. once a model element is selected the corresponding annotations shall be shown. It shall support several ontologies and stay consistent with the model once an identifier is changed.

In order to achieve this we use an aspect-oriented mechanism. In aspect-oriented modeling (e.g. [19]) there is a difference between positive and negative variability [20]. Negative variability is about removing optional parts from a given structure, whereas positive variability is about adding optional parts to a given core. Our approach utilizes positive variability. We define external aspects that are woven into the core metamodel. This metamodel can be each language as long as the language conforms to a given meta-metamodel. Hence, we do not change the original metamodel or modeling tool, but only define the additional aspects that are automatically shown in the original modeling tool.

The semantic-based planning requires that all process actions include a semantic annotation. Let A be the set of all process actions. A process action $a \in A$ is characterized by a set of input parameters $In_a \in P$, which is required for the execution of a and a set of output parameters $Out_a \subseteq P$, which is provided by a after execution. All elements $a \in A$ are stored as a triple $(name_a, In_a, Out_a)$ in a process library lib_A.

P is the set of all input and output parameters with Dom describing the set of domains of these parameters. Here, a domain describes the range of possible values a parameter can adopt according to its type. A single parameter p (with $p \in P$) can be either atomic - it consists of a triple (l_p, dom_p, r_p) then - or it is a composite parameter. l_p thereby specifies the identifier of the parameter p, $dom_p \in Dom$ the domain and r_p the restriction on the domain. The label and domain are concepts defined in an ontology or in other type definitions with the label being more concrete than the domain (e.g. label `amount` has domain `integer`).

This domain is either a primitive data type (cp. data types in XML-Schema [21]) or an ontological class. Additionally, each process action defines an individual restriction r_p for each of its input and output parameters p (with $r_p \subseteq dom_p$ if dom_p is a primitive datatype resp. $r_p \sqsubseteq dom_p$ if dom_p is an ontological class). If for instance a process action is only executable for a proper subset of the domain dom_p (in the case of primitive data types) the input parameter p is restricted to $r_p \subset dom_p$.

For restrictions we follow the approach of [22] who add

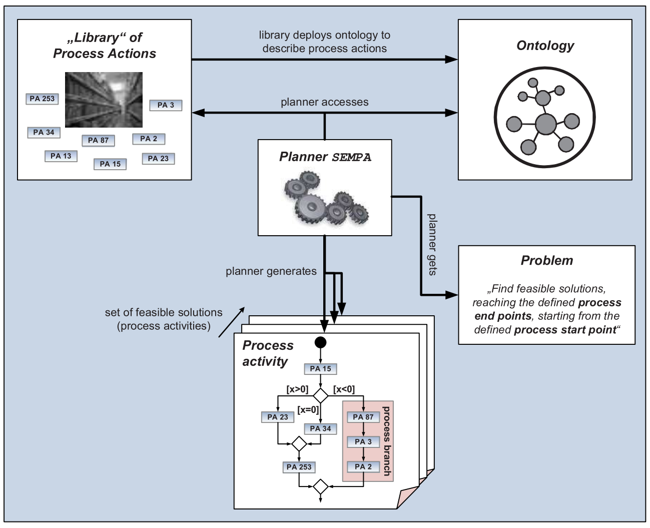
„Library“ of Process Actions
PA 253
PA 3
PA 34
PA 87
PA 2
PA 13
PA 15
PA 23
library deploys ontology to describe process actions
Ontology
planner accesses
Planner SEMPA
planner gets
Problem
„Find feasible solutions, reaching the defined process end points, starting from the defined process start point“
planner generates
set of feasible solutions (process activities)
Process activity
PA 15
[x>0]
[x<0]
[x=0]
PA 23
PA 34
PA 87
PA 3
PA 2
PA 253
process branch

Fig. 1. The SEMPA Approach [2]

constraints to attributes of web services. Their approach describes that each attribute constraint has the form *attribute-name attribute-type constraint-expression*. The *attribute-type* can be String, Integer, Date, etc. and the *constraint-expression* has a keyword (enumeration such as {“EUR”, “US-$”, “AUD”, “CHF”, “GBP”} or range (100 to 500)) or a comparison operator ($\leq, \geq, =$, etc.) and a value list. Similar to them our approach also has *label domain restriction* for the description of each parameter whereas we also allow ontology concepts as domain (and not only simple datatypes).

B. Semantic-based Planning

In the project SEMPRO (funded by the German Research Foundation) a (semi-)automatic creation and adjustment of process models via automatic planning of process models using semantic process actions is intended. The authors speak of semi-automated because the planned process models (which are created fully automatically) are considered as proposals that need to be assessed by an expert regarding business aspects. It provides a goal-driven and declarative way of process modeling.

The developed planning algorithm SEMPA (SEMantic-based Planning Approach, cf. Figure 1) proceeds in three steps. First, input and output parameters of process actions and fragments stored in the process library (lib_A) are semantically matched and the information concerning the action dependencies are stored for the following steps [23]. Second, a forward-search collects all applicable processes from the graph that can help to achieve a certain goal [2]. This basis is used in the last step to create the process model by using control structures and syntactical elements of a specific modeling notation, e.g. UML activity diagrams.

The division of the algorithm into three parts has a couple of advantages:

- As the number of available process actions may be significantly large, the division allows for a reduction of complexity while efficiency can be increased. Process actions that can never be part of a feasible solution are separated out in the first step and are not processed in further steps. Accordingly, by means of the semantic dependency analysis the search space for the following forward search can be significantly reduced. Moreover, dependencies analyzed once can be reused for further planning problems.
- Semantic reasoning is a complex and thus time-consuming task, especially when different data types and restrictions are involved. Therefore, semantic reasoning is accomplished only once (in the first step) which avoids redundant analysis during the following steps of the algorithm.
- The planning graph forms a (language independent) basis for the extraction of process models. Characteristics of process modeling languages concerning for instance control structures are only considered in the last part. This keeps the flexibility to consider different process modeling languages at a time without adapting the fundamental search algorithm. On the other hand it provides a possibility to develop different search algorithms without affecting the extraction of process models.

IV. Process Mining for Semantic Business Process Models

The basic idea to use Process Mining for Semantic Business Process Models is to assist the user in the task of filling the process library lib_A which is used for planning a process model later on.

Especially during the migration from the present where everything is modeled manually and no semantic annotation is used to a time in the future where all process models are semantically annotated and planned, this is crucial to persuade people that Semantic Business Process Modeling is usable at all. These people do not want to go through all their existing process models and store each action or even fragments of a process model into a process library. This must be achieved automatically. That might seem like quite an easy task if all processes have been modeled correctly and one only needs to go through all processes and save them. But in reality the processes are sometimes completely different than the modeled processes [24]. Additionally, manually created process models often include errors: [25] showed that in 600 process models under consideration enacted at SAP, at least 34 contained errors.

Process Mining techniques allow the discovery of existing models, testing the conformance between models and log files and an extension of models based on information derived from log files. However, these mining techniques are unable to reason over the concepts behind the labels in the log [26]. I.e., only business manager understand the meaning of the concepts and why a process has been designed as it is.

In our approach we try to reuse as much from existing frameworks as possible. E.g. the existing α-Algorithm is implemented as part of the ProM-framework. Therefore, the gen-

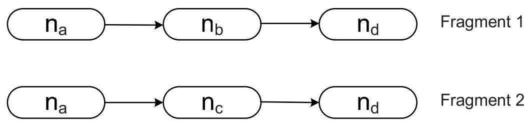

Fig. 2. Process Fragments

eral parts of ProM that are independent of any algorithm need to be provided in other frameworks, too. The α-Algorithm creates a Petri net, whereas our planning result is currently displayed as UML activity diagrams. That is why we adapted the α-Algorithm, included the general parts of ProM and created a new algorithm that we called `Cystid`. On contrast to the α-Algorithm `Cystid` recognizes existing fragments in the process library and extends or adapts these (e.g. in adding control nodes where necessary). In the following we will describe the advantages of our algorithm `Cystid` :

- `Cystid` is adapted to UML Activity diagrams: It does recognize workflow patterns [27] such as Sequence, And Split, Xor Join, etc. and builds the corresponding control structures (DecisionNode, MergeNode, etc.).
- `Cystid` can be used to create a new process model as well as extend an existing one.
- It allows the user to semantically annotate process actions or in particular the first and last action of a process fragment.
- It builds on algorithms and data structures already implemented in ProM and as part of the planner `SEMPA`.

Thereby, `Cystid` works on a pure syntactical basis and does only at the end allow the user to annotate the process fragment semantically. The algorithm consists of four steps:

1) The log files are loaded and tupels are created.
2) The content of the process library is added to the tupels.
3) Process fragments including control structures are build considering the tupels.
4) Process fragments are semantically annotated.

In the first step tuples of the form $T = (\{A\}, \{B\})$ (with $A, B \subseteq \mathcal{N}$) of all sequences of actions that have been found in the process log are created. These tuples store in the beginning only the execution order of actions in the process log.

In the second step these tuples are adapted and also store the process actions of the process library. The tuples then store whether a process action has one following node, several alternative following nodes or several parallel nodes.

Consider the two fragments shown in Figure 2. Fragment 1 has already been added and fragment 2 is now under consideration. Created tuples from the first found fragment are $(\{n_a\}, \{n_b\})$ and $(\{n_b\}, \{n_d\})$. Since n_d does already exist, n_c is investigated (line 5 sqq). There already exists a tuple with n_d as the target but only with n_b as source. Therefore, n_c is added to the already existing tuple $(\{n_b, n_c\}, \{n_d\})$ (line 7). If n_d had been part of the model with a merge node before, then the algorithm considers also the model elements that come before the merge node (line 14 sqq.). Similarly, a join node before needs to be investigated similar to the action node (lines 23 sqq). In the same way also the nodes following are considered (line 37).

Algorithm 1 `Cystid` - Adding Process Library to Tupels

```
1: for all $a \in lib_A$ do
2:   if $\mathcal{E}_{in}(a)$ defined then
3:     $b := \mathcal{N}_{source}(\mathcal{E}_{in}(a))$
4:     if $b$ ofType $\mathcal{N}_{action}$ then
5:       for all Tuple $t = (B; A) \in T$ with $a \in A$ do
6:         if $b \notin B$ then
7:           $t = (B \cup \{b\}; A)$
8:         end if
9:         tupleFound := true
10:      end for
11:      if $\neg$ tupleFound then
12:        $T = T \cup (\{b\}; \{a\})$
13:      end if
14:    else if $b$ ofType $\mathcal{N}_{merge}$ then
15:      $C := \bigcup \mathcal{N}_{source}(\mathcal{E}_{in}(b))$
16:      for all tuples $t = (B, A) \in T$ with $a \in A$ do
17:        $t = (B \cup C; A)$
18:        tupleFound := true
19:      end for
20:      if $\neg$ tupleFound then
21:        $T = T \cup (C; \{a\})$
22:      end if
23:    else if $b$ ofType $\mathcal{N}_{join}$ then
24:      for all $c := \mathcal{N}_{source}(\mathcal{E}_{in}(b))$ do
25:        for all Tuple $t = (B; A) \in T$ with $a \in A$ do
26:          if $c \notin B$ then
27:            $t = (B \cup \{c\}; A)$
28:          end if
29:          tupleFound := true
30:        end for
31:        if $\neg$ tupleFound then
32:          $T = T \cup (\{c\}; \{a\})$
33:        end if
34:      end for
35:    end if
36:  end if
37:  $b := \mathcal{N}_{target}(\mathcal{E}_{out}(a))$
38:  similar to above
39: end for
```

Afterwards (in the third step), these tuples are studied in order to create the correct control structure (Algorithm 2). If the tuple only consists of one source and one target element, then it will get a simple sequence. The only exception is the case where there already exists a following action after the source element in the process model. Then a fork node needs to be created and the corresponding edges need to be added (line 9 et sqq.). If there are two source elements in the tuple, then a merge node needs to be created, if there are two elements in the target of the tuple then a decision node must be created (line 24).

When all process actions and control structures have been created applying the information of the log file, then all contiguos fragments are semantically annotated (fourth step) whereby `Cystid` proposes ontology concepts based on String comparison between the name of the process action and classes in the ontology. In addition to this name-based matching also a process context-based matching could be performed. These ontology concepts are then used for process planning (for an overview of the different steps of `Cystid` combined with `SEMPA` see Figure 3).

V. Case Study and Evaluation

For a proof of concept we chose an example from the financial services industry. The basic idea for this example has

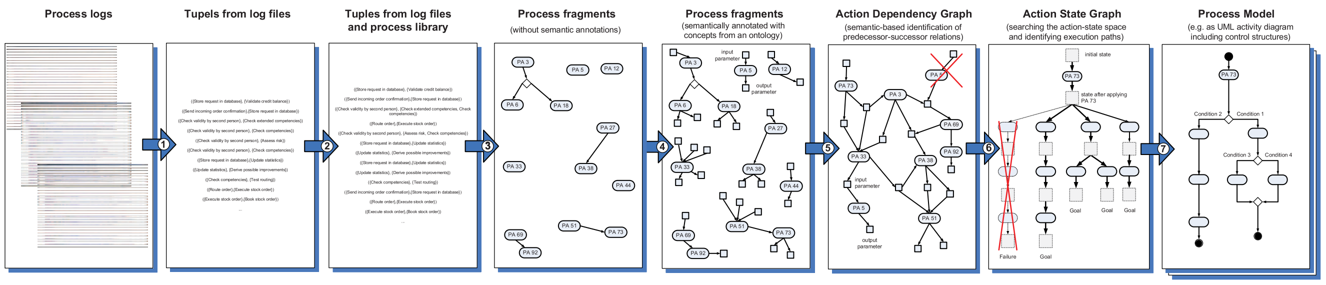
Process logs
Tupels from log files
Tuples from log files and process library
Process fragments
(without semantic annotations)
Process fragments
(semantically annotated with concepts from an ontology)
Action Dependency Graph
(semantic-based identification of predecessor-successor relations)
Action State Graph
(searching the action-state space and identifying execution paths)
Process Model
(e.g. as UML activity diagram including control structures)
1
2
3
4
5
6
7

Fig. 3. Process Mining for Semantic Business Process Modeling

Algorithm 2 *Cystid* - Building Process Fragments including Control Structures

```
1: for all Tuple $t = (A, B) \in T$ with $A, B \subseteq \mathcal{N}$: do
2:   if $|A| = 1 \wedge |B| = 1$ then
3:     if $|\mathcal{E}_{out}(a)| > 0$ then
4:       $c := \mathcal{N}_{target}(\mathcal{E}_{out}(a))$
5:       if $c$ ofType $\mathcal{N}_{action}, \mathcal{N}_{decision}, \mathcal{N}_{merge}$ or $\mathcal{N}_{join}$ then
6:         if $c \notin B$ then
7:           create $\mathcal{N}_{fork}$ $f$
8:           $\mathcal{N}_{source}(\mathcal{E}_{in}(c)) := f$
9:           create edge between $a$ and $f$ as well as between $f$ and $b$
10:        end if
11:      else if $c$ ofType $\mathcal{N}_{fork}$ then
12:        create edge between $c$ and $b$
13:      end if
14:    else
15:      create edge between $a$ and $b$
16:    end if
17:    if $|\mathcal{E}_{in}(b)| > 0$ then
18:      ...(similar)
19:    end if
20:  else if $|A| \geq 2 \wedge |B| = 1$ then
21:    Create $\mathcal{N}_{merge}$
22:  else if $|A| = 1 \wedge |B| \geq 2$ then
23:    Create $\mathcal{N}_{decision}$
24:  else if $|A| \geq 2 \wedge |B| \geq 2$ then
25:    Create $\mathcal{N}_{merge}$ with a following $\mathcal{N}_{decision}$ afterwards
26:  end if
27: end for
28: Store all elements in the process library $lib_A$
```

been extracted from a real life case which has been conducted at a large financial service institution, but has been highly simplified for this paper. Our sample process describes the processing of an order, e.g. a stock order.

In our process library we already know that once an order request has been received and stored in a database, the credit balance of the requester needs to be validated and in parallel a statistics must be updated. Additionally, we have already stored that after the execution of an `Order` the same order is booked. We know that after the validity of the order has been checked by a second person, the competencies must be checked, too. Additionally, the process library contains process actions that are not related to orders, e.g. the transfer of money between different accounts or withdrawing cash from an account (compare Figure 4).

Now the event logs in the MiningXML-format (MXML for short, see [13]) which are summarized in Table I are processed. Three different event logs are read each of those including the name of an action that has been performed in one of the IT-systems, the instance number of the process and the person

TABLE I
PROCESS LOGS

Process action	**Log file**	**Instance**	**Orig.**
Store request in database	1	0815	Sonja
Validate credit balance	1	0815	Sonja
Send incoming order confirmation	1	4711	System
Store request in database	1	4711	Sonja
Store request in database	1	1610	Sonja
Update statistics	1	1610	System
Derivy possible improvements	1	1610	System
Check validity by second person	2	4711	Chris
Check competencies	2	4711	Ralph
Test routing	2	4711	John
Check validity by second person	2	2009	Chris
Check extended competencies	2	2009	Ralph
Assess risk	2	2009	Ralph
Test routing	2	2009	John
Check validity by second person	2	1610	Chris
Assess risk	2	1610	Ralph
Check extended competencies	2	1610	Ralph
Test routing	2	1610	John
Route order	3	2009	John
Execute stock order	3	2009	Marc
Execute stock order	3	1234	Marc
Book stock order	3	1234	Marc

who has caused the log entry. Based on these process logs the first basic tuples are build. These start with ({`Store request in database`}, {`Validate credit balance`}), ({`Send incoming order confirmation`},{`Store request in database`}), etc.

In the second step of the algorithm `Cystid` the process actions of the process library are added to the tuples. For the sequence in the process library `Check validity by second person` followed by `Check competencies` this means that tuples that start with `Check validity by second person` are searched and in each of these tuples `Check competencies` is added on the right side which results in two tuples ({`Check validity by second person`},{`Check extended competencies`,`Check competencies`}) and ({`Check validity by second person`},{`Assess risk`,`Check competencies`}).

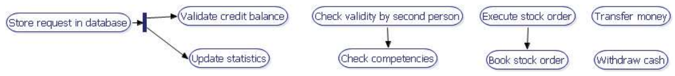

Fig. 4. Process Library before Mining

In the third step control structures are created based on the tuples. For the two tuples introduced before we now create a decision node between `Check competencies` and recognize a possible parallelization between `Check extended competencies` and `Assess risk`. The process library after creating the control structure includes the fragments shown in Figure 5.

In the next step the semantic input and output parameters for these fragments are requested from the user: e.g. the first fragment is annotated as follows: the input of `Send incoming order confirmation` is set as an incoming *`Order`* which includes a validation number of 0. *`Order`* is a composite parameter which consists in our ontology of an *`OrderType`* ({*`OrderType`*, {incoming, checked, routed, executed, booked}, {incoming}}), an *`OrderAmount`* (*`OrderAmount`*, positiveInteger, positiveInteger) and a number of validations (*`Validations`*, positiveInteger, positiveInteger). After `Validate credit balance` this validation number is set to one and the *`OrderType`* is still restricted to {incoming}. The semantic output of `Derive possible improvements` is a concept in the ontology called *`ListOfImprovements`*.

The second process log generates a new fragment which starts with checking the validity by a second person, then either checks competencies or checks extended competencies and assess the risks (depending on the *`OrderAmount`*) and at the end tests whether the order can be routed to the stock market. The semantic input of this fragment is an *`Order`* which has been validated only by one person and the output is an *`Order`* that has been validated twice and where the routing has been checked.

The third process log extends the already existing fragment (`Execute stock order` and `Book stock order`) by routing the order to the stock market, before it gets executed there. The semantic input of this fragment is annotated as a checked *`StockOrder`* (which is an equivalent to *`Order`* in the ontology) which should have at least two *`Validations`* and the output is a booked *`StockOrder`*. The process library after processing all three event logs can be seen in Figure 5.

Now the planning algorithm can be started. This evaluates all process actions and fragments, determines the dependencies between them and creates an Action Dependency Graph (ADG) in the first place (Step 5 in Figure 3). Afterwards, the planning starts with the information specified and an Action State Graph (ASG) is computed (Step 6). Thereby, only the actions and fragments are considered that lead to the specified goal state. The initial state for `SEMPA` is an incoming *`Order`* with no *`Validations`*. The goal state is set as an *`Order`* that is booked.

In the last step a syntactically correct UML activity diagram

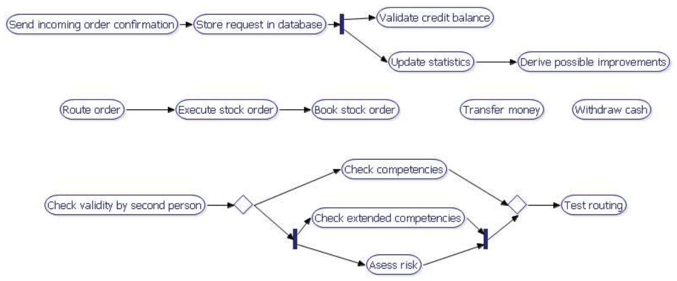

Fig. 5. Process Library after Mining

is created whereas missing control structures are added. For the `SEMPA` planning steps the fragments can be treated as single actions with input and output parameters. Only in the last step during the creation of the corresponding control nodes the already existing control nodes in the fragments need to be considered again. The resulting activity diagram can be seen in Figure 6.

Our approach has been prototypically implemented as an Eclipse plugin on top of plugins from the Eclipse project Java Workflow Tooling (JWT) [28]. JWT is a set of plugins for the modeling, validation, deployment and execution of processes. It has already been extended with plugins for the semantic annotation as well as with the planner `SEMPA` that has been described before. In addition to our case study we also tested *`Cystid`* with sample log files available at http://prom.win.tue.nl/tools/prom/.

In our case study not all systems in the company supported process logs, so only some process actions and fragments were added to the process library automatically, whereas others had to be added manually. Our approach is currently limited by the manual process of semantic annotating the process fragments. This requires that an ontology already exists for the domain of discourse which is still not the case for many business domains. The process of gathering a *shared conceptualization* requires a lot of time and only if the benefits are clearly visible for all participants they are motivated for creating an ontology and adding semantic annotations to process actions.

VI. Related Work

At the moment different approaches for process analysis exist: according to [29] they can be categorized in four different areas: action analysis, process mapping, coordination analysis and social grammar analysis. We are only concerned about action analysis which itself can be structured in three different views: the process view, resource view and object view [30]. In this work we focus on the process view.

The idea of combining Process Mining techniques and semantic annotation of process actions with ontology concepts is quite a new one. Although taxonomies have already been used to improve process execution (see e.g. [31], [32]), only recently ontologies were directly connected with Process Mining tools: In [26] the authors describe the idea of Semantic Business Process Mining and that the usage of ontologies

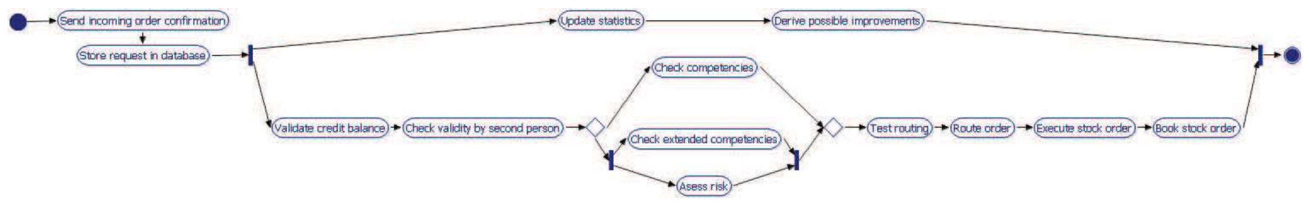
Send incoming order confirmation
Store request in database
Update statistics
Derive possible improvements
Check competencies
Validate credit balance
Check validity by second person
Check extended competencies
Asess risk
Test routing
Route order
Execute stock order
Book stock order

Fig. 6. Resulting UML Activity Diagram

might help to discover not only flat (a single large model without any hierarchy structure), but hierarchical process models. One possible ontology, the core ontology for business process analysis, is described in more detail in [33]. This idea is extended in [34] where core building blocks for semantic Process Mining such as an own format called SA-MXML (Semantically-annotated Mining XML) and an own tool called Semantic LTL checker are introduced. SA-MXML builds on the idea of SAWSDL [35] which uses a new property called *modelReference* in order to reference events in the process log to concepts in an ontology. Whereas this is an interesting approach, it does not aim to achieve complete process models in the end, especially when events in the log files are missing. Our approach combines classical process mining with process planning tools in order to retrieve a complete process model in the end.

VII. Conclusion and Prospects

In this paper we have described how Process Mining and SBPM can be combined. We introduced our algorithm `Cystid` which allows to use the fragments that are the outcome of Process Mining for semantic-based planning in order to get a complete process model. Thereby, we combine a goal-driven and a state-driven approach for process modeling.

Using Process Mining alone can sometimes lead to complete process models, too. But as described before, especially when *noise* or *hidden tasks* occur, Process Mining still has to cope with problems since it can not reconstruct the whole process model again. Our approach allows to annotate the reconstructed process fragments with semantic concepts in order to allow the planning of process models afterwards, whereby already existing process actions in the process library are considered, too. Hence, our approach can support Process Mining on the one hand as well as Semantic Business Process Modeling on the other.

Of course the inclusion of semantic information (references to existing and open available ontologies) already in the process log would be very helpful. As outlined before, a semantic extension of the MXML-format has already been presented in other works called SA-MXML. This format includes model references to concepts in an ontology. As a next step we consider to integrate this format into our approach in order to reduce the manual work for semantically annotating the process actions and fragments once more.

References

[1] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel, "Semantic business process management: A vision towards using semantic web services for business process management," in *Proceedings of IEEE ICEBE, Beijing, China*, October 2005, pp. 535–540.

[2] M. Henneberger, B. Heinrich, F. Lautenbacher, and B. Bauer, "Semantic-based planning of process models," in *Proceedings of Multikonferenz Wirtschaftsinformatik (MKWI)*, February 2008.

[3] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5 (2), pp. 199–220, 1993.

[4] A. Tiwari, C. Turner, and B. Majeed, "A review of business process mining: state-of-the-art and future trends," *Business Process Management Journal*, vol. 14, no. 1, pp. 5–22, 2008.

[5] M. Weske, *Business Process Management - Concepts, Languages, Architectures*. Berlin Heidelberg: Springer Verlag, 2007.

[6] S. Goedertier and J. Vanthienen, "Declarative process modeling with business vocabulary and business rules," in *Proceedings of OTM 2007, Ws, Part I*, ser. LNCS, R. Meersman, Z. Tari, and P. Herrero, Eds., vol. 4805. Springer-Verlag, 2007, pp. 603–612.

[7] WfMC, "Workflow reference model," 1995. [Online]. Available: http://www.wfmc.org/standards/docs/tc003v11.pdf

[8] J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through SESE decomposition," in *Proceedings of ICSOC*, ser. LNCS, B. Krämer, K.-J. Lin, and P. Narasimhan, Eds., vol. 4749. Springer-Verlag Berlin Heidelberg, 2007, pp. 43–55.

[9] OMG, "Unified modeling language (UML) superstructure, version 2.1.2," OMG Specification, November 2007. [Online]. Available: http://www.omg.org/docs/formal/05-07-04.pdf

[10] A. Weijters and W. van der Aalst, "Process mining: Discovering workflow models from event-based data," in *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, B. Kröse, M. De Rijke, G. Schreiber, and M. v. Someren, Eds., Maastricht, 2001, pp. 283–290.

[11] J. Cook and A. Wolf, "Discovering models of software processes from event-based data," *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 3, pp. 215–249, 1998.

[12] R. Agrawal, D. Gunopulos, and F. Leymann, *Mining process models from workflow logs*. Springer-Verlag, 1998.

[13] B. van Dongen, A. de Medeiros, H. Verbeek, A. Weijters, and W. van der Aalst, "The ProM framework: A new era in process mining tool support," in *ICATPN*, ser. LNCS, G. Ciardo and P. Darondeau, Eds., vol. 3356. Springer-Verlag Berlin Heidelberg, 2005, pp. 444–454.

[14] W. van der Aalst and A. Weijters, "Process mining: a research agenda," *Computers in Industry*, vol. 53, pp. 231 – 244, 2004.

[15] A. de Medeiros, B. van Dongen, W. van der Aalst, and A. Weijters, "Process mining: Extending the alpha-algorithm to mine short loops," Eindhoven University of Technology, Eindhoven, BETA Working Paper Series WP 113, 2007.

[16] O. Thomas and M. Fellmann, "Semantic business process management: Ontology-based process modeling using event-driven process chains," *International Journal of Interoperability in Business Information Systems*, vol. 2 (1), 2007.

[17] M. Born, F. Dörr, and I. Weber, "User-friendly semantic annotation in business process modeling," in *WISE 2007 Workshops*, ser. LNCS, M. Weske, M.-S. Hacid, and C. Godart, Eds., no. 4832. Springer-Verlag, 2007, pp. 260–271.

[18] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "OWL2: The next step for OWL," *Journal on Web Semantics*, vol. 6, no. 4, pp. 309 – 322, November 2008. [Online]. Available: http://www.w3.org/TR/owl2-syntax/,asof2008-04-16

[19] S. Simmonds, A. Solberg, R. Reddy, R. France, and R. Ghosh, "An aspect oriented model driven framework," in *Proceedings of the Ninth IEEE EDOC, Enschede, Netherlands, September*, 2005.

[20] I. Groher and M. Voelter, "Xweave: Models and aspects in concert," in *AOM Workshop, Marc 12, 2007, Vancouver, British Columbia, Canada*, 2007.

[21] P. Biron and A. Malhotra, "Xml schema part 2: Datatypes second edition," 2004. [Online]. Available: http://www.w3.org/TR/xmlschema-2/,asof2007-10-04

[22] S. Degwekar, H. Lam, and S. Y. Su, "Constraint-based brokering (CBB) for publishing and discovery of web services," *International Journal on Electronic Commerce Research*, vol. 7, no. 1, pp. 45 – 67, 2007.

[23] B. Heinrich, M. Henneberger, A. Krammer, M. Bewernik, and F. Lautenbacher, "SEMPA – Ein Ansatz des Semantischen Prozessmanagements zur Planung von Prozessmodellen," *Journal Wirtschaftsinformatik*, vol. November/December, 2008.

[24] W. van der Aalst, "Trends in business process analysis - from verification to process mining," in *Proceedings of International Conference on Enterprise Information Systems (ICEIS)*, Funchal, Portugal, June, 12-16 2007, pp. IS–13 – IS–22.

[25] J. Mendling, H. Verbeek, B. van Dongen, W. van der Aalst, and G. Neumann, "Detection and prediction of errors in EPCs of the SAP reference model," *Data & Knowledge Engineering*, vol. 64, pp. 312 – 329, 2008.

[26] A. Alves de Medeiros, C. Pedrinaci, W. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral, "An outlook on semantic business process mining and monitoring," in *OTM 2007 Ws, Part II*, ser. LNCS, R. Meersman, Z. Tari, and P. Herrero, Eds., vol. 4806. Springer-Verlag Berlin Heidelberg, 2007, pp. 1244–1255.

[27] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros, "Workflow patterns," *Distributed and Parallel Databases 14*, vol. 3, pp. 5 – 51, 2003.

[28] F. Lautenbacher, "Execute your processes with Eclipse JWT," in *Proceedings of JAX / SOACon / Eclipse Forum Europe, Mainz, Germany*, April, 20 - 24 2009.

[29] S. Biazzo, "Approaches to business process analysis: a review," *Business Process Management Journal*, vol. 6, no. 2, pp. 99 – 112, 2000.

[30] M. zur Muehlen, *Workflow-based Process Controlling. Foundation, Design and Implementation of Workflow-driven Process Information Systems*, ser. Advances in Information Systems and Management Science. Logos, Berlin, 2004, vol. 6.

[31] F. Casati and M.-C. Shan, "Semantic analysis of business process executions," in *Proceedings of the 8th International Conference on Extending Database Technology (EDBT)*, London, UK, 2002, pp. 287–296.

[32] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan, "Business process intelligence," *Computers in Industry*, vol. 53, no. 3, pp. 321–343, 2004.

[33] C. Pedrinaci, J. Domingue, and A. K. Alves de Medeiros, "A core ontology for business process analysis," in *Proceedings of ESWC 2008, Tenerife, Spain*, ser. LNCS, vol. 5021, 2008, pp. 49–64.

[34] A. K. Alves de Medeiros, W. M. van der Aalst, and C. Pedrinaci, "Semantic process mining tools: Core building blocks," in *Proceedings of the 16th European Conference on Information Systems (ECIS)*, W. Golden, T. Acton, and K. Conboy, Eds., 2008.

[35] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic annotations for WSDL and XML schema," *IEEE Internet Computing*, vol. November/December, pp. 60–67, 2007.