Published version at: http://dx.doi.org/10.1016/j.eswa.2017.03.063

Multi-level abstraction for trace comparison and semantic process mining

Stefania Montani[a], Giorgio Leonardi[a,*], Manuel Striani[b], Silvana Quaglini[c], Anna Cavallini[d]

[a] *DISIT, Computer Science Institute, Università del Piemonte Orientale, Viale Michel 11, Alessandria, Italy*

[b] *Department of Computer Science, Università di Torino, Corso Svizzera 105, Torino, Italy*

[c] *Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Via Ferrata 1, Pavia, Italy*

[d] *I.R.C.C.S. Fondazione "C. Mondino", Via Mondino 2, Pavia, Italy - on behalf of the Stroke Unit Network (SUN) collaborating centers*

Abstract

Many information systems record executed process instances in the *event log*, a very rich source of information for several process management tasks, like process mining and trace comparison. In this paper, we present a framework, able to convert actions in the event log into higher level concepts, at different levels of abstraction, on the basis of domain knowledge. Abstracted traces are then provided as an input to trace comparison and semantic process mining. Our abstraction mechanism manages non trivial situations, such as interleaving actions or delays between two actions that abstract to the same concept. Trace comparison resorts to a similarity metric able to take into account abstraction phase penalties, and to deal with quantitative and qualitative temporal constraints. As for process mining, we rely on classical algorithms embedded in the framework ProM, made "semantic" by the capability of abstracting the actions on the basis of their conceptual meaning. The approach has been tested in stroke care, where we adopted abstraction

*Corresponding author. Tel. +39 0131 360158

Email addresses: stefania.montani@uniupo.it (Stefania Montani), giorgio.leonardi@uniupo.it (Giorgio Leonardi), striani@di.unito.it (Manuel Striani), silvana.quaglini@unipv.it (Silvana Quaglini), anna.cavallini@mondino.it (Anna Cavallini)

and trace comparison to cluster event logs of different stroke units, to highlight (in)correct behavior, abstracting from details. We also provide semantic process discovery results, showing how the abstraction mechanism allows to obtain more readable stroke process models.

Keywords: Abstraction, Trace comparison, Semantic process mining, Stroke management

---

1. Introduction

Many commercial information systems and enterprise resource planning tools, routinely adopted by organizations worldwide, record information about the executed business process instances in the form of an *event log* (IEEE Taskforce on Process Mining: Process Mining Manifesto). The event log stores the sequences (*traces* (der Aalst, 2011) henceforth) of actions that have been executed at the organization, typically together with some key parameters, such as execution times.

Event logs constitute a very rich source of information for several business process management tasks. Indeed, the experiential knowledge embedded in traces is directly resorted to, e.g., in *operational support* (der Aalst, 2011) and in *agile workflow* tools (Weber & Wild, 2005), which can take advantage of **trace comparison** and retrieval, to make predictions about a running process instance completion, or to provide instance adaptation support, in response to expected situations as well as unanticipated exceptions in the operating environment. Moreover, event logs are the input to **process mining** (IEEE Taskforce on Process Mining: Process Mining Manifesto; der Aalst et al., 2003; van der Aalst et al., 2004) algorithms, a family of a-posteriori analysis techniques able to extract non-trivial knowledge from these historic data; within process mining, process model discovery algorithms, in particular, take as input the event log and build a process model, focusing on its control flow constructs.

All of these activities, however, provide a purely syntactical analysis, where actions in the event log are compared and processed only referring to their names. Action names are strings without any semantics, so that identical actions, labeled by synonyms, will be considered as different, or actions that are special cases of other actions will be processed as unrelated.

Upgrading trace comparison and process mining to the conceptual layer can enhance existing algorithms towards more advanced and reliable ap-

proaches. Indeed, the capability of relating semantic structures such as taxonomies or ontologies to actions in the log can enable both trace comparison and process mining techniques to work at **different levels of abstraction** (i.e., at the level of instances and/or concepts) and, therefore, to mask irrelevant details, to promote reuse, and, in general, to make trace and/or process model analysis much more flexible, and closer to the real user needs. As a matter of fact, **semantic process mining**, defined as the integration of semantic processing capabilities into classical process mining techniques, has been proposed in the literature since the first decade of this century (see, e.g., (de Medeiros et al., 2008; de Medeiros & van der Aalst, 2009), and Section 4). However, while more work has been done in the field of semantic conformance checking (de Medeiros et al., 2008; Grando et al., 2011), to the best of our knowledge semantic process discovery needs to be further investigated.

In this paper, we present a **semantic-based, multi-level abstraction mechanism**, able to operate on event log traces. In our approach, actions in the log are mapped to instances of ground concepts (leaves) in a taxonomy, so that they can be converted into higher-level concepts by navigating the hierarchy, up to the desired level, on the basis of the user needs. The abstraction mechanism is then provided as an input to further analysis mechanisms, namely **trace comparison** and **process mining**.

The **abstraction mechanism** has been designed to properly tackle non-trivial issues that could emerge. Specifically:

- two actions having the same ancestor in the taxonomy (at the chosen abstraction level) may be separated in the trace by a *delay* (i.e., a time interval where no action takes place), or by actions that descend from a different ancestor (*interleaved actions* henceforth). Our approach allows to deal with these situations, by creating a single *macro-action*, i.e., an abstract action that covers the whole time span of the two actions at hand, and is labeled as the common ancestor; the macro-action is however built only if the total delay length, or the total number/length of interleaved actions, do not overcome proper admissibility thresholds set by the user. The delays and interleaved actions are quantified and recorded, for possible use in further analyses. In particular, we present a **similarity metric** where this information is accounted for as a penalty, and affects the similarity value in abstract trace comparison;

- abstraction may generate different types of temporal constraints be-

tween pairs of macro-actions; specifically, given the possible presence of interleaved actions, we can obtain an abstracted trace with two (or more) overlapping or concurrent macro-actions. Our approach allows to represent (and exploit) this information, by properly maintaining both quantitative and qualitative temporal constraints in abstracted traces. Once again, this temporal information can be exploited in further analyses. In particular, the **similarity metric** we adopt in trace comparison can deal with all types of temporal constraints.

The most significant and original methodological contributions of our work thus consist in:

1. having defined a proper ***mechanism for abstracting event log traces***, able to manage non trivial situations (originating from the treatment of interleaving actions or delays between two actions sharing the same ancestor);
2. having provided ***a trace comparison facility***, which resorts to a ***similarity metric*** (extending the metric we presented in Montani & Leonardi (2014)), able to take into account also the information recorded during the abstraction phase.

On the other hand, as for process mining, we rely on classical algorithms embedded in the open source framework ProM (van Dongen et al., 2005)[1].

In addition to these methodological contributions, in the paper we also describe our experimental work in the field of stroke care, in which we adopted multi-level abstraction and trace comparison to cluster event logs of different stroke units, in order to highlight correct and incorrect behaviors, abstracting from details, such as local resource constraints or local protocols, that are irrelevant to verify the medical appropriateness of a macro-action. We also provide semantic process discovery results, showing how the abstraction mechanism allows to obtain simpler and more understandable stroke process models.

The paper is organized as follows. Section 2 presents methodological and technical details of the framework. Section 3 describes experimental results. Section 4 addresses comparisons with related work. Finally, Section 5 is devoted to conclusions and future research directions.

[1]It is worth noting that the abstraction mechanism could, in principle, be given as an input to different analysis techniques as well, besides the ones described in this paper.

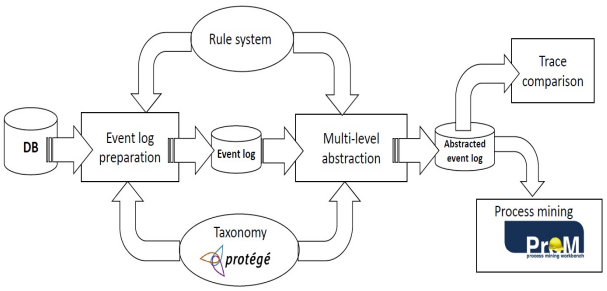

Figure 1: Framework architecture and data flow

2. Methods

This section describes methodological and technical details of our approach.

The architecture and the data flow of the framework we have developed are shown in Figure 1, where rectangles represent computational modules, while ovals and cylinders represent domain knowledge sources and databases, respectively.

The first step to be executed is *event log preparation*, that takes as input the available database (recording executed actions and additional data), and exploits domain knowledge sources (a taxonomy and a rule base); the event log then undergoes *multi-level abstraction*, which resorts to the domain knowledge sources as well. The abstracted event log can be given as an input to *trace comparison*, or to *process mining*, currently realized by resorting to ProM (van Dongen et al., 2005).

The terminology we use, and the details about domain knowledge sources and computational modules, are described in the following subsections.

2.1. Terminology

Action or **ground action**: an action recorded in an event log trace (corresponding to a leaf concept in the taxonomy described in Section 2.2).
Delay: time interval between two ground actions logged in a trace, where no other action takes place.
Interleaved action: a ground action that descends from a different ancestor in the taxonomy of actions described in Section 2.2, with respect to the two ground actions that are currently being considered for abstraction. In

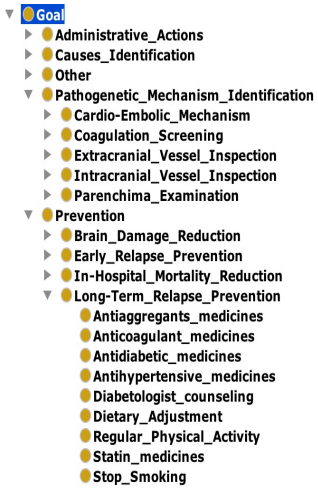

Figure 2: An excerpt from the stroke domain taxonomy

the trace, the interleaved action is placed between the two actions at hand.
Macro-action: partial output of the abstraction process; a macro-action is an abstracted action that covers the whole time span of multiple ground actions, and is labeled as their common ancestor in the taxonomy, at the specified abstraction level. As a special case, the macro-action can abstract a single ground action as its ancestor.
Singleton: a ground action that can not be abstracted to any macro-action, because some precondition for the abstraction process is not verified. Preconditions are formalized in the rule base described in Section 2.2.
Abstracted trace: global output of the abstraction process; an abstracted trace is the transformation of an input trace into a new trace containing only macro-actions or singletons.

2.2. Domain knowledge sources

In our framework, domain knowledge is provided by means of a taxonomy and of a rule base. In the paper, all examples will refer to the domain of stroke management.

An excerpt of our stroke management taxonomy is reported in Figure 2.

The taxonomy, which has been formalized by using the Protègè editor, has been organized by goals. Indeed, a set of classes, representing the main goals in stroke management, have been identified, namely: "Prevention", "Pathogenetic Mechanism Identification", "Causes Identification", "Administrative

Actions" and "Other". Some of these main goals, in a parent-child relation, are further specialized into subclasses, according to more specific goals (e.g., "Prevention" specializes into "Early Relapse Prevention", "Long Term Relapse Prevention", "Brain Damage Reduction" and "In-Hospital Mortality Reduction"), down to the ground actions, that will implement the goal itself (e.g., "Long Term Relapse Prevention", aiming at preventing another stroke in the long run, specializes into several ground actions, including "Anticoagulant Medicines" and "Diabetologist Counseling" - see Figure 2). Overall, our taxonomy is composed by 136 classes, organized in a hierarchy of four levels.

The rule base is represented as an XML file where actions are coupled with preconditions. Precondition verification enables the abstraction process; preconditions can be stated in AND or in OR, and might provide temporal constraints to be respected between the occurrence of the precondition and the starting time of the action.

An example rule is provided below. The rule refers to the action "Diabetologist Counseling", and has "Diabetes" OR "Hyperglycemia" as preconditions. As already observed (see Figure 2), the action "Diabetologist Counseling" is a descendant of the more abstract "Long Term Relapse Prevention" goal, stating that the goal of diabetologist counseling is the one of preventing stroke recurrence in the long run. However, a diabetologist counseling action correctly abstracts as long term relapse prevention only if the patient at hand is diabetic, or experienced hyperglycemia. If this is not the case, diabetologist counseling is not correctly identifiable as an action for preventing long term relapse. Indeed, in this situation this action was probably inappropriate, wasting time and human resources. The rule below exactly states this semantics. Rules thus allow to highlight incorrect behaviors in the process, that will be identified as *singletons* by the abstraction algorithms. Abstraction details are provided in Section 2.4.

```
<Rule name="rule_Diabetologist_counseling">
  <preconditionsOr>
      <preconditionOr>Diabetes</preconditionOr>
      <preconditionOr>Hyperglycemia</preconditionOr>
  </preconditionsOr>
</Rule>
```

2.3. Event log preparation

As illustrated in Figure 1, the *event log preparation* module takes as input the database (containing action execution information, such as starting and ending times, and additional data, like, e.g., patient's demographics and clinical data in the medical domain); it also takes as input the domain knowledge sources, i.e., the taxonomy and the rule base.

In this phase, the starting/ending times of actions are used to calculate action ordering within every trace.

The log preparation module generates an event log where traces are represented in an eXtensible Event Stream (XES) (Verbeek et al., 2011) file. The XES format is an extension of the MXML (van Dongen & van der Aalst, 2005) format where elements have an optional extra attribute called $modelReference$. This attribute allows to link an action to a concept in an ontology: in our case, to a leaf in the taxonomy. Proper action attributes also allow to record precondition values, to be verified by rules in the abstraction phase (see Section 2.4).

2.4. Multi-level abstraction of the event log

Our multi-level abstraction procedure operates as described in Algorithm 1 below. The function *abs_greedy* takes as input an event log *trace*, the domain taxonomy *taxo*, the *rule* base, and the *level* in the taxonomy chosen for the abstraction (e.g., $level = 1$ corresponds to the choice of abstracting the actions up to the sons of the taxonomy root). It also takes as input three thresholds ($delay\_th$, n_inter_th and $inter\_th$). These threshold values have to be set by the domain expert in order to limit the total admissible delay time within a macro-action, the total number of interleaved actions, and the total duration of interleaved actions, respectively. In fact, it would be hard to justify that two ground actions share the same goal (and can thus be abstracted to the same macro-action), if they are separated by very long delays, or if they are interleaved by many/long different ground actions, meant to fulfill different goals.

The function outputs an abstracted trace.

For every action i in *trace*, an iteration is executed (lines 3-32). First, the preconditions of i, set in the log preparation phase, are considered. If the set of preconditions of i is empty (because no rule involving i was provided by domain knowledge), or if the preconditions of i are verified (verification is performed resorting to the *rule* system, whose detailed description is outside the scope of this paper; rules could state, for instance, if preconditions have to

be considered in AND or in OR), then a macro-action m_i, initially containing just i, and sharing its starting and ending times, is created. m_i is labeled referring to the ancestor of i at the abstraction *level* provided as an input. Accumulators for this macro-action (total-delay, num-inter and total-inter, commented below) are initialized to 0 (lines 4-10). Then, a nested cycle is executed (lines 11-25): it considers every element j following i in the trace, where a trace element can be an action, or a delay between a pair of consecutive actions. Different scenarios can occur:

- if j is a delay, $total - delay$ is updated by summing the length of j (lines 12-14).
- if j is an action, the set of preconditions of j is empty or its preconditions are verified, and j shares the same ancestor of i at the input abstraction *level*, then j is incorporated into the macro-action m_i. This operation is always performed, provided that $total - delay$, $number - inter$ and $total - inter$ do not exceed the threshold passed as an input (lines 15-19). j is then removed from the actions in *trace* that could start a new macro-action, since it has already been incorporated into an existing one (line 18). This kind of situation is described in Figure 3 (a).
- if j is an action, but does not share the same ancestor of i, or violates some preconditions, then it is treated as an interleaving action. In this case, $num - inter$ is increased by 1, and $total - inter$ is updated by summing the length of j (lines 20-23). This situation, in the end, may generate different types of temporal constraints between macro-actions, as the ones described in Figure 3 (b) (Allen's *during* (Allen, 1984)) and Figure 3 (c) (Allen's *overlaps* (Allen, 1984)).

On the other hand, if some of the preconditions of i are not verified, i cannot be abstracted referring to its ancestor in the taxonomy. In this case a singleton is created, i.e., a dummy macro-action m_i, sharing the starting and ending times of i, that will not aggregate with any other action (lines 26-30).

Finally, the macro-action m_i is appended to *abs_trace*, that, in the end, will contain the list of all the macro-actions and singletons that have been created by the procedure (line 31).

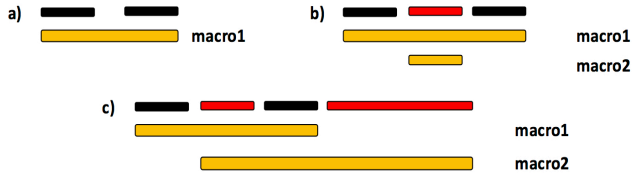

Figure 3: Different trace abstraction situations: (a) two actions are abstracted to a single macro-action *macro*1, with a delay in between; (b) two actions are abstracted to a macro-action *macro*1, with an interleaved action in between, resulting in a different macro-action *macro*2 during *macro*1; (c) two actions are abstracted to a macro-action *macro*1, with an interleaved action in between, which is later aggregated to a fourth action, resulting in a macro-action *macro*2 overlapping *macro*1.

Complexity. The cost of abstracting a trace is $O(actions * elements)$, where *actions* is the number of actions in the input trace, and *elements* is the number of elements (i.e., actions + delay intervals) in the input trace.

2.5. Trace comparison

In our approach, every trace (abstracted trace) is a sequence of actions (macro-actions, respectively), each one stored with its execution starting and ending times. Therefore, an action is basically a symbol (plus possible execution parameters, in particular the temporal information). Starting and ending times allow to get information about action durations, as well as qualitative (e.g., Allen's *before*, *overlaps*, *equals* etc. (Allen, 1984)) and quantitative temporal constraints (e.g., delay length, overlap length (Lanz et al., 2010)) between pairs of consecutive actions/macro-actions.

In order to calculate the distance between two abstracted traces, we have extended a metric for ground trace comparison we published in Information Systems in 2014 (Montani & Leonardi, 2014). The main features of this metric are summarized below. The extensions needed to deal with abstracted traces are also discussed in this section.

In the metric in Montani & Leonardi (2014), we first take into account action types, by calculating a modified edit distance which we have called **Trace Edit Distance** (Montani & Leonardi, 2014). As the classical edit distance (Levenshtein, 1966), Trace Edit Distance tests all possible combinations of editing operations that could transform one trace into the other one. However, the cost of a *substitution* is not always set to 1. Indeed, as already observed, we have organized actions in a taxonomy: we can therefore adopt a more **semantic** approach, and apply Palmer's distance (Palmer &

ALGORITHM 1: Multi-level abstraction algorithm

```
1  abs_trace = abs_greedy(trace, taxo, rule, level, delay_th,
   n_inter_th, inter_th);
2  abs_trace = ∅;
3  for every i ∈ actions in trace do
4      if ((i.precond = ∅ ∨ i.precond = verified) ∧ (i.startFlag = yes)) then
5          create : m_i as ancestor(i, level);
6          m_i.start = i.start;
7          m_i.end = i.end;
8          total_delay = 0;
9          num_inter = 0;
10         total_inter = 0;
11         for (every j ∈ tokens in trace) do
12             if (j is a delay) then
13                 total_delay = total_delay + j.length;
14             else
15                 if (j.precond = ∅ ∨ j.precond = verified) ∧
                   (ancestor(j, level)=ancestor(i, level)) then
16                     if (total_delay < delay_th ∧ num_inter <
                       n_inter_th ∧ total_inter < inter_th) then
17                         m_i.end = max(m_i.end, j.end);
18                         j.startFlag = no;
19                     end
20                 else
21                     num_inter = num_inter + 1;
22                     total_inter = total_inter + j.length;
23                 end
24             end
25         end
26     else if (precond = ¬verified) then
27         create : m_i as singleton;
28         m_i.start = i.start;
29         m_i.end = i.end;
30     end
31     append m_i to abs_trace;
32 end
33 return abs_trace;
```

Wu, 1995), to impose that the closer two actions are in the taxonomy, the less penalty we introduce for substitution. Trace Edit Distance then takes the combination of editing operations associated to the minimal cost. Such a choice corresponds to a specific alignment of the two traces (*optimal alignment* henceforth), in which each action in one trace has been matched to an action in the other trace–or to a gap.

Given the optimal alignment, we can then take into account temporal information. In particular, we compare the durations of aligned actions by means of a metric we called **Interval Distance** (Montani & Leonardi, 2014).

Moreover, we take into account the temporal constraints between two pairs of subsequent aligned actions on the traces being compared (e.g., actions A and B in trace P; the aligned actions A' and B' in trace Q). We quantify the distance between their qualitative constraints (e.g., A and B overlap in trace P; A' meets B' in trace Q), by resorting to a metric known as **Neighbors-graph Distance** (Montani & Leonardi, 2014). If Neighbors-graph Distance is 0, because the two pairs of actions share the same qualitative constraint (e.g., A and B overlap in trace P; A' and B' also overlap in trace Q), we compare quantitative constraints by properly applying Interval Distance again (e.g., by calculating Interval Distance between the two overlap lengths).

In the metric in Montani & Leonardi (2014), these three contributions (i.e., Trace Edit Distance, Interval Distance between durations, Neighbors-graph Distance or Interval Distance between pairs of actions) are finally combined as a linear combination with non-negative weights.

When working on macro-actions, however, the metric in Montani & Leonardi (2014) needs to be extended, by considering, given the optimal macro-actions alignment, two additional contributions:

- a penalty due to the different length of the delays incorporated into the two aligned macro-actions;
- a penalty due to the different number, length and type of interleaved actions in the two aligned macro-actions being compared.

Delay penalty is defined as follows:

Definition 1: Delay Penalty.

Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $delay_A = \sum_{i=1}^{k} length(i)$ be the sum of the lengths of all the k delays that have been incorporated into A in the abstraction phase,

calculated by Algorithm 1 (and let $delay_B$ be analogously defined). Let $maxdelay$ be the maximum, over all the abstracted traces, of the sum of the lengths of the delays incorporated in an abstracted trace. The Delay Penalty $delay_p(A, B)$ between A and B is defined as:

$$delay_p(A, B) = \frac{|delay_A - delay_B|}{maxdelay}$$

As for interleaved actions penalty, we operate analogously to delay penalty, by summing up the lengths of all interleaved actions that have been incorporated within a single macro-action in the abstraction phase.

Definition 2: Interleaving Length Penalty.
Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $inter_A = \sum_{i=1}^{k} length(i)$ be the sum of the lengths of all the k interleaved actions that have been incorporated into A in the abstraction phase, calculated by Algorithm 1 (and let $inter_B$ be analogously defined). Let $maxinter$ be the maximum, over all the abstracted traces, of the sum of the lengths of the interleaved actions incorporated in an abstracted trace. The Interleaving Length Penalty $interL_p(A, B)$ between A and B is defined as:

$$interL_p(A, B) = \frac{|inter_A - inter_B|}{maxinter}$$

The extended metric working on abstracted traces includes in the linear combination these two penalties as well.

It is worth noting that our metric, given its capability to manage both quantitative and qualitative temporal constraints, enables to properly deal with temporal information at all abstraction levels.

By allowing the treatment of abstraction penalties and the management of temporal information, the extended metric is therefore able to address all the issues we cited in the Introduction.

2.6. Process mining

In our approach, we are resorting to the well-known process mining tool ProM, extensively described in van Dongen et al. (2005). ProM (and specifically its newest version ProM 6) is a platform-independent open source

framework that supports a wide variety of process mining and data mining techniques, and can be extended by adding new functionalities in the form of plug-ins.

For the experimental work described in this paper, we have exploited ProM's Heuristic Miner (Weijters et al., 2006). Heuristic Miner (Weijters et al., 2006) is a plug-in for process discovery, able to mine process models from event logs. Heuristic Miner receives as input the log, and considers the order of the actions within every single trace. It can mine the presence of short-distance and long-distance dependencies (i.e., direct or indirect sequence of actions), and information about parallelism, with a certain degree of reliability. The output of the mining process is provided as a graph, known as the "dependency graph", where nodes represent actions, and edges represent control flow information. The output can be converted into other formalisms as well.

Currently, we have chosen to rely on Heuristics Miner, because it is known to be tolerant to noise, a problem that may affect medical event logs (e.g., sometimes the logging may be incomplete). Anyway, testing of other mining algorithms available in ProM 6 is foreseen in our future work. Moreover, the interface of our framework to ProM will allow us to test additional analysis plug-ins in the future.

3. Results

In this section, we describe two experimental works we have conducted, in the application domain of stroke care. In the first one (see Section 3.1), we have studied the impact of multi-level abstraction on trace comparison; in particular, we have designed a set of clustering experiments, to verify whether it is possible to highlight correct behaviors and anomalies with respect to the latest clinical practice guidelines for stroke management, abstracting from details (such as, e.g., local resource constraints or local medical practice), that are irrelevant to the verification of medical appropriateness of a macro-action.

In the second work (see Section 3.2), we wished to verify whether our support to semantic process mining, and specifically the capability of abstracting the traces on the basis of their semantic goal, allowed to obtain clearer medical process models, where unnecessary details are hidden, but key behaviors are clear.

In the experiments, thresholds to be passed as input to the abstraction algorithm (see Algorithm 1) were common to all traces in the log, and set as follows: $delay\_th = 300$ minutes, $n\_inter\_th = 3$, $inter\_th = 300$ minutes. This choice was set by our medical co-author, on the basis of medical knowledge. Interestingly, we also made tests with different thresholds (making changes of up to 10%), but results (not reported due to lack of space) did not differ significantly.

The metric we adopted for trace comparison is the one we described in Section 2.5, where the linear combination weights were all equal and their sum was 1.

Experiments were run on a machine equipped with an Intel(R) Xeon(R) CPU E5-2640v2, CPU @ 2GHz, 4GB RAM.

Results are provided in the following.

3.1. Trace comparison

As a first experimental work, we have analyzed the impact of our abstraction mechanism on trace comparison, and more precisely on the quality of trace clustering.

The available event log was composed of more than 15000 traces, collected at the 40 Stroke Unit Network (SUN) collaborating centers of the Lombardia region, Italy. Our medical co-author belongs to one of the SUN stroke units. Thus, she has a very deep insight into the registry data. Traces were composed of 13 actions on average. The 40 Stroke Units (SUs) are not all equipped with the same human and instrumental resources: in particular, according to resource availability, they can be divided into 3 classes. Class-3 SUs are top class centers, able to deal with particularly complex stroke cases; class-1 SUs, on the contrary, are the more generalist centers, where only standard cases can be managed. Class 3 counts 9 SUs, class 2 includes 25 SUs, and class 1 is composed by 6 SUs.

In our study, we first considered the traces of every single SU separately, and compared clustering results on ground traces with respect to those on abstracted traces. We then repeated the experiment by keeping together the traces of the SUs classified as belonging to the same class. In these additional experiments, once again, we compared clustering results on ground traces with respect to those on abstracted traces.

For the sake of brevity, only two experimental results will be shown in this section.

Specifically, we resorted to a hierarchical clustering technique, known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) (Sokal & Michener, 1958). UPGMA is typically applied in bioinformatics, where sequences of symbols (similar to our traces) have to be compared. The algorithm operates in a bottom-up fashion. At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters A and B is taken to be the average of all distances between pairs of objects "x" in A and "y" in B, that is, the mean distance between elements of each cluster. After the creation of a new cluster, UPGMA properly updates a pairwise distance matrix it maintains. UPGMA also allows to build the phylogenetic tree (the hierarchy) of the obtained clusters.

In all of these experiments, the hypothesis we wished to test was the following: "the application of the abstraction mechanism allows to obtain more *homogeneous* and compact clusters (i.e., able to aggregate closer examples); however, outliers are still clearly identifiable, and isolated in the cluster hierarchy". Homogeneity is a widely used measure of the quality of the output of a clustering method (see e.g., (Yip et al., 2003; Sharan & Shamir, 2000; Duda et al., 2001; Francis et al., 2004)). A classical definition of cluster homogeneity is the following (Yip et al., 2003):

$$H(C) = \frac{\sum_{x,y \in C}(1 - dist(x,y))}{\binom{|C|}{2}}$$

where $|C|$ is the number of elements in cluster C, and $1 - dist(x,y)$ is the similarity between any two elements x and y in C. Note that, in the case of one-trace clusters, homogeneity is set to 1 (see e.g., (Francis et al., 2004)). The higher the homogeneity value, the better the quality of clustering results. The average of the homogeneity H of the individual clusters can be calculated on (some of) the clusters obtained through the method at hand, in order to assess clustering quality.

We computed the average of cluster homogeneity values level by level in the hierarchies.

First, we worked on single SUs. As an example, we report on the results of applying UPGMA to the 240 traces of SUcl2, a class-2 SU. The obtained cluster hierarchy height was 19 when working on ground traces, and 21 when working on abstracted ones. Figure 4 shows a comparison of the average homogeneity values, computed by level in the cluster hierarchies, on ground

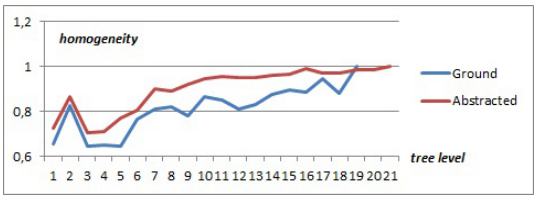

Figure 4: Comparison between average homogeneity values, computed level by level in the two cluster hierarchies obtained by UPGMA on ground traces and on abstracted traces, on a specific class-2 SU

vs. abstracted traces. As it can be observed, homogeneity on abstracted traces was higher then the one calculated on ground traces.

It is also interesting to study the management of outliers, i.e., in our application domain, traces that could correspond to the treatment of atypical patients, or to medical errors. These traces record rather uncommon actions, and/or present uncommon temporal constraints among their actions. For instance, in SUcl2, trace 105 is very peculiar: it describes the management of a patient suffering from several inter-current complications (diabetes, hypertension, venticular arrythmia, venous thrombosis), who required many extra-tests and many specialist counseling sessions, interleaved to more standard actions.

Ideally, these anomalous traces should remain isolated as a one-trace cluster for many UPGMA iterations, and be merged to other nodes in the hierarchy as late as possible, i.e., close to the root (level 0).

Indeed, when working on ground traces, outliers of SUcl2 were merged very late to the hierarchy. As shown in figure 5, 8 particularly significant outliers (according to our medical co-author), were merged between level 6 and level 1. Trace 105 was merged at level 5. Very interestingly, this capability of "isolating" outliers was preserved when working on abstracted traces. Indeed, the 8 outlying traces considered above were merged between level 6 and level 1 in the abstracted traces hierarchy as well - with minor variations with respect to the ground trace hierarchy; specifically, trace 105 was merged at level 4, highlighting its anomaly even better then in the ground trace case.

We then repeated the experiment on all the SUs, divided by level. As an example, we present the results on class-3 SUs. For the test shown in this paper, we randomly sampled 33 to 34 traces for each one of the 9 SUs in class-3 group, thus obtaining a working dataset of 300 traces. We then

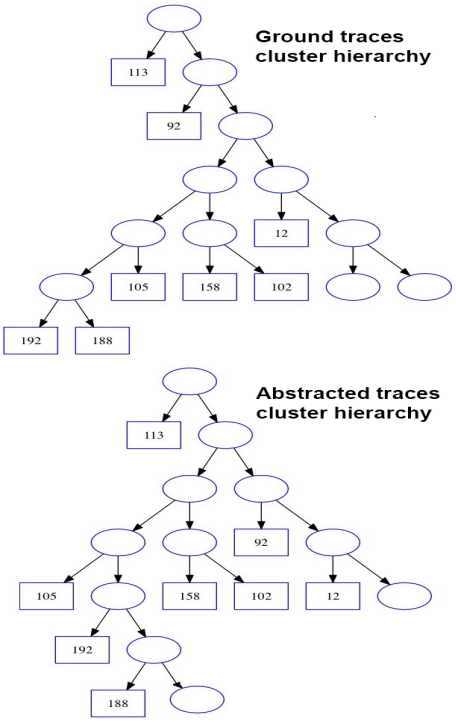

Figure 5: Identification of outliers (in rectangles) in cluster hierarchies. Only the upper hierarchy levels are shown

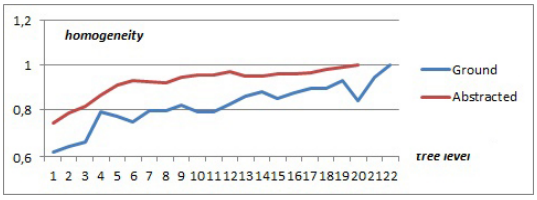

Figure 6: Comparison between average homogeneity values, computed level by level in the two cluster hierarchies obtained by UPGMA on ground traces and on abstracted traces, on 300 traces randomly chosen from class-3 SUs

applied UPGMA to the 300 ground and abstracted traces. The obtained cluster hierarchy height was 22 when working on ground traces, and 20 when working on abstracted ones. Figure 6 shows a comparison of the average of cluster homogeneity values, computed by level in the cluster hierarchies. As it can be observed, homogeneity on abstracted traces was always higher then the one calculated on ground traces, where the difference could be up to 0.2 (in a [0 1] range) in some levels of the hierarchies. The capability of isolating outliers was preserved in this experiment as well. Referring to 5 particularly significant outliers (again, according to our medical co-author), they were merged between level 5 and level 3 in the ground traces hierarchy, and between level 4 and level 3 in the abstracted traces hierarchy.

In conclusion, our hypothesis was verified by the experiments, since the application of the abstraction mechanism allowed to obtain more homogeneous clusters, still clearly isolating outlying traces.

3.2. Semantic process mining

As a second experimental work, we have tested whether our capability to abstract the event log traces on the basis of their semantic goal allowed to obtained process models where unnecessary details are hidden, but key behaviors are clear. Indeed, if this hypothesis holds, in our application domain it becomes easier to compare process models of different SUs, highlighting the presence/absence of common paths, regardless of minor action changes (e.g., different ground actions that share the same goal) or irrelevant different action ordering or interleaving (e.g., sets of ground actions, all sharing a common goal, that could be executed in any order).

Figure 7 compares the process models of two different SUs (SU1 and SU2), mined by resorting to Heuristic Miner (Weijters et al., 2006), operating on ground traces. Figure 8, on the other hand, compares the process models of

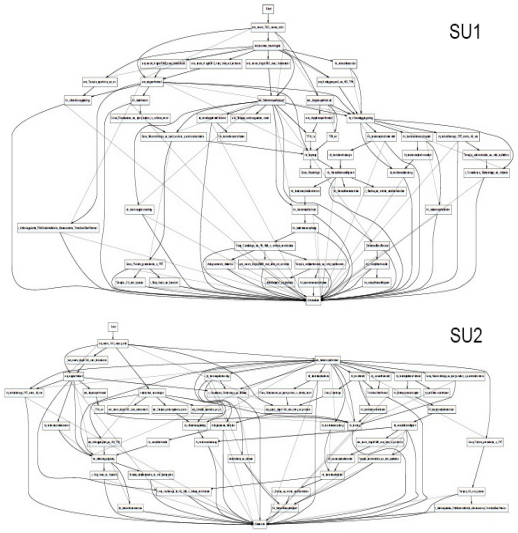

Figure 7: Comparison between two process models, mined by resorting to Heuristic Miner, operating on ground traces. The figure is not intended to be readable, but only to give an idea of how complex the models can be

the same SUs as Figure 7, again mined by resorting to Heuristic Miner, but operating on traces abstracted at the second level of the taxonomy in Figure 2 (where the root is considered as level 0).

Generally speaking, a visual inspection of the two graphs in Figure 7 is very difficult. Indeed, these two ground processes are "spaghetti-like" (der Aalst, 2011), and the extremely large number of nodes and edges makes it hard to identify commonalities in the two models. The abstract models in Figure 8, on the other hand, are much more compact, and it is possible for a medical expert to analyze them. In particular, the two graphs are not identical, but in both of them it is easy to a identify a path containing some macro-actions, which corresponds to the treatment of a typical stroke patient, namely: "Causes Identification" (which does not further specialize in subclasses according to the taxonomy in Figure 2), "Cardio-Embolic Mechanism" (subclass of "Pathogenetic Mechanism Identification"), "Early Relapse Prevention", "Long Term Relapse Prevention", "In-Hospital Mortality Reduction" (all subclasses of "Prevention"), "Dismissal" (subclass of "Administrative Actions"). The macro-actions at hand are highlighted in bold in the figure. The (different) interleaving of a few additional actions between the six steps is just due to minor changes in the two hospital practices.

The model for SU1 at the top in Figure 8 also shows a larger number of paths, while the model for SU2 at the bottom has fewer treatment options. This is a very reasonable outcome, since SU1 is a class-3 SU, where different kinds of patients, including atypical ones, can be managed, thanks to the availability of different skills and instrumental resources. On the other hand, SU2 is a class-2 SU, i.e., a more generalist one, where very specific human knowledge or technical resources are missing. As a consequence, its process model is more homogeneous, since atypical patients are not admitted here. For instance, one path shows that SU1 can perform extracranical vessel inspection, which is typically absent in a less specialized SU. On the other hand, SU2 performs a neuroprotection intervention, which is not prescribed anymore by the most recent guidelines: this is an indication that SU2 personnel may have less up-to-date knowledge. Very interestingly, our abstraction mechanism, while hiding irrelevant details, allows to still appreciate these differences.

4. Related work

The use of semantics in business process management, with the aim of operating at different levels of abstractions in process discovery and/or analysis, is a relatively young area of research, where much is still unexplored.

One of the first contributions in this field was proposed in Casati & Shan (2002), which introduces a process data warehouse, where taxonomies are exploited to add semantics to process execution data, in order to provide more intelligent reports. The work in Grigori et al. (2004) extends the one in Casati & Shan (2002), presenting a complete architecture that allows business analysts to perform multidimensional analysis and classify process instances, according to flat taxonomies (i.e., taxonomies without subsumption relations between concepts). The work in Sell et al. (2005) develops in a similar context, and extends OLAP tools with semantics (exploiting ontologies rather than (flat) taxonomies). Hepp et al. (Hepp et al., 2005) propose a framework able to merge semantic web, semantic web services, and business process management techniques to build semantic business process management, and use ontologies to provide machine-processable semantics in business processes (Hepp & Roman, 2007).

Semantic business process management is further developed in the SUPER project (Pedrinaci et al., 2008), within which several ontologies are created, such as the process mining ontology and the event ontology (Pedrinaci

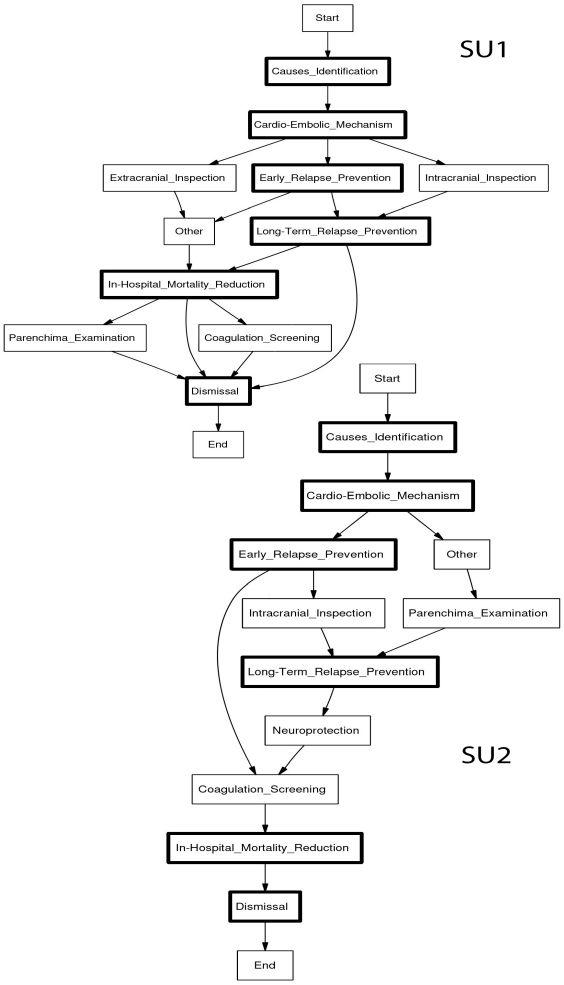

Figure 8: Comparison between the two process models of the same SUs as Figure 7, mined by resorting to Heuristic Miner, but operating on abstracted traces

& Domingue, 2007); these ontologies define core terminologies of business process management, usable by machines for task automation. However, the authors do not present any concrete implementations of semantic process mining or analysis.

Ontologies, references from elements in logs to concepts in ontologies, and ontology reasoners (able to derive, e.g., concept equivalence), are described as the three essential building blocks for semantic process mining and analysis in de Medeiros et al. (2008). This paper also shows how to use these building blocks to extend ProM's LTL Checker (van der Aalst et al., 2005) to perform semantic auditing of logs.

The work in de Medeiros et al. (2007) focuses on the use of semantics in business process monitoring, an activity that allows to detect or predict process deviations and special situations, to diagnose their causes, and possibly to resolve problems by applying corrective actions. Detection, diagnosis and resolution present interesting challenges that, on the authors' opinion, can strongly benefit from knowledge-based techniques.

In de Medeiros et al. (2007); de Medeiros & van der Aalst (2009) the idea to explicitly relate (or annotate) elements in the event log with the concepts they represent, linking these elements to concepts in ontologies, is also addressed.

In Okoye et al. (2015) the authors show through experiments how data from learning processes can be extracted, semantically prepared (by annotating the log referencing ontology concepts), and transformed into mining executable formats for improved analysis.

In de Medeiros & van der Aalst (2009) an example of process discovery at different levels of abstractions is presented. It is however a very simple example, where a couple of ground actions are abstracted according to their common ancestor. However, neither the management of interleaved actions or delays, nor the correct identification of temporal constraints generated when aggregating different macro-actions are addressed.

Moreover, most of the papers cited above (including (Hepp et al., 2005; Kharbili et al., 2008; de Medeiros et al., 2007, 2008; de Medeiros & van der Aalst, 2009)) present theoretical frameworks, and not yet a detailed technical architecture nor a concrete implementation of all their ideas.

In Bose & van der Aalst (2009) the authors characterize the manifestation of commonly used process model constructs in the event log and adopt pattern definitions that capture these manifestations, and propose a means to form abstractions over these patterns. In particular, the approach identifies

loops in traces, and replaces the repeated occurrences of the manifestation of the loop by an abstracted entity that encodes the notion of a loop. It also identifies common functionalities in the traces and replaces them with abstract entities. This work, however, does not make use of semantic information.

Another interesting approach to abstraction in process models is the one in Smirnov et al. (2012). The authors propose abstraction to generate more readable high-level views on business process models. They are able to discover sets of related actions, where each set corresponds to a coarse-grained task in an abstract process model. Specifically, abstraction resorts to a clustering technique, where action properties (such as, e.g., roles and resources) are exploited to aggregate the different actions into the common task. The authors adopt the enhanced Topic Vector Space Model to reflect the semantic relations between action property values: in this way, the distance between two different, but related values, can be lower that 1.

Differently from our approach, however, the abstraction solution described in Smirnov et al. (2012) is not applied to traces - and therefore cannot be adopted for trace comparison. Moreover, it requires that all action properties are available and logged - which, unfortunately, is often not the case, for instance in medicine, where logging may be incomplete in practice. Moreover, clustering does not take into account temporal relations between actions, in the sense that it may also aggregate actions executed at temporally distant phases of the model control flow; on the other hand, our approach, by operating on traces, which log the temporal sequence of action executions and their temporal constraints, strongly relies on temporal information, maintains it, and allows to exploit it in further analyses, such as abstracted trace comparison.

Thus, the work in Smirnov et al. (2012) adopts a significantly different technique to process model abstraction with respect to our proposal; nonetheless, it is certainly a relevant related work, and it would be interesting to compare abstraction results obtained through that method to our medical logs, in order to evaluate pros and cons of the two methodologies.

Referring to medical applications, the work in Grando et al. (2011) proposes an approach, based on semantic process mining, to verify the compliance of a Computer Interpretable Guideline with medical recommendations. In this case, semantic process mining refers to conformance checking rather than to process discovery (as it is also the case in de Medeiros et al. (2008)). These works are thus only loosely related to our contribution.

As regards trace comparison, as already observed, in this paper we have extended a metric we published in Montani & Leonardi (2014), able to exploit domain knowledge in action comparison, and to manage all types of temporal constraints. Other metrics for trace comparison have been proposed in the literature. In particular, (Kapetanakis et al., 2010) combines a contribution related to action similarity, and a contribution related to delays between actions. As regards the temporal component, it relies on an interval distance definition which is quite similar to ours. Differently from what we do, however, no search for the optimal action alignment is performed. The distance function in Kapetanakis et al. (2010) does not exploit action duration, and does not rely on semantic information about actions, as we do. Finally, it does not deal with different types of qualitative temporal constraints. Another interesting contribution is Combi et al. (2009), which addresses the problem of defining a similarity measure able to treat temporal information, and is specifically designed for clinical workflow traces. Interestingly, the authors consider qualitative temporal constraints between matched pairs of actions, resorting to the Neighbors-graph Distance, as we do. However, in Combi et al. (2009) the alignment problem is strongly simplified, as they only match actions with the same name. In this sense, our approach is also much more semantically oriented. Several metrics for comparing process models, instead of traces, also exist. Most of them are based on proper extensions of the edit distance as well (Minor et al., 2008; Bergmann & Gil, 2014; Montani et al., 2015a; Dijkman et al., 2009; LaRosa et al., 2013), and, in some cases, allow for a semantic comparison among model actions (Bergmann & Gil, 2014; Montani et al., 2015a). However, given the very different structure of a process model (which is a graph) with respect to a trace, these works are only loosely related to our contribution.

In conclusion, in the current research panorama, our work appears to be very innovative, for several reasons:

- many approaches, presenting very interesting and sometimes ambitious ideas, just provide theoretical frameworks, while concrete implementations of algorithms and complete architectures of systems are often missing;

- in semantic process mining, more work has been done in the field of conformance checking (also in medical applications), while process discovery still deserves attention (also because many approaches are still at the theoretical level, as commented above);

- as regards trace abstraction, it is often proposed as a very powerful means to obtain better process discovery and analysis results, but technical details of the abstraction mechanism are usually not provided, or are illustrated through very simple examples, where the issues we presented in the Introduction (related to the management of interleaved actions or delays, and to the correct identification of temporal constraints generated when aggregating different macro-actions) do not emerge;
- as regards trace comparison, to the best of our knowledge, our previously published metric (Montani & Leonardi, 2014), enhanced to deal with abstracted traces, still represents one of the most complete contributions to properly account for both non temporal and temporal information, and to perform a semantic comparison between actions.

5. Conclusions

In this paper, we have presented a framework for multi-level abstraction of event log traces. In our architecture, abstracted traces are then provided as an input to different analysis techniques – namely, trace comparison and semantic process mining in the current implementation. Our trace comparison facility relies on a metric that extends our previous contribution in Montani & Leonardi (2014): such a distance is able to manage both temporal and non temporal information in traces, and has been properly extended to work on abstracted traces as well. Semantic process mining relies on ProM algorithms; indeed, the overall integration of our approach within ProM is foreseen in our future work.

Experimental results on trace comparison (and more specifically on trace clustering) in the field of stroke management have shown that it is easier to identify common behaviors in abstracted traces, with respect to ground traces: in fact, cluster homogeneity, when operating on abstracted traces, reaches higher values. At the same time, outliers (i.e., anomalies and incorrect behaviors) are still clearly visible in abstracted traces as well (and clearly detected by the clustering method we used). Further experiments have proved that the capability of abstracting the event log traces on the basis of their semantic goal allows to mine clearer process models, where unnecessary details are hidden, but key behaviors are clear. In the future, we plan to conduct further experiments, e.g., by comparing different process models (of different SUs) obtained from abstracted traces. Comparison

will resort to knowledge-intensive process similarity metrics, such as the one we described in Montani et al. (2015b). We will also extensively test the approach in different application domains.

From a methodological viewpoint, we plan to extend our approach in different directions. First, we will consider different knowledge structures, such as ontologies, or multiple taxonomies, able to provide abstraction information from different viewpoints (e.g., not only the viewpoint of action goals - as it happens in the single taxonomy we are currently adopting - but also the one of roles and responsibilities of the involved actors, when available). As a consequence, the similarity metric will need proper extensions or adjustments (e.g., by considering the work in Hwang et al. (2012) in the case of multiple taxonomies). The modularity of our approach will make this extensions relatively easy. Second, we will consider more complex rules, e.g., having as an antecedent the execution of some action registered in the log. This will allow us to control the abstraction process on the basis of the context, i.e., of the already executed actions. Temporal constraints (e.g., the delay since the completion of the already executed action) will also be taken into account in these rules. We believe that such improvements will make our framework more complete and much more useful in practice.

References

van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, *16*, 1128–1142.

van der Aalst, W. M. P., de Beer, H. T., & van Dongen, B. F. (2005). Process mining and verification of properties: An approach based on temporal logic. In R. Meersman, Z. Tari, M. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H. Jacobsen, J. P. Loyall, M. Kifer, & S. Spaccapietra (Eds.), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I* (pp. 130–147). Springer volume 3760 of *Lecture Notes in Computer Science*.

der Aalst, W. V. (2011). *Process Mining. Discovery, Conformance and Enhancement of Business Processes*. Springer.

der Aalst, W. V., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. (2003). Workflow mining: a survey of issues and approaches. *Data and Knowledge Engineering*, *47*, 237–267.

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence*, *23*, 123–154.

Bergmann, R., & Gil, Y. (2014). Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, *40*, 115–127.

Bose, R. P. J. C., & van der Aalst, W. (2009). Abstractions in process mining: A taxonomy of patterns. In U. Dayal, J. Eder, J. Koehler, & H. A. Reijers (Eds.), *Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings* (pp. 159–175). volume 5701 of *Lecture Notes in Computer Science*.

Casati, F., & Shan, M. (2002). Semantic analysis of business process executions. In C. S. Jensen, K. G. Jeffery, J. Pokorný, S. Saltenis, E. Bertino, K. Böhm, & M. Jarke (Eds.), *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27, Proceedings* (pp. 287–296). Springer volume 2287 of *Lecture Notes in Computer Science*.

Combi, C., Gozzi, M., Oliboni, B., Juarez, J., & Marin, R. (2009). Temporal similarity measures for querying clinical workflows. *Artificial Intelligence in Medicine*, *46*, 37–54.

Dijkman, R., Dumas, M., & Garca-Banuelos, R. (2009). Graph matching algorithms for business process model similarity search. In U. Dayal, J. Eder, J. Koehler, & H. Reijers (Eds.), *Proc. International Conference on Business Process Management* (pp. 48–63). volume 5701 of *Lecture Notes in Computer Science*.

van Dongen, B., & van der Aalst, W. (2005). A meta model for process mining data. In M. Missikoff, & A. D. Nicola (Eds.), *EMOI - INTEROP'05, Enterprise Modelling and Ontologies for Interoperability, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-located with CAiSE'05 Conference, Porto (Portugal), 13th-14th June 2005*. CEUR-WS.org volume 160 of *CEUR Workshop Proceedings*.

van Dongen, B., De Medeiros, A. A., Verbeek, H., Weijters, A., & der Aalst, W. V. (2005). The proM framework: a new era in process mining tool support. In G. Ciardo, & P. Darondeau (Eds.), *Knowledge Mangement and its Integrative Elements* (pp. 444–454). Springer, Berlin.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern classication*. Wiley-Interscience, New York.

Francis, P., Leon, D., Minch, M., & Podgurski, A. (2004). Tree-based methods for classifying software failures. In *Int. Symp. on Software Reliability Engineering* (pp. 451–462). IEEE Computer Society.

Grando, M. A., Schonenberg, M. H., & van der Aalst, W. M. P. (2011). Semantic process mining for the verification of medical recommendations. In V. Traver, A. L. N. Fred, J. Filipe, & H. Gamboa (Eds.), *HEALTHINF 2011 - Proceedings of the International Conference on Health Informatics, Rome, Italy, 26-29 January, 2011* (pp. 5–16). SciTePress.

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., & Shan, M. (2004). Business process intelligence. *Computers in Industry*, *53*, 321–343.

Hepp, M., Leymann, F., Domingue, J., Wahler, A., & Fensel, D. (2005). Semantic business process management: A vision towards using semantic web services for business process management. In F. C. M. Lau, H. Lei, X. Meng, & M. Wang (Eds.), *2005 IEEE International Conference on e-Business Engineering (ICEBE 2005), 18-21 October 2005, Beijing, China* (pp. 535–540). IEEE Computer Society.

Hepp, M., & Roman, D. (2007). An ontology framework for semantic business process management. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, & B. Schnizler (Eds.), *eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik - Band 1, WI 2007, Karlsruhe, Germany, February 28 - March 2, 2007* (pp. 423–440). Universitaetsverlag Karlsruhe.

Hwang, S. J., Grauman, K., & Sha, F. (2012). Semantic kernel forests from multiple taxonomies. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information*

Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. (pp. 1727–1735).

IEEE Taskforce on Process Mining: Process Mining Manifesto (). *http : //www.win.tue.nl/ieeetfpm.* IEEE Taskforce on Process Mining: Process Mining Manifesto (last accessed on 4/11/2013).

Kapetanakis, S., Petridis, M., Knight, B., Ma, J., & Bacon, L. (2010). A case based reasoning approach for the monitoring of business workflows. In I. Bichindaritz, & S. Montani (Eds.), *Proc. International Conference on Case Based Reasoning (ICCBR)* (pp. 390–405). Springer, Berlin volume 6176 of *Lecture Notes in Computer Science*.

Kharbili, M. E., Stein, S., & Pulvermüller, E. (2008). Policy-based semantic compliance checking for business process management. *MobIS Workshops*, *420*, 178–192.

Lanz, A., Weber, B., & Reichert, M. (2010). Workflow time patterns for process-aware information systems. In *Proc. BMMDS/EMMSAD* (pp. 94–107).

LaRosa, M., Dumas, M., Uba, R., & Dijkman, R. (2013). Business process model merging: An approach to business process consolidation. *ACM Trans. Softw. Eng. Methodol.*, *22*, 11.

Levenshtein, A. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, *10*, 707–710.

de Medeiros, A. K. A., & van der Aalst, W. M. P. (2009). Process mining towards semantics. In T. S. Dillon, E. Chang, R. Meersman, & K. P. Sycara (Eds.), *Advances in Web Semantics I - Ontologies, Web Services and Applied Semantic Web* (pp. 35–80). Springer volume 4891 of *Lecture Notes in Computer Science*.

de Medeiros, A. K. A., van der Aalst, W. M. P., & Pedrinaci, C. (2008). Semantic process mining tools: Core building blocks. In W. Golden, T. Acton, K. Conboy, H. van der Heijden, & V. K. Tuunainen (Eds.), *16th European Conference on Information Systems, ECIS 2008, Galway, Ireland, 2008* (pp. 1953–1964).

de Medeiros, A. K. A., Pedrinaci, C., van der Aalst, W. M. P., Domingue, J., Song, M., Rozinat, A., Norton, B., & Cabral, L. (2007). An outlook on semantic business process mining and monitoring. In R. Meersman, Z. Tari, & P. Herrero (Eds.), *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part II* (pp. 1244–1255). Springer volume 4806 of *Lecture Notes in Computer Science*.

Minor, M., Tartakovski, A., Schmalen, D., & Bergmann, R. (2008). Agile workflow technology and case-based change reuse for long-term processes. *International Journal of Intelligent Information Technologies*, *4*, 80–98.

Montani, S., & Leonardi, G. (2014). Retrieval and clustering for supporting business process adjustment and analysis. *Information Systems*, *40*, 128–141.

Montani, S., Leonardi, G., Quaglini, S., Cavallini, A., & Micieli, G. (2015a). A knowledge-intensive approach to process similarity calculation. *Expert Syst. Appl.*, *42*, 4207–4215.

Montani, S., Leonardi, G., Quaglini, S., Cavallini, A., & Micieli, G. (2015b). A knowledge-intensive approach to process similarity calculation. *Expert Syst. Appl.*, *42*, 4207–4215.

Okoye, K., Tawil, A. H., Naeem, U., & Lamine, E. (2015). Semantic process mining towards discovery and enhancement of learning model analysis. In *17th IEEE International Conference on High Performance Computing and Communications, HPCC 2015, 7th IEEE International Symposium on Cyberspace Safety and Security, CSS 2015, and 12th IEEE International Conference on Embedded Software and Systems, ICESS 2015, New York, NY, USA, August 24-26, 2015* (pp. 363–370). IEEE.

Palmer, M., & Wu, Z. (1995). Verb Semantics for English-Chinese Translation. *Machine Translation*, *10*, 59–92.

Pedrinaci, C., & Domingue, J. (2007). Towards an ontology for process monitoring and mining. In M. Hepp, K. Hinkelmann, D. Karagiannis,

R. Klein, & N. Stojanovic (Eds.), *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007, held in conjunction with the 3rd European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, June 7, 2007*. volume 251 of *CEUR Workshop Proceedings*.

Pedrinaci, C., Domingue, J., Brelage, C., van Lessen, T., Karastoyanova, D., & Leymann, F. (2008). Semantic business process management: Scaling up the management of business processes. In *Proceedings of the 2th IEEE International Conference on Semantic Computing (ICSC 2008), August 4-7, 2008, Santa Clara, California, USA* (pp. 546–553). IEEE Computer Society.

Sell, D., Cabral, L., Motta, E., Domingue, J., & dos Santos Pacheco, R. C. (2005). Adding semantics to business intelligence. In *16th International Workshop on Database and Expert Systems Applications (DEXA 2005), 22-26 August 2005, Copenhagen, Denmark* (pp. 543–547). IEEE Computer Society.

Sharan, R., & Shamir, R. (2000). CLICK: A clustering algorithm for gene expression analysis. In *Proc. International Conference on Intelligent Systems for Molecular Biology* (p. 260268).

Smirnov, S., Reijers, H. A., & Weske, M. (2012). From fine-grained to abstract process models: A semantic approach. *Inf. Syst.*, *37*, 784–797.

Sokal, R., & Michener, C. (1958). A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, *38*, 1409–1438.

Verbeek, H. M. W., Buijs, J. C. A. M., van Dongen, B. F., & van der Aalst, W. M. P. (2011). Xes, xesame, and prom 6. In P. Soffer, & E. Proper (Eds.), *Information Systems Evolution: CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers* (pp. 60–75). Berlin, Heidelberg: Springer Berlin Heidelberg.

Weber, B., & Wild, W. (2005). Towards the agile management of business processes. In K. D. Althoff, A. Dengel, R. Bergmann, M. Nick, & T. Roth-Berghofer (Eds.), *Professional knowledge management WM 2005, LNCS 3782* (pp. 409–419). Washington DC: Springer, Berlin.

Weijters, A., der Aalst, W. V., & de Medeiros, A. A. (2006). *Process Mining with the Heuristic Miner Algorithm, WP 166*. Eindhoven University of Technology, Eindhoven.

Yip, A., Chan, T., & Mathew, T. (2003). *A Scale Dependent Model for Clustering by Optimization of Homogeneity and Separation, CAM Technical Report 03-37*. Department of Mathematics, University of California, Los Angeles.