A Time-Series Similarity Measure for Case-Based Deviation Management to Support Flexible Workflow Execution

Erik Schake[1], Lisa Grumbach[1,2], Ralph Bergmann[1]

[1] Business Information Systems II, University of Trier, 54286 Trier, Germany
s4ekscha,bergmann@uni-trier.de
http://www.wi2.uni-trier.de
[2] Trier University of Applied Sciences, 55761 Birkenfeld, Germany
l.grumbach@umwelt-campus.de

Abstract Our objective is to develop an approach based on case-based reasoning that detects and handles unforeseen deviations that occur in flexible workflow execution. With a case-based approach we aim at supporting the continuation of a deviant workflow execution by utilizing successfully completed processes, where similar deviations emerged. As a first step, this work introduces a novel similarity measure based on time sequence similarity that is able to compare running and completed workflow instances. We implemented and evaluated our approach in the ProCAKE framework. The proposed similarity measure achieves promising results considering runtime and similarity assessment.

Keywords: Knowledge Management · Process Management · Case-Based Reasoning · Flexibility by Deviation

1 Introduction

Digitalization is advancing in today's business. Small and medium-sized enterprises (SMEs) appear to be lagging behind compared to large companies. Process-aware information systems [1] are well established for standardized processes, but they lack support for flexibility, which is often required in SMEs and may lead to competitive advantages. Additionally, in SMEs there are often few experts who are responsible for certain processes. The knowledge about how things are done is implicit, not stored digitally and information is often only shared orally. These experts usually deviate and perform processes due to their expertise and experiences without losing control or missing the objective, but rather optimizing the process [18]. Tracing these processes automatically and using them for process control may simplify the transfer and preservation of "best practices". This in turn may lead to an increase of efficiency and enhanced assistance possibilities especially for inexperienced users. Furthermore, bypassing the system and thus a loss of knowledge and transparency is prevented.

A main characteristic of an adequate approach for SMEs is to allow unexpected deviations and support unforeseen changing circumstances. Thereby, the

key challenge is to determine how to continue with the workflow, while still achieving a successful completion. Existing approaches either require a manual handling of upcoming deviations or an effortful modeling of expert knowledge, resulting in a large effort for knowledge acquisition and leading to high costs for maintenance [2,5,7,8,14,18,20]. To avoid these disadvantages we propose to utilize past experiences, more specifically, successfully completed processes that contain similar deviations. Applying case-based reasoning (CBR) seems to be promising, as previous made experiences are exploited for adapting to unknown situations. Our objective is to develop an approach based on CBR that handles deviations that occur in flexible workflow execution, but still guides the user by recommending adequate work items. As a first, but important step into this direction, this paper presents a novel similarity measure based on time sequence similarity that is able to compare running and completed workflow instances.

In the remainder of the paper we introduce the notions of workflow flexibility and deviation management as well as our pursued approach in section 2. Related similarity measures are sketched in section 3. Our proposed similarity measure is presented in section 4, while an evaluation of our concept is outlined in section 5. We conclude with a summary and an insight into our future work.

2 Deviation Management for Flexible Workflows

In this section our previous work on workflow flexibility and its limitations will be presented, before we introduce the idea of a case-based deviation management.

2.1 Workflow Flexibility

Flexible Workflows have been focused in research for more than a decade [17]. Four different flexibility principles are distinguished [17]. *Flexibility by Design, Change* and *Underspecification* either require an entire awareness about all possible upcoming situations at design-time in order to manually model all possible alternatives, or a remodeling of the workflow is necessary at run-time. *Flexibility by Deviation* in contrast "is the ability for a process instance to deviate at run-time from the execution path prescribed by the original process without altering its process definition" [17]. Hence, the user is able to execute tasks that are not suggested as next activity, as the work list is only seen as a guideline. Thus, single instances may not fit to the process model. We therefore explicitly distinguish between modeled workflow, denoted as *de jure* and executed workflow, called *de facto* [1]. This, however, raises the problem of deciding how to continue with the deviating workflow to achieve a successful completion. To solve this problem we developed a workflow engine that facilitates flexibility by deviation.

2.2 Constraint-Based Workflow Engine

During the SEMAFLEX [11] and the SEMANAS [12] project we developed a flexible workflow engine based on constraints. We presented an approach [9],

that allows flexible deviations from prescribed workflows, but still maintains control and recommends valid work items to some extent.

The proposed method is applied to imperatively modeled block-oriented workflows, which additionally comprise semantic information. Block-oriented workflows are constructed through a single root block element, which in turn consists of a single task node, a sequence of two block elements or a control-flow block. Start and end of control-flow blocks are clearly defined through control-flow nodes, of which three different types exist ($type \in \{+, \times, \circ\}$). These workflows are represented as semantic graphs (denoted as *NESTGraph*), where nodes and edges additionally link to semantic descriptions [3]. Fig. 1 shows an excerpt of the workflow, which we further on used for evaluating our approach.

This workflow consists of six task nodes (blue rectangles), three data nodes (ovals), two control-flow nodes (red rhombuses), which represent an exclusive block ("×") and one workflow node (blue rhombus). Additionally, the edges denote either control-flow (solid black lines), data-flow (solid grey lines, input/output relation) or part-of edges (dashed lines). Furthermore each node is associated with a semantic description (grey rectangles), which contains additional information, e.g. data node d_3 is assigned a *name* "volume" with a *value* of "1000000".

In our approach [9] we transform these imperative workflow models into declarative constraints, which indicate temporal dependencies between task activations, to be able to determine task activations and thus possible executions in a specific state of the workflow. A constraint satisfaction problem (CSP) is constructed on the basis of these generated constraints and logged task enactments. A solution of the CSP is searched for, which represents a valid enactment sequence of all tasks. Thus, with a CSP solution we are able to recommend which task to execute next. As we only trace task executions, deviations from this prescribed work list are permitted. Ideally, the user decides to do something else if s/he is sure that this is more appropriate in the specific situation than what the de jure workflow proposes. In such a case, constraints are violated but they can be retracted easily and fast at run-time in order to restore consistency. By regarding the remaining constraints, valid solutions can be computed and thus, the workflow engine is re-enabled to recommend further work items to complete the workflow.

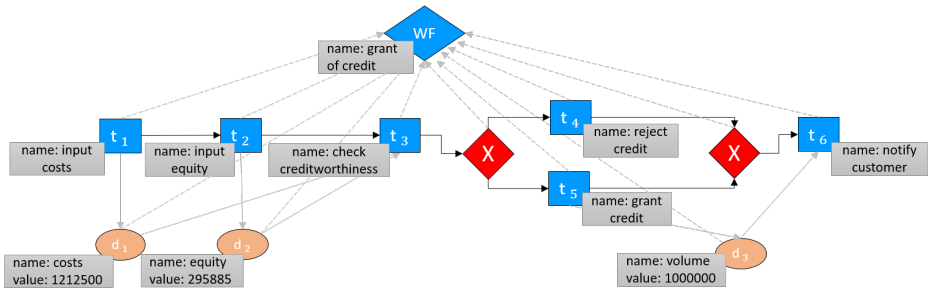

Fig. 1. Exemplary Block-Oriented Semantic Workflow Graph

However, a categorization of deviations and their cause is not considered until now. Several strategies for resolving inconsistencies and their limitations have been presented in [10]. Consider the example in Fig. 2, where the left upper row shows the initial situation and the second row represents one step further with an additional executed task. Rectangles indicate tasks and edges represent the execution order. Blue filled nodes were already executed, green ones are currently recommended and colourless nodes are not activated yet. In the left lower row there is a task executed, which is not in a valid order, thus a deviation occurred (marked in orange). The cause of the violation is not explicit without semantic knowledge and there may be different reasons and consequences. Tasks should be skipped either due to becoming obsolete or due to being executed but without notice of the system (cf. a) in Fig. 2). Alternatively, task order should be adopted due to unknown reasons according to b) in Fig. 2.

This scenario is a simple type of deviation, but nevertheless requires a decision about which resolving strategy to apply. Besides, deviations might lead to additional deviations or might be more complex, which makes it impossible to determine similar strategies to handle all possible scenarios. Several approaches to deviation management exist, that utilize rules or knowledge which is specified beforehand [2,5,7,8,14,18]. CBRFlow [20] is a system that exploits conversational CBR to react to unexpected deviations. Business rules are used to model workflow run-time changes and may then be recommended in similar situations. To handle deviations, these approaches either require user interaction or domain knowledge resulting in a significant acquisition effort and limited usability.

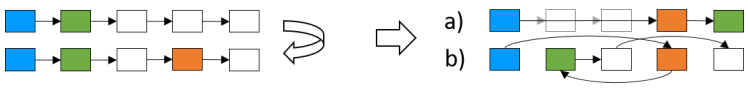

Fig. 2. Example Sequential Workflow with Deviation and Different Interpretation

2.3 Case-Based Deviation Management

Our aim is to construct methods that can be applied in an automated manner and support the user without the need of manual intervention or prior knowledge acquisition. Therefore we focus on CBR as technique to overcome the previously mentioned disadvantages. With a case-based approach we expect to support workflow continuation after a deviation by recommending adequate work items on the basis of similar cases. The overall approach is sketched in Fig. 3.

Case As cases we regard all de facto workflows and their corresponding de jure workflow. The de jure workflow is block-oriented and the de facto workflow is a simple sequence of tasks, which were traced at run-time. This includes workflows with deviations as well as without deviations, as both cases might be useful for task recommendation.

Query A running workflow instance, i.e a de facto workflow which is not completed yet, will be used as the query. This instance is a subsequence of a completed de facto workflow and contains at least one deviation (see orange-coloured node in Fig. 3) concerning its de jure workflow that occurred at the time of the request. The associated de jure workflow is also available.

Deviation Management All phases of the CBR cycle are important for a holistic view of deviation management and lead to a self-learning system.

Retrieve With an adequate similarity measure we aim at searching for similar de facto workflows whose subsequences match the current instance, containing a similar deviation compared to the query.

Reuse This most similar case or even several similar cases might then be used to recommend tasks that were successive to the subsequence ending with the deviation (see green-coloured nodes in Fig. 3). In some circumstances, a simple transfer of the solution will not be reasonable, but rather an adaptation of the recommended remaining workflow part might be necessary.

Revise As tasks are only recommended, the user is still able to execute a task, which was not part of the solution resulting from the reuse step. Thus, by continuing the query workflow, the solution might be revised.

Retain When the query workflow has successfully terminated, its de facto workflow, containing the actual execution, can be integrated in the case base.

In this paper we focus on the retrieval phase and present a similarity measure that is necessary for this case-based approach. The de facto workflows that are compared are perceived as time series that consist of tasks with additional semantic information. Therefore an adequate similarity measure based on time sequence similarity is searched for.

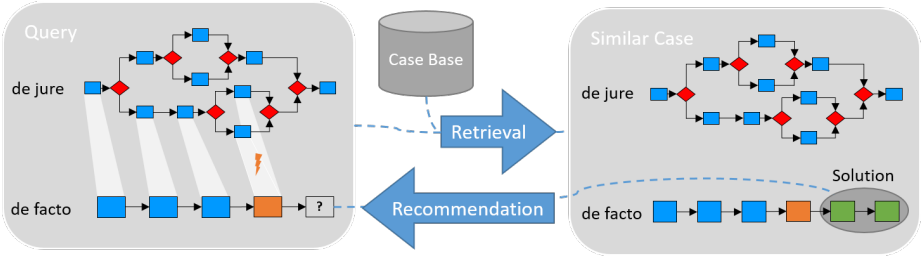

Fig. 3. Case-Based Approach for Handling Deviations

3 Similarity Measures for Time Series

The pursued adoption of a case-based task recommender requires a suitable similarity measure to empower the retrieval phase. In our domain, it must compare de facto workflows. As they comprise a sequentially ordered series of tasks,

they are comparable with time series and sequence similarity algorithms can be applied. Two basic approaches can be identified in literature. Either the series are transformed into a representation format, such as an n-dimensional vector, which then is compared, or the algorithm finds an alignment between both series which is then assigned a similarity score by aggregating local element similarities. Three promising approaches will be introduced in more detail in this section.

3.1 Dynamic Time Warping

Dynamic time warping (DTW) was originally developed to properly align distorted time series data collected from voice recordings [4,16]. Contrary to simple euclidean distance, which only evaluates distances between elements with the same timestamps, DTW allows elements to be warped onto elements with different timestamps. It therefore is resistant to compression and stretching.

Natively, DTW was defined using distance functions, however, since distances can be losslessly converted into similarity scores, we will present the algorithm using a similarity function. Given two sequences v and w with elements v_i $(i = 1, \ldots, m)$ and w_j $(j = 1, \ldots, n)$ to be compared, their leading elements are first set to a null value: $v_0 = w_0 = -$. The scoring matrix H is then constructed in the following way. Its first row and column are initialized to zero: $H_{0,j} = H_{i,0} = 0 \; for \; j = 0, \ldots, n, i = 0, \ldots, m$. Every other cell's value can be calculated by the recursive function $H_{i,j}$:

$$H_{i,j} = \max \begin{cases} H_{i,j-1} + sim(v_i, w_j), & \text{step horizontally} \\ H_{i-1,j-1} + 2 * sim(v_i, w_j), & \text{step diagonally} \\ H_{i-1,j} + sim(v_i, w_j), & \text{step vertically} \\ 0 \end{cases}$$

A cell (k, l) in the matrix is interpreted to hold the similarity score between the subsequences $v_1 v_2 \ldots v_k$ and $w_1 w_2 \ldots w_l$. In the fully constructed matrix, we must therefore select the cell with the maximum score - representing the final similarity value between best matching subsequences - and backtrack in the inverted direction of each cell's original calculation. Every cell (i, j) contained in the path through the matrix represents a warp from element v_i onto w_j.

3.2 Smith-Waterman-Algorithm

Another adoption of finding subsequence alignments is the Smith-Waterman-Algorithm (SWA) [19]. Inspired by the Levenshtein distance, the alignment is found by successively either matching, inserting or deleting elements from one series with regard to the other.

Initialization of the scoring matrix is conducted equivalently to DTW, however, the scoring function itself differentiates the two approaches. Horizontal and vertical steps in the matrix represent either deletion or insertion (indels) of an element, while diagonal steps are interpreted as aligning two elements. Therefore, a penalty scheme must be introduced that assigns a negative score to any

indel operation (cf. *penaltyInsertion* and *penaltyDeletion*). The scoring matrix is constructed in the following way:

$$H_{i,j} = \max \begin{cases} H_{i,j-1} + penaltyInsertion(b_j), & \text{insertion} \\ H_{i-1,j-1} + sim(a_i, b_j), & \text{match/mismatch} \\ H_{i-1,j} + penaltyDeletion(a_i), & \text{deletion} \\ 0 \end{cases}$$

Again, the highest score in the matrix represents the obtained similarity value. However, the alignment path must be interpreted differently. In contrast to DTW, only cells that originated from diagonal steps represent an alignment of those elements. Horizontally and vertically created cells represent insertions and deletions with regard to sequence v.

SWA is very noise-resistant, since outliers can simply be deleted in the sequences. Both DTW and SWA not only find a similarity score, but also an alignment which can be exploited in the reuse phase of CBR. Furthermore, both algorithms have time complexities of $O(n^2)$. The main difference is the interpretation of the found alignment path. SWA considers sequences to be equal if one is a subsequence of the other, whereas DTW considers sequences as equal if they are simply stretched or compressed versions of each other.

Zarka et al. [22] adopt SWA for an approach called trace-base reasoning. They utilize this method in order to find suggestions for workflow continuation in a mobile video editing suite by also incorporating a CBR approach. However, the context of deviations is not considered by the authors.

3.3 Weighted Vector Similarity

Contrary to the other two approaches which find series alignments by directly comparing the sequences, a similarity measure introduced by Gundersen [13] transforms the sequences into a vector representation. It is based on the assumption that recent tasks are most important when comparing processes.

Originally, this measure was defined on event sequences. Therefore we define the end of each task to be an event. Given n events, the sequence A is then transformed into a vector $V_A = (w_1, w_2, ..., w_n)$ with weights w_i depicting each event's importance in A based on its temporal position pos_i, calculated as follows:

$$w_i = \frac{1}{2^{\frac{end - pos_i}{h}}} \tag{1}$$

The halving distance h specifies the distance to the end where the events' importance shall be one half. Therefore, with this formula, the weight of an event at the end of the sequence will be one, the weight of an event at $end - h$ will be one half, and the weight of an event prior to $end - h$ will be between zero and one half. Those weighted vector representations can now be compared by utilizing measures such as cosine similarity and relative component fraction.

The main strength of this measure is its efficiency having a computational complexity of only $O(n)$. Major drawbacks however include that it is not able to consider semantic similarity and can only compare entire sequences.

4 Similarity Measure for Deviating Workflows

In this section we first sketch specific properties of flexibility by deviation that are important for determining similarity. A pre-processing of the cases is described subsequently. Finally, the developed algorithm for similarity assessment which is based on the previously presented approaches is introduced in detail.

4.1 Characteristics of Similarity for Deviating Workflows

The similarity measure has to deal with a subset relation between query and case, as the de facto workflow of the query has not terminated yet, whereas the de facto workflows of the cases are complete. For the comparison of these instances we propose to use a sequence similarity measure.

Another important aspect of the pursued similarity is that under certain conditions some mismatches of subsequences of case and query should have low or rather no influence on the similarity score. This refers to two aspects. On the one hand, we assume that differences with a greater temporal distance to the occurrence of the deviation should have a lower impact on the similarity. Therefore a weighting function will be included in the final similarity measure.

On the other hand, we incorporate a property of case and query which we call model-consistency. Any de facto workflow or any subsequence of a de facto workflow without a deviation, but conforming to the de jure workflow is denoted as model-consistent. In some cases, if case and query contain the same deviation but additionally differ in some other prior subsequences, which are both model-consistent, this difference might be ignored, if it is not related to the deviation.

4.2 Pre-Processing of the Case Base

The pre-processing algorithm is used to determine irrelevant workflow blocks in case and query which are excluded from similarity assessment by deleting these subsequences prior to comparison. This results in reduced cases and thus, a potentially less complex and faster similarity assessment. Furthermore, pre-processing can be done beforehand such that retrieval time is not affected but rather enhanced. As prerequisite, case and query need to be based on the same de jure workflow such that those subsequences might be deleted that belong to corresponding parts. This is necessary for a reasonable and meaningful comparison of reduced case and reduced query. As our overall approach is limited to block-oriented workflows we exploit the block structuring for this purpose. The pre-processing algorithm prepares the case base with further information which is used when a query arises. An example is presented in Fig. 4.

Step 1 First, the nodes of the de jure workflow of each case are labeled. Every control-flow node pair, i.e. split and join node, is assigned a unique id, $bid \in blockids \subset \mathbb{N}$, and each task node t is labeled with certain information. This includes the operator of the nearest preceding control-flow node, $op^t \in \{+, \times, \circ, \backslash\}$ that can be any type of control-flow node or "\", which

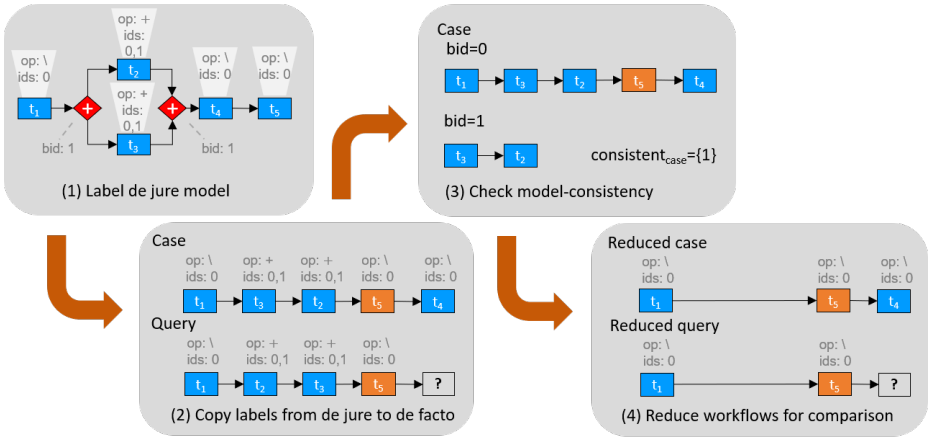

Fig. 4. Exemplary Pre-Processing

denotes the highest level where tasks are not nested in a control-flow block. Furthermore, a set of ids, $ids^t \in \mathcal{P}(blockids)$, of all preceding control-flow split nodes that have not been joined until the currently regarded task node, is added to denote the position concerning levels of nested workflow blocks.

Step 2 These labels are subsequently transferred to all corresponding de facto workflows of the cases. This can be done easily, since each task node in the de facto contains a reference id of the specific task of the de jure workflow.

Step 3 Model-consistency can be checked for each de facto workflow. For each $bid \in blockids$ the set of task ids $T_B = \{t | bid \in ids^t\}$ is extracted from the de facto workflow, preserving their sequential order, and validated by the constraint-based workflow engine. Every *bid* of consistent control-flow blocks is stored in a set $consistent \subseteq blockids$. In the example in Fig. 4 this is shown for the case workflow. The sequence for $bid = 0$ is not model-consistent, as a deviation is included (cf. position of t_5), whereas the subsequence for $bid = 1$ is model-consistent and this *bid* is thus added to *consistent*. The same procedure will be applied to the de facto workflow of the query.

Step 4 Prior to similarity assessment, query and cases can be reduced. If any *bid* coincides in the set *consistent* of both query and case, every task t where $bid \in ids^t$ can be deleted from the de facto workflows. Hence, these tasks do not affect the similarity score. In the example in Fig. 4 $bid = 1 \in consistent_{case} = consistent_{query}$ and thus, t_2 and t_3 are deleted from both de facto workflows. In this case the omission of tasks of the parallel workflow block is reasonable, since the order of the tasks may only differ due to the sequential tracing of terminated tasks, whereas in reality in both cases tasks t_2 and t_3 might be executed in parallel. As the deviation occurs not until after the parallel block, the difference in the preceding part might have no impact on the deviation and the right choice to continue. Therefore this partial mismatch is simply ignored when assessing the similarity.

4.3 Similarity Assessment

As similarity measure we propose a combined approach of the previously mentioned techniques to take advantage of each algorithm's specific strengths. We adopt a dynamic programming approach, which includes either SWA or DTW, and extend it by the idea of a weighting function considering temporal ordering. Furthermore local task similarities include data-flow and semantic information. The similarity assessment can either be applied to reduced cases or non-reduced ones. We compare a de facto workflow of the query with m tasks to a de facto workflow of a case with n tasks.

Local Task Similarity We define local task similarity sim_t of a query task q_i^t and a case task c_j^t as follows:

$$sim_t(q_i^t, c_j^t) = \frac{l_t * sim_N(q_i^t, c_j^t) + l_i * sim_{df}(in_{q_i^t}, in_{c_j^t}) + l_o * sim_{df}(out_{q_i^t}, out_{c_j^t})}{l_t + l_i + l_o} \tag{2}$$

This value is composed of task, input and output similarity each assigned a respective weight l_t, l_i, l_o. sim_N represents an aggregated similarity score of the semantic description of the task nodes. sim_{df} describes the similarity of input and output data nodes respectively. The set of input data nodes of the query $in_{q_i^t}$ is compared to the set of input data nodes of the case $in_{c_j^t}$ by finding the best possible mapping. The same is applied to the sets of output data nodes $(out_{q_i^t}, out_{c_j^t})$. These three local similarities are finally summed up and normalized by the sum of all weights.

Scoring Matrix The scoring matrix that combines either SWA or DTW with a temporal distance factor and the presented local similarity is defined as follows:

$$H_{i,j} = max \begin{cases} H_{i-1,j-1} & + wTemp_i * ALG_{i,j}(diagonal), \\ H_{i-1,j} & + wTemp_i * ALG_{i,j}(vertical), \\ H_{i,j-1} & + wTemp_i * ALG_{i,j}(horizontal), \\ 0 & \end{cases} \tag{3}$$

As starting point the first row and column are initialized to 0, $\forall j : 0, \ldots, n : H_{0,j} = 0$ and $\forall i : 0, \ldots, m : H_{i,0} = 0$. The stepping function $ALG_{i,j}$ differs depending on which algorithm is used. Equations 4 and 5 show both variants.

$$SWA_{i,j}(x) := \begin{cases} sim_t(q_i^t, c_j^t) & \text{if } x = \text{'diagonal'}, \\ penaltyDeletion(q_i^t) & \text{if } x = \text{'horizontal'}, \\ penaltyInsertion(c_j^t) & \text{if } x = \text{'vertical'}, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$DTW_{i,j}(x) := \begin{cases} 2 * sim_t(q_i^t, c_j^t) & \text{if } x = \text{'diagonal'}, \\ sim_t(q_i^t, c_j^t) & \text{if } x = \text{'horizontal'}, \\ sim_t(q_i^t, c_j^t) & \text{if } x = \text{'vertical'}, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This score is further multiplied by the weight depending on temporal distance. Therefore, equation 1 will be adapted as follows. Here, m is the length of the query, as this serves as baseline for the similarity calculation. The halving time as a parameter is assigned a value $h \in [0, m]$.

$$wTemp_i = \frac{1}{2^{\frac{m-i}{h}}} \tag{6}$$

Similarity Score The non-normalized overall similarity score sim_{raw} can be obtained by searching for the largest value in the last row of the matrix. By limiting this search to $H_{m,j}$ for $0 < j \leq n$, it is ensured that the found alignment always ends with the last task of the query, thus, $sim_{raw} = H_{m,k}$ with $H_{m,k} > H_{m,j}$ for all $j \in \{0, \ldots, n\} \setminus \{k\}$.

To normalize the obtained score, it must be divided by the maximum possible score with regard to the query. In SWA local positive similarity values are only aggregated when stepping diagonally in the matrix, hence the maximum possible score can be calculated by finding the amount of diagonal steps in the alignment. In contrast DTW adds a positive value to the score when stepping in either direction. Let $align = \{(0,0), \ldots, (m,k)\}$ denote all cells from the alignment path. Let $diag = \{(i_0, j_0), \ldots, (i_p, j_p)\} \subseteq align$ denote all cells that originated from a diagonal step and let $other = align \setminus diag$ denote all other assignments. Normalized similarity is then calculated as follows:

$$sim_{defacto}^{SWA}(q, c) = \frac{sim_{raw}}{\sum_{(i,j) \in diag} wTemp_i} \tag{7}$$

$$sim_{defacto}^{DTW}(q, c) = \frac{sim_{raw}}{\sum_{(i,j) \in diag} 2 * wTemp_i + \sum_{(i,j) \in other} wTemp_i} \tag{8}$$

5 Evaluation

We evaluate our method by comparing it to a baseline approach. Bergmann and Gil, who developed the NESTGraph formalism, presented a graph-based similarity measure for modeled workflows [3]. They utilize heuristic A* search in order to find a mapping m that assigns nodes and edges from graph A to nodes and edges of graph B. Given such a legal mapping m, local similarities between the mapped elements are calculated and aggregated into a global similarity score.

While many related approaches have been researched (c.f. [6,15,21]) that all utilize workflow mappings, the method presented by Bergmann and Gil constitutes an exception to these mostly infeasible proposals for searching the mappings. Therefore, their A* approach will serve as the baseline similarity algorithm. It must be noted that in contrast to our approach, A* additionally searches for mappings between the two graphs' edges which accounts for approximately halve of the runtimes. Furthermore, due to resource restrictions, the search queue is limited to 10000 entries which may hinder the algorithm from finding the overall best mapping.

We want to show that our approach outperforms A* both regarding the obtained ranking when a retrieval is performed, and regarding overall retrieval times. To validate this, three hypotheses are considered:

- **(H1)** For every query instance, our approach will retrieve the *best case* from the case base at a higher or equal rank compared to the A* approach.
- **(H2)** Using pre-processing, rankings will be as high or higher than without using pre-processing.
- **(H3)** The overall retrieval times using our approach are significantly lower than those of A*.

As prerequisite, we must clarify how we identify the best case from the case base for each query. Through communication with field experts, we modeled a realistic de jure workflow, based on a credit grant process, consisting of 31 task nodes, 18 data nodes and 4 control-flow blocks. As queries, non-terminated workflow instances were created which contain at least one deviation at the end with regard to the process model. Based on a pool of realistic deviations, randomly chosen ones were further included in the queries.

Then, a matching *best case* was constructed for every query through insertions of tasks leading to the termination of the query workflow. Additionally, not more than three deviations from the pool which are not included in the query have been added to this case. However, model-consistent passages may have been replaced with different but still model-consistent passages in order to evaluate pre-processing. Those generated instances were then inserted into the case base. Thereby, each query is associated with exactly one *best case*. The case base was filled up with additional 27 cases, each of which contains at least as much deviations with regard to every query as their respective *best cases*.

We performed a simple linear retrieval using the A* approach and our approaches with and without pre-processing for every query, respectively. Each retrieval produces a ranking of the cases based on their respective similarities to the query. The rank of each query's *best case* is shown in Fig 5. In 50% of the retrievals, A* performs as well as at least two configurations of our approaches, however, in the other halve, it produces significantly poorer results. In no case does it outperform all of our approaches regarding the ranking. Thus, hypothesis H1 is approved. Most of the time it does not matter whether pre-processing was applied or not. Only in queries 6 and 14, the utilization of pre-processing improves the rankings significantly. There are also queries (5,11,13) where it may even impair the results. However, with the exception of query 11, this is due

to already poor results of DTW when no pre-processing is being applied. Thus, hypothesis H2, only holds partially.

The retrieval times of the approaches differ by several orders of magnitude even after correction for edge mapping. In our sample, mean retrieval time using SWA or DTW was 45 ms in contrast to A* taking 13437 ms on average[3]. Pre-processing can reduce runtimes even further. This shows a significant advantage of our approach and verifies hypothesis H3.

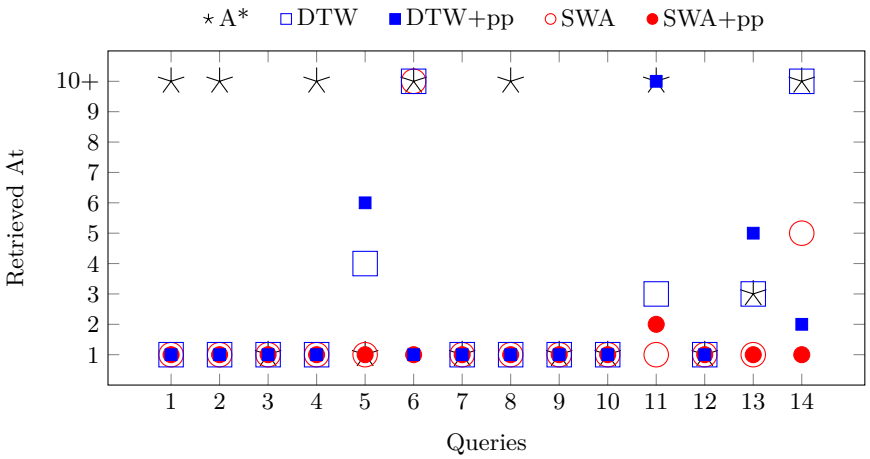

Fig. 5. Similarity Ranking for Best Matching Cases

6 Conclusion and Future Work

We presented an approach for a case-based deviation management in the context of flexible workflow execution. In this paper we focussed on the retrieval phase and therefore proposed a similarity measure that is able to compare completed and running workflow instances where a deviation occurred. Therefore we utilized time sequence similarity enhanced by a weighting factor that determines the temporal distance to the deviation. Furthermore, we presented a pre-processing method for the case base in order to ignore differences that result from the simple sequential tracing of tasks in the de facto workflows. We implemented the developed algorithms in ProCAKE and evaluated our similarity measure in comparison to a graph-based mapping algorithm based on A* search. The runtime and similarity assessment comparison show promising results. As

[3] Retrieval time of the A* approach was corrected for edge mapping which accounts for approximately halve of the runtime. The experiments were conducted on a personal computer with an 8-core Intel Core i7-6700 CPU @3.4GHz and 32GB of RAM.

we aim at supporting workflow continuation with a retrieved similar case, an evaluation of usefulness is still pending. Therefore recommended tasks after the deviation, resulting from the most similar case, need to be rated.

Furthermore, as future work the remaining phases of the CBR cycle that were not considered in detail until now, but rather briefly sketched, will be investigated. Adaptation methods will be developed in order to increase the suitability of the solution for the query. Therefore the existing constraints that are processed by the workflow engine, will be taken into account, as they represent the de jure workflow, actual information about the de facto workflow as well as additional semantic information, e.g. state of the constraints (valid vs. violated).

Additionally, the application of the presented pre-processing algorithm needs to be refined. The results of the evaluation already indicate that the reduction cannot be applied reasonably in every condition, but rather needs to be weighed. The decision, whether model-consistency should be more important than regarding additional differences between case and query, should rather be made based upon the de jure workflow model. Data-flow dependencies could be taken into account or a manual decision about which workflow block simply needs to be model-consistent during case comparison, for example depending on the type of control-flow block (parallel vs. exclusive), could be considered. In case of an exclusive block the reduction would result in equal treatment of excluding task executions, even if they possibly have a totally different impact on the deviation or rather on the continuation after the deviation. A concept for deciding in which circumstances reduction of case and query is appropriate will be elaborated.

Acknowledgements. This work is part of the research project SEMANAS and is funded by the Federal Ministry of Education and Research (BMBF), grant no. 13FH013IX6.

References

1. van der Aalst, W.M.P.: Business Process Management - A Comprehensive Survey. ISRN Software Engineering **2013**, 1–37 (2013)
2. Adams, M., ter Hofstede, A.H.M., van der Aalst, W.M.P., Edmond, D.: Dynamic, extensible and context-aware exception handling for workflows. In: On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I. pp. 95–112 (2007)
3. Bergmann, R., Gil, Y.: Similarity Assessment and Efficient Retrieval of Semantic Workflows. Information Systems **40**, 115–127 (2014)
4. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370 (1994)
5. Casati, F., Ceri, S., Paraboschi, S., Pozzi, G.: Specification and implementation of exceptions in workflow management systems. ACM Trans. Database Syst. **24**(3), 405–451 (1999)
6. Dijkman, R., Dumas, M., Van Dongen, B., Krik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems **36**(2), 498–516 (2011)

7. Döhring, M., Zimmermann, B., Godehardt, E.: Extended workflow flexibility using rule-based adaptation patterns with eventing semantics. In: Informatik 2010: Service Science - Neue Perspektiven für die Informatik, Beiträge der 40. Jahrestagung der GI, Band 1, Leipzig, Deutschland. pp. 195–200 (2010)
8. Grambow, G., Oberhauser, R., Reichert, M.: Event-driven exception handling for software engineering processes. In: Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I. pp. 414–426 (2011)
9. Grumbach, L., Bergmann, R.: Semaflex: A novel approach for implementing workflow flexibility by deviation based on constraint satisfaction problem solving. Expert Systems (03 2019)
10. Grumbach, L., Bergmann, R.: Towards case-based deviation management for flexible workflows. In: Proceedings of the Conference LWDA, Berlin, Germany, September 30 - October 2, 2019. pp. 241–252 (2019)
11. Grumbach, L., Rietzke, E., Schwinn, M., Bergmann, R., Kuhn, N.: SEMAFLEX - Semantic Integration of Flexible Workflow and Document management. In: Proceedings of the Conference LWDA, Potsdam, Germany, September 12-14, 2016. pp. 43–50 (2016)
12. Grumbach, L., Rietzke, E., Schwinn, M., Bergmann, R., Kuhn, N.: SEMANAS - semantic support for grant application processes. In: Proceedings of the Conference LWDA , Mannheim, Germany, August 22-24, 2018. pp. 126–131 (2018)
13. Gundersen, O.E.: Toward measuring the similarity of complex event sequences in real-time. In: Case-Based Reasoning Research and Development , ICCBR 2012, Lyon, France, September 3-6, 2012. Proceedings. pp. 107–121 (2012)
14. Müller, R., Greiner, U., Rahm, E.: $A_{gent}w_{ork}$: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng. **51**(2), 223–256 (2004)
15. Obweger, H., Suntinger, M., Schiefer, J., Raidl, G.: Similarity searching in sequences of complex events. In: 2010 4th International Conference on Research Challenges in Information Science - Proceedings, RCIS 2010. pp. 631–640 (2010)
16. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43–49 (1978)
17. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Towards a taxonomy of process flexibility. In: Proceedings of the Forum at the CAiSE'08 Conference, Montpellier, France, June 18-20, 2008. pp. 81–84 (2008)
18. da Silva, M.A.A., Bendraou, R., Robin, J., Blanc, X.: Flexible deviation handling during software process enactment. In: Workshops Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOCW 2011, Helsinki, Finland, August 29 - September 2, 2011. pp. 34–41 (2011)
19. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of molecular biology **147**(1), 195–197 (1981)
20. Weber, B., Wild, W., Breu, R.: Cbrflow: Enabling adaptive workflow management through conversational case-based reasoning. In: Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings. pp. 434–448 (2004)
21. Wombacher, A., Rozie, M.: Evaluation of workflow similarity measures in service discovery. In: Service-Oriented Electronic Commerce, Proceedings zur Konferenz im Rahmen der MKWI 2006. Gesellschaft für Informatik eV (2006)
22. Zarka, R., Cordier, A., Egyed-Zsigmond, E., Lamontagne, L., Mille, A.: Similarity measures to compare episodes in modeled traces. In: International Conference on Case-Based Reasoning. pp. 358–372. Springer (2013)