HAL
archives-ouvertes.fr

A Case-Based Approach to Business Process Monitoring

Stefania Montani, Giorgio Leonardi

▶ **To cite this version:**

Stefania Montani, Giorgio Leonardi. A Case-Based Approach to Business Process Monitoring. Max Bramer. Third IFIP TC12 International Conference on Artificial Intelligence (AI) / Held as Part of World Computer Congress (WCC), Sep 2010, Brisbane, Australia. Springer, IFIP Advances in Information and Communication Technology, AICT-331, pp.101-110, 2010, Artificial Intelligence in Theory and Practice III. .

HAL Id: hal-01058343
https://hal.inria.fr/hal-01058343

Submitted on 26 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Case-based Approach to Business Process Monitoring

Stefania Montani, Giorgio Leonardi

Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria, Italy

Abstract. The *agile workflow* technology deals with flexible workflow adaptation and overriding, in case of foreseen as well as unforeseen changes and problems in the operating business environment. One key issue that an agile workflow system should address is *Business Process (BP) monitoring*. This consists in properly highlighting and organizing non-compliances and adaptations with respect to the default process schema. Such an activity can be the starting point for other very critical tasks, such as quality assessment and process reengineering.
In this paper, we introduce an automated support to BP monitoring, which exploits the Case-based Reasoning (CBR) methodology. CBR is particularly well suited for managing exceptional situations, and has been proposed in the literature for process change reuse and workflow adaptation support. Our work extends these functionalities by retrieving traces of process execution similar to the current one, which can then be automatically clustered. Retrieval and clustering results can provide support both to end users, in the process instance execution phase, and to process engineers, in (formal) process quality evaluation and long term process schema redefinition. Our approach in practice is illustrated by means of a case study in the field of stroke management.

1 Introduction

Business Process (BP) Management is a set of activities aimed at defining, executing, monitoring and optimizing BP, with the objective of making the business of an enterprise as effective and efficient as possible, and of increasing its economic success. Such activities are highly automated, typically by means of the workflow technology [1, 22].

BP optimization, in particular, may ask the enterprise to be able to flexibly change and adapt the predefined process schema, in response to expected situations (e.g. new laws, reengineering efforts) as well as to unanticipated exceptions and problems in the operating environment (e.g. emergencies) [6].

Agile workflow technology [26] is the technical solution which has been invoked to deal with such adaptation and overriding needs. It can support both ad-hoc changes of individual process instances [18, 25], operated by end users, and modifications at the general process schema level, operated by process engineers - applicable even if the default schema is already in use by some running instances [18, 5].

In order to provide an effective and quick workflow change support, many agile workflow systems share the idea of recalling and reusing concrete *examples of changes* adopted in the past. To this end, Case-based Reasoning (CBR) [2] has been proposed as a natural methodological solution. CBR is a reasoning paradigm that exploits the specific knowledge of previously experienced situations, called *cases*. It operates by *retrieving* and *reusing* - by possibly *revising* - them in order to solve the problem at hand. CBR is particularly well suited for managing exceptional situations, even when they cannot be foreseen or preplanned. As a matter of fact, in the literature cases have often been resorted to in order to describe exceptions, in various domains (see e.g. [20]), and many examples of CBR-based process change reuse and workflow adaptation support have been proposed (see e.g. [12, 10, 25, 14, 15]).

A less explored (see however e.g. [8]), but very critical issue to be addressed in agile workflow systems is the one of BP *monitoring*. This activity consists in highlighting and organizing non-compliances with respect to the default process schema. It can be the starting point for a formal verification of process conformance to proper semantic constraints (see e.g. [11]), or for suggesting long term changes, in front of frequent, similar non-compliances. Providing BP monitoring functionality is non trivial. As a matter of fact, deviations from a default process schema generate process instances, typically stored as traces of actions, that are different from how they were supposed to be. Monitoring BP from traces is particularly hard when no contextual information, which could justify the reasons for deviation, is recorded in the traces themselves. A facility able to intelligently exploit traces of process executions, by retrieving similar ones, and by automatically organizing them, would then be an added value for an agile workflow tool. It could also provide support both to end users, in the process instance execution phase, and to process engineers, in (formal) quality evaluation and/or in long term process schema redefinition.

In this paper, we propose a CBR-based approach to BP monitoring, based on execution traces retrieval. In particular, in this work we focus on the definition of a proper similarity measure, and on its use for calculating trace distance. Our next step will be the implementation of a trace clustering technique, in order to support automatic non-compliances organization. Technical details of the approach are presented in section 2, and a case study on stroke management execution traces is described in section 3. Finally, section 4 addresses some comparisons, discussion and concluding remarks.

2 Case-based BP monitoring

Our framework is meant to support end users in process instance modification by retrieving traces of execution similar to the current one: suggestions on how to modify the default process schema in the current situation may be obtained by analyzing the most similar retrieved examples of change, recorded as traces that share the starting sequence of actions with the current query.

Interestingly, we could also automatically cluster the execution traces stored in the database on the basis of their similarity, and allow process engineers to inspect the obtained clusters, in order to visualize the most frequent changes. Since changes can be an indicator of non-compliance, clustering can be seen as the first step in a process quality evaluation activity, which can be realized by means of formal (e.g. logic-based) verification approaches. Additionally, since changes can also be due to a weak or incomplete initial process schema definition, engineers could exploit clustering results to draw some suggestions on how to redefine process schemas, in order to incorporate the more frequent and significant changes once and for all.

From a technical viewpoint, both the decision support functionalities illustrated above require a preliminary design phase, in which:

1. the case structure is defined;
2. a proper similarity measure, to be used for retrieval and clustering, is identified.

Such issues are the main topic of the paper, and will be illustrated in this section.

Issue 1 is rather straightforward. We have defined a case as a trace of execution of a given process schema. In particular, every trace is a sequence of actions, each one stored with its execution time and duration information.

Issue 2 is more interesting. In the literature, a number of similarity measure definitions for agile workflows exist, that require further information in addition to the workflow structure, such as semantic annotations [21], or conversational knowledge [26, 25]. The approaches are typically context-aware, that is, the contextual information is considered as a part of the similarity assessment of workflows.

Unfortunately, any contextual information, as well as conversational knowledge, is not always available, especially when instances of process execution are recorded as traces of actions. Starting from this observation, a rather simple graph edit distance measure [4] has been proposed and adapted for similarity assessment in workflow change reuse [14, 8]. Our approach moves from the same graph edit distance definition, and properly adapts it in order to work on trace comparison.

Basically, in our approach similarity is modeled through a set of edit operations on traces. Each edit operation performs a modification of the following kinds: (i) *substitute* one action with a different one, (ii) *insert* a new action, or (iii) *delete* an action.

The cost of a *substitution* is not always set to 1, as in the classical edit distance. In fact, as in the weighted edit distance (see e.g. [9]), we define it as a value $\in [0, 1]$ which depends on what action appears in a trace as a substitution of the corresponding action in the other trace. In particular, we organize actions in a *taxonomy*, on the basis of domain knowledge. The more two actions are close in the taxonomy, the less penalty has to be introduced for substitution ([17]; see also [3, 19, 16]). In detail, in our work substitution penalty is set to the *taxonomical distance dt* between the two actions ([17], see Definition 2), i.e.

to the normalized number of arcs on the path between the two actions in the taxonomy.

Insertions do not always cost 1 as well. In fact, an insertion may introduce a degree of indirection in a path otherwise connecting the same pair of actions in the two traces (see figure 1, third case). Our distance definition allows to capture this situation - which is very relevant for BP monitoring. The definition introduces a knowledge-based parametrized weight $\in [0, 1]$ for such insertions, depending on the action type. The final penalty of an insertion which generates an indirection is therefore equal to 1 multiplied by the weight. Naturally, the more insertions are performed, the more indirection is obtained in the path, and the more penalties are added. *Deletions* simply work dually with respect to insertions.

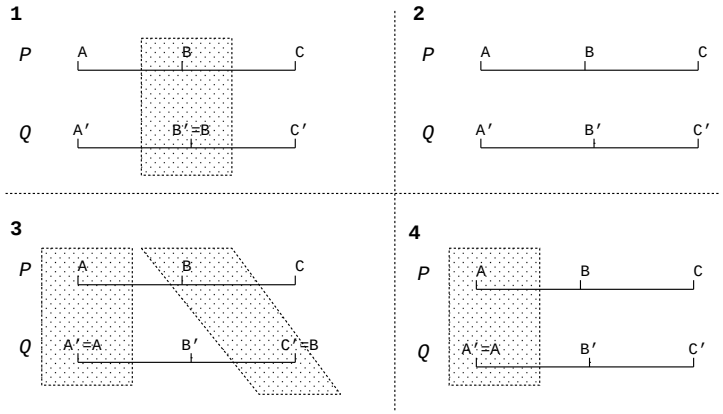

Fig. 1. Comparing the i-th action B in trace P with the j-th action B' in trace Q.

The global distance $d(P, Q)$ between two traces P and Q is finally defined as the total cost of a sequence of edit operations which transform one trace into the other.

Formally, we provide the following definitions:

Definition 1: Trace edit distance. Let P and Q be two traces of actions, and let α and β be two actions. The *trace edit distance* between P and Q is defined as:

$$d(P, Q) = \sum_{i=1}^{k} c(e_i)$$

where $(e_1, \ldots, e_k)$ transforms P into Q, and:

– $c(e_i) = dt(\alpha, \beta)$, if e_i is the substitution of α (appearing in P) with β (appearing in Q), with $dt(\alpha, \beta)$ defined as in Definition 2 below;

– $c(e_i) = 1 * w_\alpha$, if e_i is the insertion (the deletion) of action α in P (from Q), with w_α defined as in Definition 3 below.

Definition 2: Taxonomical distance [**17**]. Let α and β be two actions in the taxonomy t, and let γ be the closest common ancestor of α and β. The *taxonomical distance dt* between α and β is defined as:

$$dt(\alpha, \beta) = \frac{N_1 + N_2}{N_1 + N_2 + 2 * N_3}$$

where N_1 is the number of arcs in the path from α and γ in t, N_2 is the number of arcs in the path from β and γ, and N_3 is the number of arcs in the path from the taxonomy root and γ[1].

Terminology. Two actions α and β are **comparable** if $dt(\alpha, \beta) \leq \tau$, where τ is a threshold to be set on the basis of domain knowledge.

Definition 3: Action weight. Let P and Q be two traces of actions and let α be an action (appearing in Q). The *action weight* w_α of α is defined as:

– $w_\alpha \in [0, 1[$ - to be set on the basis of domain knowledge - if α generates an indirection (see figure 1, third case) in Q with respect to P;

– $w_\alpha = 1$ otherwise (e.g. if P is a substring of Q, and α is an action in the head or tail portion of Q, not matched to any action in P).

Operatively, at a generic iteration of our algorithm we compare the i-th action (say B) in trace P, with the j-th action B' in trace Q. If they are *comparable* (see figure 1, first case), the distance is not increased (or minimally increased, if the actions are not identical), and the next actions are considered. Otherwise, in order to discover insertions that generate indirections, we also compare the $(i-1)$-th action A in P with the $(j-1)$-th action A' in Q. If their are not comparable (see figure 1, second case), no indirection is being generated, and $dt(B, B')$ is added to the distance calculation. On the other hand, if A and A' are comparable, we need to compare B with the $(j+1)$-th action in Q: if they are comparable as well (see figure 1, third case), B' has been inserted to create an indirection on trace Q, and the cost $1 * w_{B'}$ must be added to the distance calculation; otherwise (see figure 1, fourth case), a cost equal to $dt(B, B')$ will be added.

In our approach similar traces retrieval is then performed by classical K-Nearest Neighbor techniques, where the most proper value of k has to be experimentally set according to the specific application domain needs.

As a next step, we will concentrate on the implementation of the clustering facility, which will be based on hierarchical clustering techniques.

[1] Note that, if $dt(\alpha, \beta) > th$, being $th \in [0, 1]$ a proper, domain-dependent threshold, $dt(\alpha, \beta)$ can be forced to 1.

3 The framework in practice: an application to stroke management

Health-Care Organizations (HCO) place strong emphasis on efficiency and effectiveness, to control their health-care performance and expenditures. Therefore, it is important to evaluate existing infrastructures and the services provided. To perform this operation, it is crucial to explore and process the data collected by the HCO systems, organizing them in form of process logs (i.e. traces of execution), which can be seen as the history of what happened in the HCO. Traces can be helpful to gain a clear picture of the actual care process, through the use of BP monitoring techniques [13], like the ones introduced in the previous section. As an example, our framework is currently being tested in the stroke management domain.

A stroke is the rapidly developing loss of brain function(s) due to disturbance in the blood supply to the brain. This can be due to ischemia (lack of glucose and oxygen supply) caused by thrombosis or embolism, or to a hemorrhage. As a result, the affected area of the brain is unable to function, leading to inability to move one or more limbs on one side of the body, inability to understand or formulate speech, or inability to see one side of the visual field. A stroke is a medical emergency and can cause permanent neurological damage, complications, and death. It is the leading cause of adult disability in the United States and Europe. It is the number two cause of death worldwide and may soon become the leading one. The best medical practice [7] requires that stroke patients are treated according to a management protocol, which is basically composed by four steps: (1) emergency management; (2) hospitalization; (3) dismissal; (4) follow up. Each step is in turn composed by a sequence of actions, which must respect some criteria, although inter-patients and inter-hospitals variations are admissible. In particular, in step (1), symptoms onset must be recognized, the patient must be taken to the hospital, and a brain computer-assisted tomography (CAT) must be executed. In step (2), diagnosis has to be finalized, by means of a neurological evaluation and of several additional diagnostic investigations, meant to confirm the stroke hypothesis. Diagnostic procedures may vary, but most patients undergo electrocardiogram (ECG) and chest X-ray. At the same time, administrative patient admission procedures must be fulfilled. Finally, a proper therapy has to be initiated: for instance, up to 90% patients are treated with antiaggregants. Rehabilitation also must be started as soon as possible during hospitalization.

Our system is not fully implemented; therefore it has not entered its validation phase yet. Nevertheless, we can describe some first experiments, to show how the tool works. In our experiments, we used traces collected on real patients, detailing the actions of steps (1) and (2). Our case base is currently composed by more than 300 traces, collected at one of the major Stroke Units in Lombardia Region, Italy.

As an example, we will show the retrieval results related to the query case no. 103101, which presents a rather atypical situation. As a matter of fact, patient no. 103101's CAT results, in step (1), allowed to immediately diagnose a stroke

episode, without the need of additional diagnostic procedures. This fact enabled the responsible physician to start the antiaggregant therapy already in step (1). During step (2), the severity of the patient conditions was investigated by means of additional tests, and a further anticoagulant therapy was started. However, rehabilitation was not started in phase (2) - which is a very anomalous situation - probably due to the patient's very critical health status.

Query case: 103101 → Case retrieved: 109984

Time ↓

Events from case 103101:	Events from case 109984:	
stroke_onset	stroke_onset	
arrival_emergency_ward	arrival_emergency_ward	
CAT	CAT	
antiaggregant		→ indirection
admission	admission	
neurological_evaluation	neurological_evaluation	
ECG	ECG	
coagulative screening	coagulative screening	
anticoagulant	antiaggregant	→ substitution
ric_invest_XR thorax		→ indirection
NMR brain with DWI	NMR brain with DWI	
angio NMR	angio NMR	
trans-thoracic ECG		→ indirection
echo doppler SAT	echo doppler SAT	
transcranic doppler	transcranic doppler	
discharge	discharge	
follow_up	follow_up	

Fig. 2. The best matching retrieved case in our experiment.

Our tool retrieved the 20 Nearest Neighbor cases with respect to case no. 103101, adopting the similarity measure described in the previous section - which took about 1.2 second on an Intel Core 2 Duo T9400, equipped with 4 Gb of DDR2 ram. In particular, the most similar retrieved case, reported in figure 2, is a more standard one with respect to the query case as regards step (1) actions, since diagnosis did not take place during emergency management, but was clarified only during hospitalization, as it usually happens. Therefore, the antiaggregant therapy was not started in step (1). Therapy start in case no. 103101 was then recognized as an indirection in an otherwise identical sequence of actions (see figure 1, third case). Analogously, a couple of additional diagnostic procedures took place in case no. 103101, step (2), determining indirections with respect to the retrieved case. On the other hand, antiaggregant therapy was started in the retrieved case after all the diagnostic tests were completed. This action is a substitution of the anticoagulant therapy in case no. 103101, and is also *comparable* to it, having set $\tau = 0.2$: actually, the two drugs have a very similar effect, and consequently the two therapeutic actions are very close in the domain taxonomy. Interestingly, rehabilitation was also missing in the retrieved case: this means that our tool was able to correctly retrieve one of the few cases in the case base matching the query case with respect to this very atypical feature.

The physicians working with us judged our first experimental results as very reasonable. Encouraged by these outcomes, in the next future, we plan to extensively test the performances of our tool on additional real cases, after having worked at the implementation and validation of the clustering procedure.

4 Discussion and conclusions

In this work, we have described a CBR-based approach to BP monitoring. In particular, we have defined a proper case structure and a new similarity measure, that are exploited to retrieve traces of execution similar to the current one. In the next future, the similarity measure will also be applied to cluster the traces available in the database. Such functionalities will help end users who need to adapt a process instance to some unforeseen situation, by retrieving changes applied in the past to other instances of the same process. Moreover, process engineers will take advantage of the retrieval and clustering results for identifying the most frequent changes to the same process schema. Such changes can be an index of non conformance of process executions with respect to proper constraints, but can also be a suggestion for properly revising an incorrect or obsolete process schema definition.

From the technical viewpoint, as observed in section 2, the graph edit distance measure, that we have adapted in this work, was originally resorted to in [14]. However, with respect to that approach, by focusing just on traces of execution we did not need to consider extensions to the similarity measure able to deal with control flow elements (such as alternatives and iterations). As a matter of fact, traces are always linear, i.e. they just admit the sequence control flow element. For the same reason, we did not insert any penalty for pairs of arcs in different traces with a different input or output action, which is already accounted for by the comparison between pairs of actions themselves. On the other hand, when focusing on linear traces our approach is more general and flexible than the one in [14]. As a matter of fact, we resort to taxonomical knowledge for comparing pairs of actions, so that two different actions do not always have a zero similarity. Moreover, we are able to recognize an indirect path from two actions, and to properly weight the degree of indirection in a parametrized way.

Explicitly comparing pairs of arcs in different traces will be needed if considering temporal distances between actions and actions durations, which we will explore as a future work. In particular, in the similarity measure definition, we plan to calculate a penalty for arcs connecting the same pair of actions in both the traces at hand, but introducing a different temporal delay between them. The penalty will be proportional to the difference between the two delays. This calculation will also be adapted to the case in which the same pair of actions is directly connected in one trace, and indirectly in the other trace. In this case, the difference in the arc durations will also need to properly take into account the degree of indirection. Some effort in the direction of considering time in trace comparison has been proposed in [8]. The similarity function in [8], however, does not exploit action duration, and does not rely on taxonomical information

about actions, as we do. The authors also do not distinguish between direct and indirect paths connecting the same actions, so that our approach, once extended to deal with temporal information, will potentially be more flexible.

As a final consideration, in the future we will also explore the possibility of incorporating our work as a plug-in in the ProM tool [24], which is an open source framework for process mining. Process mining [23] describes a family of a posteriori analysis techniques exploiting the information recorded in traces of actions. This process information can be used to *discover* the underlying process schema, when no a priori model is available. Once the process schema is obtained, the incorporation of our similarity measure and of the clustering facility could support an analysis of deviations from the process schema itself, which can be the input for a (formal) compliance verification. Moreover, in cooperation with other performances analysis plug-ins already embedded in ProM, our work could support a principled reengineering activity.

References

1. Workflow management coalition, *http : //www.wfmc.org/wfmc − publications.html*, last accessed on october 6th 2009.
2. A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations and systems approaches. *AI Communications*, 7:39–59, 1994.
3. R. Bergmann and A. Stahl. Similarity measures for object-oriented case representations. In B. Smyth and P. Cunningham, editors, *Proc. European Workshop on Case-Based Reasoning (EWCBR) 1998, Lecture Notes in Artificial Intelligence 1488*. Springer-Verlag, Berlin, 1998.
4. H. Bunke and B.T. Messmer. Similarity measures for structured representations. In *Proc. of the European Workshop on Case-based Reasoning (EWCBR), LNCS 837* , pages 106–118, Kaiserslautern, 1993.
5. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolutions. *Data and Knowledge Engineering*, 24:211–238, 1998.
6. P. Heimann, G. Joeris, C. Krapp, and B. Westfechtel. Dynamite: dynamic task nets for software process management. In *Proceedings International Conference of Software Engineering*, pages 331–341, Berlin, 1996.
7. D. Inzitari and G. Carlucci. Italian stroke guidelines (spread): evidence and clinical practice. *Neurological Sciences*, 27:s225–s227, 2006.
8. S. Kapetanakis, M. Petridis, J. Ma, and L. Bacon. Workflow monitoring and diagnosis using case based reasoning on incomplete temporal log data. In *Proc. Uncertainty, knowledge discovery and similarity in Case Based Reasoning Workshop, International Conference on Case Based Reasoning (ICCBR)*, pages 171–180, 2009.
9. S. Kurtz. Approximate string seraching under weighted edit distance. In *Proc. 3rd South American Workshop on String Processing*, Recife, 1996. Carlton University Press.
10. Z. Luo, A. Sheth, K. Kochut, and J. Miller. Exception handling in workflow systems. *Applied Intelligence*, 13:125–147, 2000.
11. L. Thao Ly, S. Rinderle, and P. Dadam. Integration and verification of semantic constraints in adaptive process management systems. *Data & Knowledge Engineering*, 64(1):3–23, 2008.

12. T. Madhusudan, J.L. Zhao, and B. Marshall. A case-based reasoning framework for workflow model management. *Data and Knowledge Engineering*, 50:87–115, 2004.
13. R. Mans, H. Schonenberg, G. Leonardi, S. Panzarasa, A. Cavallini, S. Quaglini, and W. vanderAalst. Process mining techniques: an application to stroke care. In S. Andersen, G. O. Klein, S. Schulz, and J. Aarts, editors, *Proc. MIE, Studies in Health Technology and Informatics 136*, pages 573–578. IOS Press, 2008.
14. M. Minor, A. Tartakovski, D. Schmalen, and R. Bergmann. Agile workflow technology and case-based change reuse for long-term processes. *International Journal of Intelligent Information Technologies*, 4(1):80–98, 2008.
15. S. Montani. Prototype-based management of business process exception cases. *Applied Intelligence (published online in February 2009- DOI 10.1007/s10489-009-0165-z)*.
16. R. Page and M. Holmes. *Molecular Evolution: A Phylogenetic Approach*. Wiley, 1998.
17. M. Palmer and Z. Wu. Verb Semantics for English-Chinese Translation. *Machine Translation*, 10:59–92, 1995.
18. M. Reichtert, S. Rinderle, and P. Dadam. Adept workflow management system: Flexible support for enterprise-wide business processes. In W. M. P. vanderAalst, editor, *Proc. of International Conference on Business Process Management (BPM) LNAI 2678*, pages 370–379. Springer, Berlin, 2003.
19. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. IJCAI*, pages 448–453, 1995.
20. J. Surma and K. Vanhoof. Integration rules and cases for the classification task. In M. Veloso and A. Aamodt, editors, *Proc. 1st Int. Conference on Case-Based Reasoning*, volume 1010 of *Lecture Notes in Computer Science*, pages 325–334, Sesimbra, Portugal, October 1995. Springer.
21. L. van Elst, F.R. Aschoff, A. Berbardi, H. Maus, and S. Schwarz. Weakly-structured workflows for knowledge-intensive tasks: An experimental evaluation. In *Proc. of 12th IEEE International Workshops on Enabling Technologies (WETICE), Infrastructure for Collaborative Enterprises*, pages 340–345. IEEE Computer Society, Los Alamitos, 2003.
22. W. VanderAalst, A. terHofstede, and M. Weske. Business process management: a survey. In *Proc. Int. Conference on Business Process Management (BPM), LNCS 2678*, pages 1–12. Springer, 2003.
23. W. VanderAalst, B. vanDongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workflow mining: a survey of issues and approaches. *Data and Knowledge Engineering*, 47:237–267, 2003.
24. B. vanDongen, A. AlvesDeMedeiros, H. Verbeek, A. Weijters, and W. VanderAalst. The proM framework: a new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Knowledge Mangement and its Integrative Elements*, pages 444–454. Springer, 2005.
25. B. Weber, M. Reichert, and W. Wild. Case-based maintenance for CCBR-based process evolution. In T. Roth-Berghofer, M. Goker, and H. Altay Guvenir, editors, *Proc. European Conference on Case Based Reasoning (ECCBR) 2006, LNAI 4106*, pages 106–120. Springer, Berlin, 2006.
26. B. Weber and W. Wild. Towards the agile management of business processes. In K. D. Althoff, A. Dengel, R. Bergmann, M. Nick, and T. Roth-Berghofer, editors, *Professional knowledge management WM 2005, LNCS 3782*, pages 409–419, Washington DC, 2005. Springer, Berlin.