Multi-level abstraction for trace comparison and process discovery

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, Italy

Keywords:
Abstraction
Trace comparison
Process discovery
Stroke management

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 activities in the event log into higher level concepts, at different levels of abstraction, on the basis of domain knowledge. Our abstraction mechanism manages non trivial situations, such as interleaved activities or delays between two activities that abstract to the same concept.
Abstracted traces can then be provided as an input to an intelligent system, meant to implement a variety of process management tasks, significantly enhancing the quality and the usefulness of its output.
In particular, in the paper we demonstrate how trace abstraction can impact on the quality of process discovery, showing that it is possible to obtain more readable and understandable process models.
We also prove, through our experimental results, the impact of our approach on the capability of trace comparison and clustering (realized by means of a metric able to take into account abstraction phase penalties) to highlight (in)correct behaviors, abstracting from details.

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 1 . The event log stores the sequences (traces van der Aalst, 2011 henceforth) of activities 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 (van 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 (van der Aalst et al., 2003; van der Aalst, Weijters, & Maruster, 2004) algorithms, a family of mainly 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 activities in the event log are compared and processed only referring to their names. Activity names are strings without any semantics, so that identical activities, labeled by synonyms, will be considered as different, or activities that are special cases of other activities will be processed as unrelated.
Upgrading trace comparison and process mining to the conceptual layer can enhance existing algorithms towards more advanced and reliable approaches. Indeed, the capability of relating semantic structures such as taxonomies or ontologies to activities in the log can enable both trace comparison and process discovery 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 and van der Aalst, 2009; de Medeiros, van der Aalst, and Pedrinaci, 2008 , and Section 5). However, while more work has been done in the field of semantic conformance checking (Grando, Schonenberg, & van der Aalst, 2011; de Medeiros et al., 2008), to the best of our knowledge semantic process discovery needs to be further investigated. In this paper:
1. We present a semantic-based, multi-level abstraction mechanism, able to operate on event log traces. In our approach, activities 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;
2. We provide the abstraction mechanism as an input to further analysis mechanisms, namely trace comparison and process discovery.
The abstraction mechanism has been designed to properly tackle non-trivial issues that could emerge. Specifically:
• Two activities 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 activity takes place), or by activities that descend from a different ancestor (interleaved activities henceforth). Our approach allows to deal with these situations, by creating a single macro-activity , i.e., an abstract activity that covers the whole time span of the two activities at hand, and is labeled as the common ancestor; the macroactivity is however built only if the total delay length, or the total number/length of interleaved activities, do not overcome proper admissibility thresholds set by the user. The delays and interleaved activities are quantified and recorded, for possible use in further analyses. In particular, we present a metric where this information is accounted for as a penalty, and affects the distance value in abstract trace comparison;
• Abstraction may generate different types of temporal constraints between pairs of macro-activities; specifically, given the possible presence of interleaved activities, we can obtain an abstracted trace with two (or more) overlapping or concurrent macro-activities. 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 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:

•	Having defined a proper mechanism for abstracting event log traces , able to manage non trivial situations (originating from the treatment of interleaved activities or delays between two activities sharing the same ancestor);
•	Having provided a trace comparison facility , which resorts to a metric (extending the one 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 discovery, we currently rely on classical algorithms embedded in the open source framework ProM (van Dongen, De Medeiros, Verbeek, Weijters, & der Aalst, 2005). 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.

Fig. 1. Framework architecture and data flow.

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-activity. We also provide 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 an application of the abstraction facility as a pre-processing step for process model discovery. Section 4 provides experimental results. Section 5 addresses comparisons with related work. Section 6 is devoted to discussion. Finally, Section 7 presents our conclusions and future research directions.

## 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 Fig. 1 .

The first step to be executed is event log preparation , that takes as input the available database (recording executed activities and additional data), and exploits domain knowledge (in the form of a taxonomy); the event log then undergoes multi-level abstraction , which resorts to domain knowledge as well. The abstracted event log can be given as an input to different process management tasks, such as trace comparison and process discovery (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

Trace : a sequences of activities that belong to a same process execution.

Activity or ground activity : an activity 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 activities logged in a trace, where no other activity takes place.

Interleaved activity : a ground activity that descends from a different ancestor in the taxonomy of activities described in Section 2.2 , with respect to the two ground activities that are currently being considered for abstraction. In the trace, the interleaved activity is placed between the two activities at hand.

Macro-activity : partial output of the abstraction process; a macro-activity is an abstracted activity that covers the whole time span of multiple ground activities, and is labeled as their common

Fig. 2. An excerpt from the stroke domain taxonomy.
ancestor in the taxonomy, at the specified abstraction level. As a special case, the macro-activity can abstract a single ground activity as its ancestor.
Abstracted trace : global output of the abstraction process; an abstracted trace is the transformation of an input trace into a new trace containing macro-activities.
Metric or distance function : a function that defines a distance between each pair of elements of a set, i.e., it provides a numerical description of how far apart two objects are.
Similarity : if the distance between two objects is d , we define similarity as 1 − d.

2.2. Domain knowledge

In our framework, domain knowledge is provided by means of a taxonomy. In the paper, all examples will refer to the domain of stroke management.
An excerpt of our stroke management taxonomy is reported in
Fig. 2 .
The taxonomy, which has been formalized by using the Protégé 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 activities, 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 activities, including “Anticoagulant Medicines” and “Diabetologist Counseling” see Fig. 2). Overall, our taxonomy is composed by 136 classes, organized in a hierarchy of four levels.

2.3. Event log preparation

As illustrated in Fig. 1 , the event log preparation module takes as input the database (containing activity 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 domain knowledge, i.e., the taxonomy.
In this phase, the starting/ending times of activities are used to calculate activity ordering within every trace.
The log preparation module generates an event log where traces are represented in an eXtensible Event Stream (XES) (Verbeek, Buijs, van Dongen, & van der Aalst, 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 activity to a concept in an ontology: in our case, to a leaf in the taxonomy. Proper activity attributes also allow to record precondition values; if preconditions do not hold, abstraction will not take place (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 _algorithm takes as input an
ALGORITHM 1: Multi-level abstraction algorithm.

```
1    abs _trace  = abs _ algorithm (trace, taxo, l e v el , del ay _th , n _int  er _th , int er _th  ) ;
2    abs _trace  = ∅ ;
3    for e v ery i ∈ act i v it ies in trace do
4    if (( i.precond = ∅ ∨ i.precond = f ul f il l ed) ∧
( i.startF lag = yes )) then
5    create : m i as ancestor(i, l e v el ) ;
```

```
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 ( e v ery j ∈ elements 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 = fulfilled) ∧
                    ( ancestor(j, l e v el ) = ancestor(i, l e v el ) ) 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                      t otal _inter      = t otal _inter             + j.length ;
23                  end
24              end
25          end
26      else if ( precond = ¬ f ul f il l ed) 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        ;
```

event log trace , the domain taxonomy taxo , and the level in the taxonomy chosen for the abstraction (e.g., le v el = 1 corresponds to the choice of abstracting the activities 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-activity, the total number of interleaved activities,

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

and the total duration of interleaved activities, respectively. In fact, it would be hard to justify that two ground activities share the same goal (and can thus be abstracted to the same macro-activity), if they are separated by very long delays, or if they are interleaved by many/long different ground activities, meant to fulfill different goals.

The function outputs an abstracted trace.

For every activity 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, or if the preconditions of i are fulfilled, then a macro-activity 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-activity (totaldelay, 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 activity, or a delay between a pair of consecutive activities. 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 activity, the set of preconditions of j is empty or its preconditions are fulfilled, and j shares the same ancestor of i at the input abstraction level , then j is incorporated into the macro-activity 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 activities in trace that could start a new macroactivity, since it has already been incorporated into an existing one (line 18). This kind of situation is described in Fig. 3 (a).
- If j is an activity, but does not share the same ancestor of i , or violates some preconditions, then it is treated as an interleaved activity. 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-activities, as the ones described in Fig. 3 (b) (Allen's during Allen, 1984) and Fig. 3 (c) (Allen's overlaps Allen, 1984).

On the other hand, if some of the preconditions of i are not fulfilled, i cannot be abstracted referring to its ancestor in the taxonomy. In this case a singleton is created, i.e., a dummy macroactivity m i , sharing the starting and ending times of i , that will not aggregate with any other activity (lines 26–30).
Finally, the macro-activity m i is appended to abs _trace , that, in the end, will contain the list of all the macro-activities and singletons that have been created by the procedure (line 31).
Complexity . The cost of abstracting a trace is
O (activities ∗elements), where activities is the number of activities in the input trace, and elements is the number of elements (i.e., activities + delay intervals) in the input trace.

2.5. Trace comparison

In our approach, every trace (abstracted trace) is a sequence of activities (macro-activities, respectively), each one stored with its execution starting and ending times. Therefore, an activity is basically a symbol (plus possible execution parameters, in particular the temporal information). Starting and ending times allow to get information about activity 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, Weber, & Reichert, 2010) between pairs of consecutive activities/macro-activities.
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 and Leonardi (2014) , we first take into account activity 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 activities in a taxonomy: we can therefore adopt a more semantic approach, and apply Palmer's distance (Palmer & Wu, 1995), to impose that the closer two activities are in the

taxonomy in terms of the steps that separate them via their common ancestor, 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 activity in one trace has been matched to an activity 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 activities 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 activities on the traces being compared (e.g., activities A and B in trace P ; the aligned activities A and B in trace Q). We quantify the distance between their qualitative constraints (e.g., A and B overlap in trace P;   meets   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 activities share the same qualitative constraint (e.g., A and B overlap in trace P;   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 and Leonardi (2014) , these three contributions (i.e., Trace Edit Distance, Interval Distance between durations, Neighbors-graph Distance or Interval Distance between pairs of activities) are finally combined as a linear combination with non-negative weights.
When working on macro-activities, however, the metric in Montani and Leonardi (2014) needs to be extended, by considering, given the optimal macro-activities alignment, two additional contributions:
•	A penalty due to the different length of the delays incorporated into the two aligned macro-activities;
•	A penalty due to the different length of interleaved activities in the two aligned macro-activities being compared.
Delay penalty is defined as follows:
Definition 1. Delay penalty . Let A and B be two macro-activities, that have been matched in the optimal alignment. Let delay A = 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: d elay p (A, B) = d elay A − d elay B | maxd elay
As for interleaved activities penalty, we operate analogously to delay penalty, by summing up the lengths of all interleaved activities that have been incorporated within a single macro-activity in the abstraction phase.
Definition 2. Interleaving length penalty . Let A and B be two macro-activities, that have been matched in the optimal alignment.
Let inter A   ki = 1 lengththat (i)have be thebeen sum incorporated of the lengths into Aof in all the the abk
interleaved activities
straction 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 activities incorporated in an abstracted trace. The Interleaving Length Penalty interL p (A, B) between A and B is defined as: int erL p (A, B) = int er A − int er B | maxint er
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 discovery

In our approach, we are currently 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 plugins.

In this paper, we have exploited ProM's Heuristic Miner (Weijters, der Aalst, & de Medeiros, 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 activities within every single trace. It can mine the presence of short-distance and long-distance dependencies (i.e., direct or indirect sequence of activities), and information about parallelism, with a certain degree of reliability. The output is provided as a graph, known as the "dependency graph", where nodes represent activities, 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

Fig. 4. 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.

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. A process discovery example

In this section, we showcase how the capability of abstracting the traces on the basis of their semantic goal has allowed us to obtain clearer medical process models, where unnecessary details are hidden, but key behaviors are clear.

Fig. 4 compares the process models of two different Stroke
Units (SUs), namely SU1 and SU2, mined by resorting to
Weijters et al. (2006) , operating on ground traces. Fig. 5 , on the other hand, compares the process models of the same SUs as Fig. 4 , again mined by resorting to Heuristic Miner, but operating on traces abstracted at the second level of the taxonomy in Fig. 2 (where the root is considered as level 0).

Generally speaking, a visual inspection of the two graphs in Fig. 4 is very difficult. Indeed, these two ground processes are "spaghetti-like" (van 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 Fig. 5 , 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-activities, 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 Fig. 2), "Cardio-Embolic Mechanism" (subclass of "Pathogenetic Mechanism Identification"), "Early Relapse Prevention", "Long Term Relapse Prevention", "InHospital Mortality Reduction" (all subclasses of "Prevention"), "Dismissal" (subclass of "Administrative Actions"). The macro-activities at hand are highlighted in bold in the

figure. The (different) interleaving of a few additional activities between the six steps is just due to minor changes in the two hospital practices.
The model for SU1 at the top in Fig. 5 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 well-equipped 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

Fig. 5. Comparison between the two process models of the same SUs as Fig. 4 , mined by resorting to Heuristic Miner, but operating on abstracted traces.

more generalist SU, 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. Experimental 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 4.1) we have validated the abstraction algorithm, by comparing the output of our system to manual trace abstraction, conducted by a human expert. In the second work (see Section 4.2), 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-activity.
The available event log was composed of more than 15,0 0 0 traces, collected at the 40 Stroke Unit Network (SUN) collaborating centers of the Lombardia region, Italy. Traces were composed of 13 activities 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 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 coauthor, 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.
Results are provided in the following.

4.1. Abstraction

As a first experimental work, we have validated the abstraction algorithm, by comparing the output of our system to manual trace abstraction, conducted by an expert.
In this first experiment, we randomly chose 20 of the available traces.
These traces were analyzed by the expert, who annotated each of them with all the macro-activities he could recognize. The same traces were analyzed by the system. As a final step, the two sets of abstracted traces were compared pair by pair, to check consistency.

On 18 abstracted traces pairs, we identified no differences: the output provided by the expert was identical to the one generated by the system. This is a reasonable outcome, since the system operates on the basis of the medical knowledge formalized in the taxonomy. In 2 cases, however, the expert was unable to correctly identify all the macro-activities: in these situations, the presence of many delays and/or interleaved activities led him to split a macroactivity, identified as unique by the system, into a set of separated and shorter macro-activities, since he was unable to manually verify that the abstraction thresholds had not been exceeded, or it proved difficult to him to identify all the goals pursued during the trace execution, and to recognize when several goals overlapped or took place simultaneously.
An example of this situation is shown in Figs. 6 and 7 . Fig. 6 shows part of a rather complex trace, together with the annotations provided by the expert, displayed below the trace itself. Fig. 7 shows how the system was able to abstract the very same trace. Our system recognized a larger number of macroactivities (i.e., “Long Term Relapse Prevention” and “In-Hospital Mortality Reduction”, not identified by the human expert). Moreover, the system was able to capture a complex pattern, suggesting that, while the physicians were preparing the patient for discharge through actions fulfilling the “Long Term Relapse Prevention” goal, he suffered from sudden complications (e.g., pneumonia), that had to be managed, requiring a phase of “In-Hospital Mortality Reduction”.
After having reviewed the results, the expert recognized that the system output was correct, since more/longer macro-activities and more complex temporal relations had been properly identified, overcoming his own abstraction capability.

Fig. 6. Abstraction of a complex trace by the human expert.

Fig. 7. Abstraction of a complex trace by the system.

4.2. Trace comparison

As a second experimental work, we have analyzed the impact of our abstraction mechanism on trace comparison, and on the quality of trace clustering.
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., Duda, Hart, & Stork, 2001; Francis, Leon, Minch, & Podgurski, 2004;

Sharan & Shamir, 20 0 0; Yip, Chan, & Mathew, 2003). A classical definition of cluster homogeneity is the following (Yip et al., 2003):
dist(x, y))
H
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.
We also computed some statistics, referring to trace comparison results, cluster by cluster. Namely, we measured:
- The mean distance value between any two traces in the cluster;
- The variance of distance values within the cluster.

First, we worked on single SUs. As an example, we report on the results of applying UPGMA to the 240 traces of SUcl2, a class2 SU. The obtained cluster hierarchy height was 19 when working on ground traces, and 21 when working on abstracted ones (see
Fig. 8).
Fig. 9 shows a comparison of the average homogeneity values, computed by level in the cluster hierarchies, on ground 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 activities, and/or present uncommon temporal constraints among their activities. 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 activities. 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 Fig. 8 , eight 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 eight 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.
Fig. 8 also provides two tables, illustrating trace comparison statistics in the various clusters. As it can be observed, the mean

Fig. 8. Identification of outliers (in rectangles) in cluster hierarchies (only the upper hierarchy levels are shown); trace comparison statistics in clusters.

Fig. 9. 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.

Fig. 10. 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.

distance between any two pairs of traces in the clusters of the abstracted traces hierarchy (bottom of the figure) is always lower than the mean distance between any two pairs of traces in the clusters of the ground traces hierarchy (top of the figure). All variance values are small. These outcomes are well aligned to the homogeneity results, and reinforce our hypothesis about the advantages of abstraction in providing more robust results.

We then repeated our 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 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. Fig. 10 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. Statistics on trace comparison provided results in line with the ones commented for SUcl2.

4.3. Conclusions on the experimental results

Experiments on trace abstraction showed that the system performance overcomes the one of a medical expert. In 83% of the test traces, human and automatic abstraction results were identical, but in 17% of the traces the system was able to identify longer macroactivities, while the human expert was misled by the presence of many delays and interleaved activities. After having reviewed the results, the expert recognized that the system output was indeed correct.

As regards the experiments on trace comparison and clustering, our hypothesis, stating that the application of the abstraction mechanism allows to obtain more homogeneous clusters (i.e., able to aggregate closer examples), where, however, outliers are still clearly identifiable, was verified. Indeed, cluster homogeneity grew when working on abstracted traces.Trace comparison results improved as well, leading to a lower mean distance between any two pairs of traces in the abstracted traces clusters. Trace comparison statistics, as expected, are thus well aligned to homogeneity results, and further testify the robustness of comparison and clustering operating after the abstraction phase.

Experiments were run on a machine equipped with an Intel(R) Xeon(R) CPU E5-2640v2, CPU @ 2GHz, 4GB RAM. Abstraction time took 32 seconds when working on 50 randomly selected traces, and up to 34 minutes when working on 500 traces (times can slightly vary depending on traces length and interleaved activi

Fig. 11. Abstraction times (in minutes) as a function of the number of traces in the event log.

ties/delays). Details are shown in Fig. 11 . These times testify the efficiency of the abstraction algorithm in practice which, anyway, is not supposed to be applied in time-critical situations.

5. 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 and 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 and 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, Cabral, Motta, Domingue, and dos Santos Pacheco (2005) develops in a similar context, and extends OLAP tools with semantics (exploiting ontologies rather than (flat) taxonomies). Hepp, Leymann, Domingue, Wahler, and Fensel (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 & 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, de Beer, & van Dongen, 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) and de Medeiros and 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 de Medeiros and 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 activities are abstracted according to their common ancestor. However, neither the management of interleaved activities or delays, nor the correct identification of temporal constraints generated when aggregating different macro-activities are addressed.
Moreover, most of the papers cited above (including Hepp et al., 2005; de Medeiros & van der Aalst, 2009; de Medeiros et al., 2008; de Medeiros et al., 2007) present theoretical frameworks, and not yet a detailed technical architecture nor a concrete implementation of all their ideas.
Bose and van der Aalst (2009) 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, Reijers, and Weske (2012) . The authors propose abstraction to generate more readable high-level views on business process models. They are able to discover sets of related activities, where each set corresponds to a coarse-grained task in an abstract process model. Specifically, abstraction resorts to a clustering technique, where activity properties (such as, e.g., roles and resources) are exploited to aggregate the different activities into the common task. The authors adopt the enhanced Topic Vector Space Model to reflect the semantic relations between activity 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 activity 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 activities, in the sense that it may also aggregate activities 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 activity 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 and Leonardi (2014) , able to exploit domain knowledge in activity comparison, and to manage all types of temporal constraints. Other metrics for trace comparison have been proposed in the literature. In particular Kapetanakis, Petridis, Knight, Ma, and Bacon (2010) , combines a contribution related to activity similarity, and a contribution related to delays between activities. 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 activity alignment is performed. The distance function in Kapetanakis et al. (2010) does not exploit activity duration, and does not rely on semantic information about activities, as we do. Finally, it does not deal with different types of qualitative temporal constraints. Another interesting contribution is Combi, Gozzi, Oliboni, Juarez, and Marin (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 activities, 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 activities with the same name. In this sense, our approach is 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 (Bergmann & Gil, 2014; Dijkman, Dumas, & Garca-Banuelos, 2009; LaRosa, Dumas, Uba, & Dijkman, 2013; Minor, Tartakovski, Schmalen, & Bergmann, 2008; Montani, Leonardi, Quaglini, Cavallini, & Micieli, 2015a), and, in some cases, allow for a semantic comparison among model activities (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.
The main approaches discussed in this section are summarized in Fig. 12 .

6. Discussion

To summarize our comparison to related works, in the current research panorama, our approach 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 activities or delays, and to the correct identification of temporal constraints generated when aggregating different macro-activities) 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 activities.

Nonetheless, some limitations can be observed, and should be tackled in our future work:

- We currently rely on a taxonomy to formalize medical knowledge; this limits the type of knowledge that we can represent. In the future, we will need to consider multiple inheritance, and more complex relations between activities, goals, roles and responsibilities, resorting to ontologies;
- We currently do not consider the context of activity execution (i.e., what activities were executed earlier in the trace, and when), before starting abstraction. To tackle this issue, in the future we will introduce a rule-based approach. Rules, having as an antecedent the execution of some activities registered earlier in the trace, or specific patient's characteristics, will properly fire, to initiate the abstraction step. Temporal constraints (e.g., the delay since the completion of the already executed activity) will also be taken into account in these rules.

We believe that such improvements will extend the applicability and the flexibility of our approach.

7. 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 process discovery, in the current implementation. Our trace comparison facility relies on a metric that extends our previous contribution in Montani and 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. Process discovery relies on ProM algorithms; indeed, the overall integration of our approach within ProM is foreseen in our future work.
Experimental results on abstraction and on its application to trace comparison and 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 particular, 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 examples have illustrated that the capability of abstracting the event log traces on the basis of their semantic goal allows to discover clearer process models, where unnecessary details are hidden, but key behaviors are clear. In the future, we plan to conduct more 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, Leonardi, Quaglini, Cavallini, and Micieli (2015b) . We will also extensively test the approach in different application domains.

Fig. 12. Comparison between our approach and the main related works in the literature.

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 activity 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, Grauman, and Sha, 2012 in the case of multiple taxonomies). The modularity of our approach will make this extensions relatively easy. Second, we will introduce a rule-based approach to initiate abstraction. Proper rules, having as an antecedent the execution of some activity registered earlier in the log, will fire, to initiate the abstraction step. This will allow us to control the abstraction process on the basis of the context, i.e., of the already executed activities. Temporal constraints (e.g., the delay since the completion of the already executed activity) 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.

Acknowledgements

This research is original and has a financial support of the University of Piemonte Orientale.

References

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 . In Lecture Notes in Computer Science: vol. 5701 (pp. 159–175) .

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 . In Lecture Notes in Computer Science: vol. 2287 (pp. 287–296). Springer .

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 .

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 20 07: OTM 20 07 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 . In Lecture Notes in Computer Science: vol. 4806 (pp. 1244–1255). Springer .
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 . In Lecture Notes in Computer Science: vol. 4891 (pp. 35–80). Springer .
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) .
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.), Proceedings of international conference on business process management . In Lecture Notes in Computer Science: vol. 5701 (pp. 48–63) .
Duda, R. , Hart, P. , & Stork, D. (2001). Pattern classiffication . New York: Wiley-Interscience .
Francis, P. , Leon, D. , Minch, M. , & Podgurski, A. (2004). Tree-based methods for classifying software failures. In International symposium 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.), e-organisation: 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) .
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.), Proceedings of international conference on case based reasoning (ICCBR) . In Lecture Notes in Computer Science: vol. 6176 (pp. 390–405). Springer, Berlin .
Lanz, A. , Weber, B. , & Reichert, M. (2010). Workflow time patterns for process-aware information systems. In Proceedings of 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 Transactions on Software Engineering and Methodology, 22 , 11 .

Levenshtein, A. (1966). Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 10 , 707–710 .
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 Systems with Applications, 42 , 4207–4215 .
Montani, S. , Leonardi, G. , Quaglini, S. , Cavallini, A. , & Micieli, G. (2015b). A knowledge-intensive approach to process similarity calculation. Expert Systems with Applications, 42 , 4207–4215 .
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 . In CEUR Workshop Proceedings: vol. 251 .
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. (20 0 0). CLICK: A clustering algorithm for gene expression analysis. In Proceedings of international conference on intelligent systems for molecular biology (pp. 260–268) .
Smirnov, S. , Reijers, H. A. , & Weske, M. (2012). From fine-grained to abstract process models: A semantic approach. Information Systems, 37 , 784–797 .
Sokal, R. , & Michener, C. (1958). A statistical method for evaluating systematic relationships. University of Kansas Science Bulletin, 38 , 1409–1438 .
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, O. 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 . In Lecture Notes in Computer Science: vol. 3760 (pp. 130–147). Springer .
van der Aalst, W. , 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 .
van der Aalst, W. , Weijters, T. , & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16 , 1128–1142 .
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. 4 4 4–454). Springer, Berlin . 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 . In CEUR Workshop Proceedings: vol. 160 . CEUR-WS.org .
van der Aalst, W. (2011). Process mining. Discovery, conformance and enhancement of business processes . Springer .
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: Eindhoven University of Technology .
Yip, A. , Chan, T. , & Mathew, T. (2003). A scale dependent model for clustering by optimization of homogeneity and separation, CAM technical report 03–37 . Los Angeles:
Department of Mathematics, University of California .