



A Case-Based Approach for Workflow Flexibility by Deviation

Lisa Grumbach¹  and Ralph Bergmann^{1,2} 

¹ German Research Center for Artificial Intelligence (DFKI),
Branch University of Trier, Behringstraße 21, D-54296 Trier, Germany
{lisa.grumbach,ralph.bergmann}@dfki.de

² Business Information Systems II, University of Trier, D-54296 Trier, Germany
bergmann@uni-trier.de
<http://www.wi2.uni-trier.de>

Abstract. This paper presents a case-based approach for workflow flexibility by deviation. In previous work, a constraint-based workflow model and engine have been developed that allow for flexible deviations from predefined workflow models during run-time. When encountering deviations, domain-independent strategies can be applied for a resolution in order to regain support for the process participant. To improve this deviation handling, a case-based approach is presented that integrates experiential knowledge by exploiting previously terminated workflows as cases. Similar cases are retrieved through a time-series based similarity measure and reused through null adaptation. The experimental evaluation showed an improvement of the defined utility value concerning the computed work items, when comparing the constraint-based workflow engine and the case-based deviation management.

Keywords: Workflow Flexibility · Case-Based Reasoning · Knowledge Management.

1 Introduction

Flexible workflows have been researched for more than a decade [14], as customer-orientation is becoming more and more important and therefore adapting to specific needs is substantial. In traditional workflow systems, models are specified at design-time and describe the ideal order of tasks. During run-time these models are instantiated and strictly prescribe the order of task execution, whereas deviations are only possible when circumventing the system. Thus, all possible execution variants have to be modelled at design-time, which requires a great effort and is barely possible, as not all situations can be foreseen. A solution would be to model the standard procedure with few but frequent workflow variants and allow controlled deviations at run-time. To this end, an approach is necessary that handles deviations from predefined models in an automated manner and offers flexibility through continuous support to the process participant about how to successfully terminate the workflow.

On the one hand, we proposed a constraint-based workflow approach, that is able to retract violated constraints at run-time and subsequently utilize remaining constraints for workflow control. On the other hand, domain-specific knowledge can be used to suggest tasks for workflow progression. For this purpose, we aim at exploiting experiential knowledge.

Flexible workflows are beneficial in several domains, such as production planning [9] or situation management [11]. We investigate this approach in the context of the exemplary domain of deficiency management in construction. This workflow is essential in the construction sector, as all projects require a final approval for which all defects have to be eliminated. For this process, flexibility is crucial, as not all types of defects and their handling can be known in advance.

In Section 2 relevant foundations for the approach are presented including workflow flexibility, semantic workflows and our constraint-based workflow engine. In Section 3, the concept of the envisioned case-based deviation management is introduced, which aims at enhancing deviation handling capabilities of the constraint-based workflow engine. The focus is laid on the retrieve and reuse phase. The approach is evaluated based on an experiment in the chosen domain of deficiency management in construction that simulates the usage of the flexible workflow system in Section 4. The paper concludes with reflecting the findings and giving an outlook on future research in Section 5.

2 Foundations

In this section a basic classification of workflow flexibility and a specification of the underlying terminology will be given. This includes the used semantic workflows and our previously presented constraint-based workflow engine.

2.1 Workflow Flexibility

Workflow flexibility can be categorized into four types [15]. *Flexibility by Design* requires to incorporate all execution alternatives into the model at design-time. *Flexibility by Underspecification* allows instantiating partial models that require integrating placeholders during design-time, either blank ones or several alternatives, specified during run-time. *Flexibility by Change* allows interventions during run-time and a re-modeling of parts of the workflow. All of these flexibility variants require either knowledge about all possible alternatives at design-time, which is impossible, or an adaptation at run-time, which requires expert knowledge concerning workflow modeling. The fourth variant, *Flexibility by Deviation* tries to counteract these disadvantages by enabling deviations at run-time, without necessary modeling effort, but still supporting the process participant with suggestions about the next activities. Thus, single instances may not fit to the workflow model. Therefore we explicitly distinguish between modeled, denoted as *de jure*, and executed workflow, called *de facto* [1]. The de facto workflow is not an instantiated de jure workflow, but represents the actually enacted tasks traced sequentially. Based on this definition we developed a workflow engine

facilitating flexibility by deviation. Only few research exists concerning this approach, which still requires a manual intervention of the process participant [2].

2.2 Constraint-Based Workflow Engine

During previous projects we developed a flexible workflow engine based on constraints. We presented an approach [5,6] that allows flexible deviations from prescribed workflows, but still maintains control and recommends valid work items to a limited extent. The proposed method is applied to imperatively modeled block-oriented workflows. Those 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. An example is shown in Fig. 1.

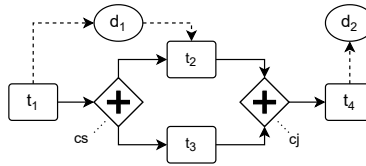


Fig. 1. Example Block-Oriented Workflow

The exemplary workflow consists of four task nodes (rectangles), two data nodes (ovals) and two control-flow nodes (rhombuses), which represent a parallel control-flow block (“+”). Additionally, the edges denote either control-flow (solid lines) or data-flow (dashed lines, input or output relation).

In our approach we transform these imperative workflow models into declarative constraints, which indicate sequential dependencies, to be able to determine task activations and thus possible executions in a specific untermiated 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 tries to calculate a valid sequential enactment of all already executed and possible future executions of tasks. Thus, with a solution we are able to recommend valid task enactments. Consider the example of Fig. 1, the constraint set as logical formula is constructed as follows: $t_1 < t_2 \wedge t_1 < t_3 \wedge t_2 < t_4 \wedge t_3 < t_4$. If task t_1 is executed, a value is assigned, in this case $t_1 = 1$, and added to the constraint set. Task recommendations are calculated by regarding possible task assignments of the next sequential value, in this case 2. Considering the constraint set either with $t_2 = 2$ or with $t_3 = 2$ a solution is found. Thus, t_2 and t_3 are added as work items to the work list.

An advantage of using a CSP is that it is easy and fast to retract violated constraints at run-time for restoring consistency in case of a deviation. Still, by regarding the remaining constraints valid solutions can be computed. In our work [5], we described a method which detects deviations and retracts violated

constraints to restore consistency and re-enable the workflow engine to recommend work items. Different domain-independent strategies can be applied that assume a deviation category and transform the constraint net to adapt to the deviating situation. However, to apply these strategies, a choice must be made that requires knowledge about the deviation and the impact of the strategy. To further automate the deviation handling and to offer more sophisticated decision support based on experiential knowledge, we proposed a case-based approach.

2.3 Semantic Workflows

In the proposed case-based deviation management, semantic workflows are used as case representation, as they allow for an enrichment of semantic descriptions for tasks and data nodes and the workflow as such. Since the similarity assessment is additionally based on these semantic annotations, it is more sophisticated. The utilized specification of semantic workflows is denoted as *NESTGraph* [3], and specified as quadruple $W = (N, E, S, T)$ where

- N is a set of nodes and
- $E \subseteq N \times N$ is a set of edges.
- $S : N \cup E \rightarrow \Sigma$ associates to each node and each edge a *semantic description* from a semantic metadata language Σ .
- $T : N \cup E \rightarrow \Omega$ associates to each node and each edge a type from Ω .
- Ω contains the following types of nodes: workflow, task, data, control-flow and types of edges: part-of, data-flow, control-flow and constraint.

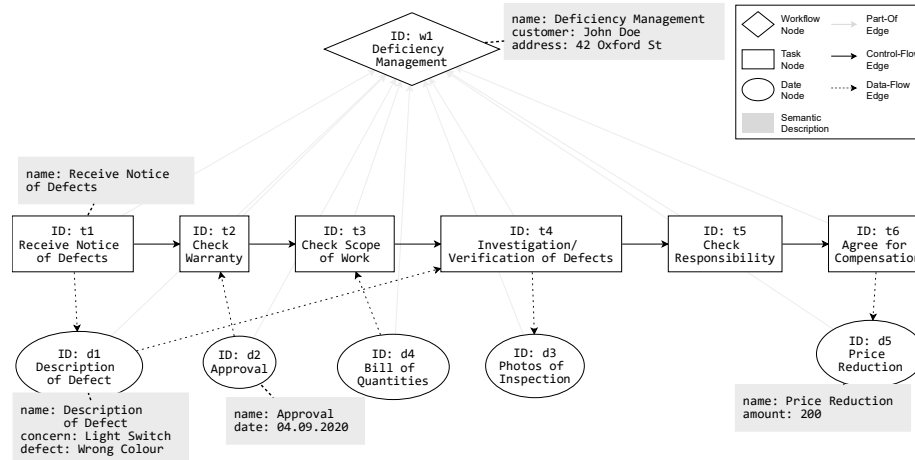


Fig. 2. Exemplary Block-Oriented Semantic Workflow Graph

Fig. 2 shows an excerpt of a workflow instance from the domain of deficiency management in construction. This workflow consists of six task nodes, five data

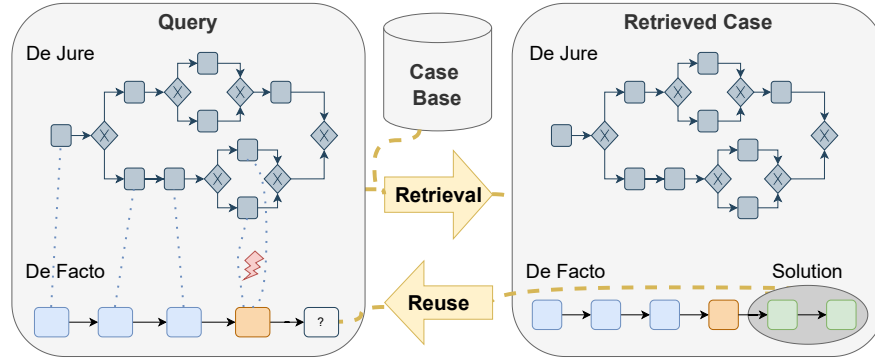


Fig. 3. Case-Based Deviation Management [5]

nodes and one workflow node. The edges denote either control-flow, data-flow (input/output relation) or part-of edges. Furthermore, each node is associated with a semantic description, which contains additional information. In Fig. 2, some exemplary semantic descriptions are shown. The workflow node relates to some general information that concerns the whole workflow. The tasks' semantic descriptions only contain the name. The additional information of data nodes differ concerning the attributes. For instance in data node *d1* further details are provided about the *description of defect* such as *concern* and *defect*.

The developed case-based approach[5,13] is presented in the next section.

3 Concept for a Case-Based Deviation Management

The case-based deviation management includes previously terminated workflows in the decision process of how to continue with the workflow after a deviation occurred. Fig. 3 shows the overall approach.

Case Structure Terminated workflows are regarded as cases. Each workflow case is a pair $WC = (J_C, F_C)$ with J_C as de jure workflow, which is the default model for suggesting an execution order, and F_C as de facto workflow, representing the actually enacted tasks, which are traced sequentially. Both workflows are represented as *NESTGraphs*. The query contains a de facto workflow that has not yet terminated and is facing a deviation concerning the de jure workflow (see orange node in Fig. 3).

Retrieve When a query is performed, the case base is searched through on the basis of an adequate similarity measure. The objective is to find similar de facto workflows whose subsequences match the current instance (see de facto workflows with blue and orange nodes in Fig. 3), containing a similar deviation (orange node) compared to the query. The de jure workflow is so far not considered in the similarity assessment, as the focus is on the deviation in the de facto workflow. In the retrieved cases, the subsequences

that succeed the deviation (see green-coloured nodes in Fig. 3) should not be considered when assessing the similarity, as this part is not existent in the query, but is rather a solution candidate.

Reuse This most similar case or even several similar cases are then used to recommend tasks. The simplest reuse option, denoted as null adaptation, is to propose those tasks of the case that followed the subsequence ending with the deviation (see green-coloured nodes in Fig. 3).

Revise As tasks are only recommended, the process participant is still able to execute a task that was not part of the solution resulting from the reuse step. Thus, by continuing the query workflow, the solution can be revised. Still, an evaluation of the terminated workflow is pending. This assessment decides whether the revised case can be retained as successful or failed case.

Retain When the query workflow has terminated, its de facto workflow, containing the actual execution, can be integrated in the case base. Ideally, some kind of validation, positive or negative, is stored with the case in order to draw the correct conclusions in subsequent reuse steps. Before integrating single cases, it needs to be evaluated whether the informativeness can be increased or whether this additional knowledge is already covered by adaptation methods. Nevertheless, the case base can be enhanced continuously, which ultimately leads to a learning system.

This paper focusses on retrieval and reuse. On account of this, two variants of a similarity measure and one adaptation method are presented. The revise and retain phase are part of future work and were therefore only presented as abstract idea.

3.1 Retrieval with a Time-Series Based Similarity Measure

The retrieval phase bases on a similarity measure that is able to compare time series, presented in our previous work [13]. This measure is applied on the sequential de facto workflows and searches for a mapping of tasks considering edit distance by applying either the Smith-Waterman-Algorithm (SWA) [16] or warping of elements through the use of dynamic time warping (DTW) [4,12]. Besides, analogously to the vector similarity of Gundersen [7], local mappings are weighted according to their distance to the currently regarded task based on the assumption that tasks which are more far in the past have less influence on the deviation and subsequent tasks. Furthermore, data-flow similarity is included in the local similarity values of tasks considering their input and output data objects. The local similarity sim_T for tasks t^Q and t^C is used during alignment:

$$sim_T(t^Q, t^C) = \frac{l_t * sim_N(t^Q, t^C) + l_i * sim_D(N_{t^Q}^{D_{in}}, N_{t^C}^{D_{in}}) + l_o * sim_D(N_{t^Q}^{D_{out}}, N_{t^C}^{D_{out}})}{l_t + l_i + l_o} \quad (1)$$

Each task node similarity is calculated by comparing the semantic descriptions of task nodes sim_N , finding a mapping of ingoing ($N^{D_{in}}$) and outgoing ($N^{D_{out}}$) data nodes sim_D . Both variants of the similarity measure are based on computing a scoring matrix, integrating the temporal weighting factor. For more details we refer to our previous work [13]. The main difference of both methods is visible in the computation of the scoring matrix. Whereas DTW weights the similarity value according to warp or mapping of elements, SWA includes a penalty, which can be a constant value or a function, when the mapping origins out of an insertion or deletion and only considers the similarity when matching elements. The maximum value in the last row of the matrix represents the non-normalized global similarity score, which is further normalized to a value $sim \in [0, 1]$. More details were presented in our previous work [5,13].

3.2 Reuse through Null Adaptation

With the previously presented similarity measure, the most similar terminated workflow or a set of workflows can be retrieved from the case base. This can include several distinct cases with the same similarity score. Let $retrieved_Q$ be the set of retrieved cases with the highest similarity values. From these case workflows, work items can be derived.

Let $WC_i \in retrieved_Q$ be one of the similar cases. Then, one work item can be derived from the alignment in the scoring matrix. The position in the de facto workflow of the case that was aligned with the deviating task of the query needs to be determined (cf. orange-coloured node in Fig. 3). Let $align = \{(0, 0), \dots, (n, k)\}$ denote the alignment path. Then, the task $t_{last}^C \in N_{FC}^T$ with position $pos(t_{last}^C) = k$ is the task of the case workflow that was the last to be aligned with an element of the query workflow. All tasks that are subsequent to t_{last}^C were not aligned to tasks of the query. The next one of these tasks can be recommended to the process participant as the next task, thus at $pos(t_{next}) = k + 1$ (cf. green-coloured nodes in Fig. 3).

$$\begin{aligned} workItems = \{ & t_{next} | t_{next} \in N_C^T \wedge pos(t_{next}) = pos(t_{last}^C) + 1 \wedge \\ & WC_i = ((N_C^T, E, S, T), F_C) \in retrieved_Q \} \end{aligned} \quad (2)$$

This adaptation method is efficient, as it simply bases on the mapping that origins from the retrieval phase without requiring additional handling.

4 Evaluation

The actual usefulness of the proposed approach considering the exemplary domain can only be evaluated in an empirical study with real experts in the construction industry. However, the necessary effort is too high. Instead, we conducted a study with simulated users under the assumption that the simulation at least roughly reflects the behaviour of real process participants[5].

4.1 Utility Criteria for Defined Process Participant Types

The essential criteria which is investigated is the *utility*. It is interpreted in this domain of flexible workflows as the degree to which the workflow can be terminated successfully. The overall utility can be determined through measuring the utility of work items during execution. To assess this property, various types of process participants are regarded in order to investigate and evaluate different behaviour of process participants and the corresponding reaction of the system.

Experienced The experienced process participant constitutes an expert and knows how to handle any situation during process execution. This not only includes the ideal way of completing a workflow, but also recovering after deviations in case the workflow system is in an exceptional state without proper task suggestions. Her/his expectations of a flexible workflow system concern the support of those workflow instances that s/he would produce. The workflow engine is useful for this expert if the proposed work items correspond to the task executions s/he has in mind. Hence, utility of the workflow engine is measured as amount of support in contrast to misguidance. This is assessed by comparing the pursued task execution with the proposed work items of the workflow engine. If the task, which the experienced process participant would execute, is not among the list of proposed work items, the workflow engine would lead to a miscontrol, which is not useful. The utility can then be assessed on the basis of the ratio of the number of miscontrols (*nrOfMiscontrols*) and the number of task executions (*nrOfTasks*) in a workflow as degree of support:

$$utility_{experienced} = 1 - \frac{nrOfMiscontrols}{nrOfTasks} \quad (3)$$

Inexperienced The inexperienced process participant, who can be regarded as a novice, relies on the support of the workflow engine, as s/he is not aware of the correct workflow execution and does not know how to continue after deviations. Each workflow instance executed by him/her is strictly adhering the suggested work items. Consequently, the success of an inexperienced process participant completely depends on the appropriateness of the work items proposed by the workflow engine. Hence, the workflow instances that can result from the execution of proposed work items are the basis to assess the degree of utility. Therefore, these resulting workflow instances can be compared to valid workflow instances. The minimum edit distance [8] gives information about the amount of conformance. Relating this edit distance (*editDistance*) to the total number of executed tasks (*nrOfTasks*) in the workflow instance indicates the utility:

$$utility_{inexperienced} = 1 - \frac{editDistance}{nrOfTasks} \quad (4)$$

Non-Conforming The non-conforming process participant executes his/her workflows in a non-conforming way, which includes executing tasks that are

not part of the proposed work items, thus causing deviations from the de jure workflow with unknown consequences. The expectation of using a flexible workflow system is that, when encountering one of those undesired deviations again, the guidance leads to a better outcome than before. Successfully supporting a non-conforming process participant is interpreted as putting him/her back on the right track. The workflow engine should provide work items in order to recover from an undesired deviation. Taking a workflow instance with more than one deviation as starting point and letting the process participant complete the workflow instance, always following the work items proposed by the workflow engine after the first deviation, an adapted workflow instance emerges. In this case utility is defined as the improvement with respect to the workflow instance the user would create without support of the workflow engine, indicated by the reduction of necessary edit steps. Therefore, the minimum edit distance to a valid workflow of both the workflow instances that emerge with and without support of the workflow engine can be compared.

$$utility_{non-conforming} = 1 - \frac{\left(\frac{editDistanceNew}{nrOfTasksNew}\right)}{\left(\frac{editDistanceOriginal}{nrOfTasksOriginal}\right)} \quad (5)$$

4.2 Hypothesis

In the experimental evaluation the following hypotheses are investigated, which refer to the defined utility criteria and a comparison of the constraint-based workflow engine and the case-based null adaptation for determining work items.

- H1** (*Experienced Process Participant*): When replaying a valid workflow, the executed tasks are mostly those proposed by the workflow engine. Consequently, the workflow engine achieves a high utility for the experienced process participant.
- H2** (*Inexperienced Process Participant*): After a deviation from the workflow model, workflows can be terminated similarly to valid ones based on the suggestions of the workflow engine. Thus, the edit distance of proposed workflow instances compared to the most similar case of valid previous executions is small, indicating a high utility for the inexperienced process participant.
- H3** (*Non-Conforming Process Participant*): The workflow engine can improve the outcome of workflow instances that have been terminated previously leading to an invalid workflow. Workflow instances that are terminated according to the proposed work items by the workflow engine after the first deviation have a smaller edit distance to the most similar valid workflow than the original workflow instance. Consequently, the utility for the non-conforming process participant is high.
- H4** (*Comparing the Utility resulting from the Constraint-Based Workflow Engine and Null Adaptation*): The utility for each type of process participant is significantly higher when using the case-based null adaptation compared to using the constraint-based workflow engine.

4.3 Experimental Setup

Figure 4 shows the overall setting of the experimental evaluation.³ As exemplary use case the domain of deficiency management in construction was regarded. Therefore two different workflow models were defined in consultation with an architect as expert. On the one hand, a model that specifies all standard cases is used as input for the workflow engine, as the situation would be when using a traditional system (see *simple model* in Fig. 4). On the other hand, this simple model was extended with various alternative execution paths, mapping the real-world handling of deficiencies (see *extended model* in Fig. 4). Generating this model means a lot of effort, which is to be avoided in practice. This extended model served for generating all possible de facto workflows that form the total list of valid cases in order to evaluate the approach. Based on these de facto

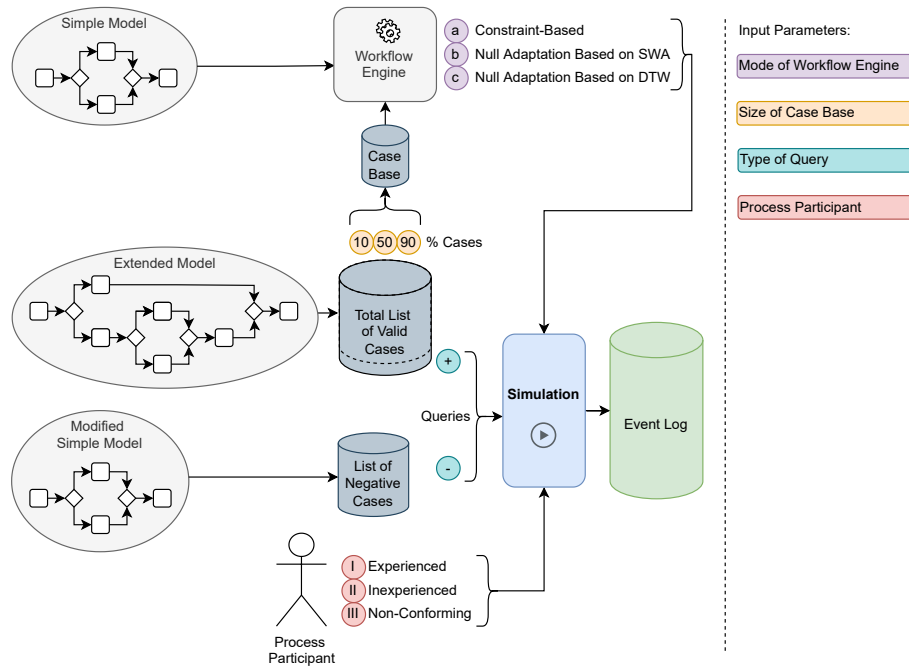


Fig. 4. Experimental Setup

workflows, workflow executions are simulated.

The experiment is done as a ten-fold cross validation, such that for each single validation run 10% of the cases of the total list of cases form the test set, which are used as queries. The total list of valid cases contains 2.184 positively rated

³ The experiments were made on a laptop with (QuadCore) Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz, 16 GB RAM, 64 bit system, Windows 10.

traces. Each query is simulated as a trace replay several times with different parameters. The simulation produces an event log that is analysed concerning the assumed hypotheses. The parameters include:

- mode of workflow engine (\textcircled{a} - \textcircled{c}), which determines how work items are computed after each task
- size of case base (10/50/90%)
- type of query (\oplus/\ominus), *positively* rated queries are simulated for the *experienced* and *inexperienced* process participants, whereas *negatively* rated queries are simulated for the *non-conforming* process participant.
- type of process participant ((I)/(II)/(III))

Experienced To evaluate the utility for the experienced process participant, a replay of the cases in the test set is simulated as presented in Fig. 5. For

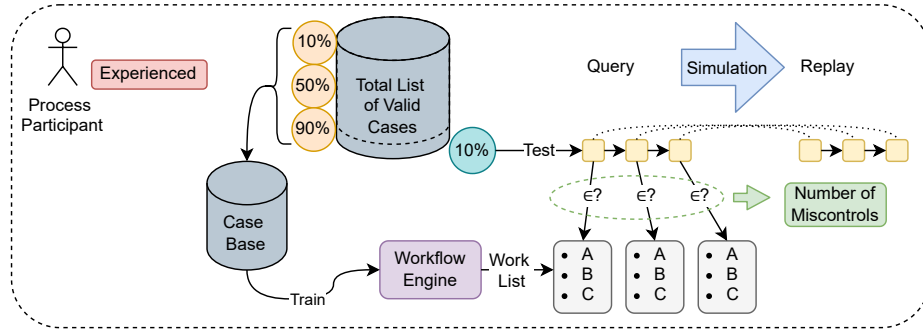


Fig. 5. Simulation of the Experienced Process Participant

the experienced process participant, the exact trace is replayed for each query. During this trace replay, for each task that is replayed and added to the de facto workflow, it is checked if it is contained in the proposed work items of the workflow engine. When the trace replay terminates, the utility for experienced process participants is computed on the basis of the number of miscontrols.

Inexperienced The simulation of the inexperienced process participant is visualized in Fig. 6. In a first step, the query workflows are re-enacted as trace replay. For this purpose, the constraint-based workflow engine is used as ideal workflow execution reference using the simple model as input. The point of investigation concerning utility starts when a deviation occurs, in terms of a task in the replay that is not part of the work items proposed on the basis of the simple model. Then, the deviation handling modes of the workflow engine are activated and the replay is completed following the suggestions of the worklist (see blue-coloured nodes of *Replay* in Fig. 6). Work items are picked randomly. When the worklist

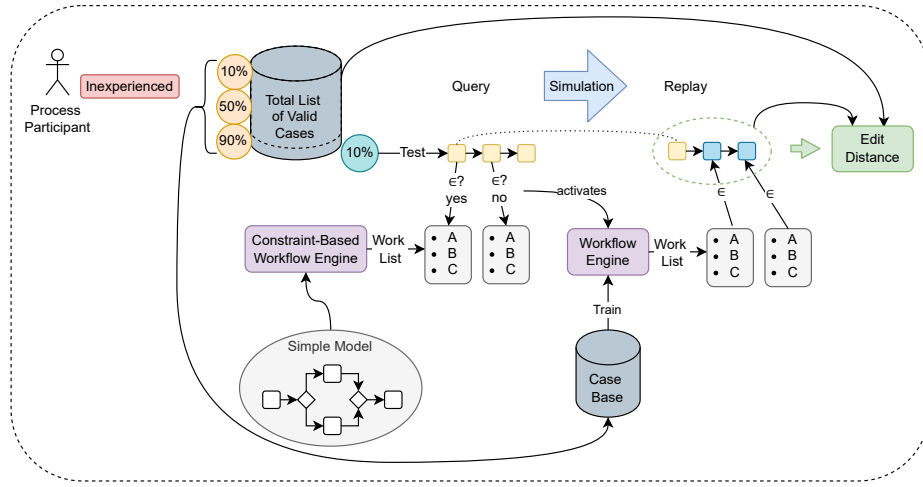


Fig. 6. Simulation of the Inexperienced Process Participant

is empty, the workflow instance concludes. This emerging workflow instance can be rated considering its similarity based on the edit distance compared to the total list of valid cases resulting in a utility value.

Non-Conforming The non-conforming process participant executes workflow instances with unknown deviations. Thus, the simulation is not based on the

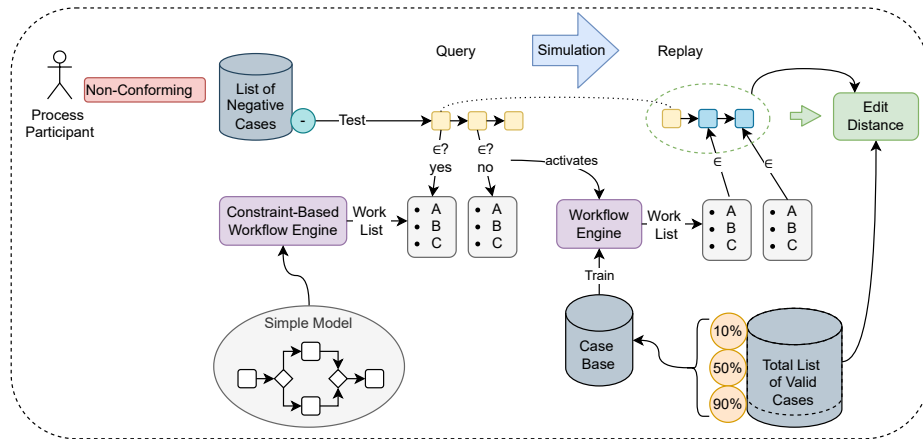


Fig. 7. Simulation of the Non-Conforming Process Participant

test set of the total list of valid cases that resulted from the extended workflow

model, but the regarded query workflows are negative cases that were generated on the basis of a manipulated simple workflow model (see Fig. 7). The simulation itself is done in the same way as for the inexperienced user. Here, the resulting similarity value can be compared to the edit distance of the original query and the improvement can be quantified, which leads to the utility assessment.

4.4 Results

In summary, 31416 trace replays resulted from the experiment in which the process participants were simulated with the previously described parameters. The replayed workflows contained ten tasks on average, at maximum 26 and at least two tasks. The resulting average utility values are presented in Fig. 8.

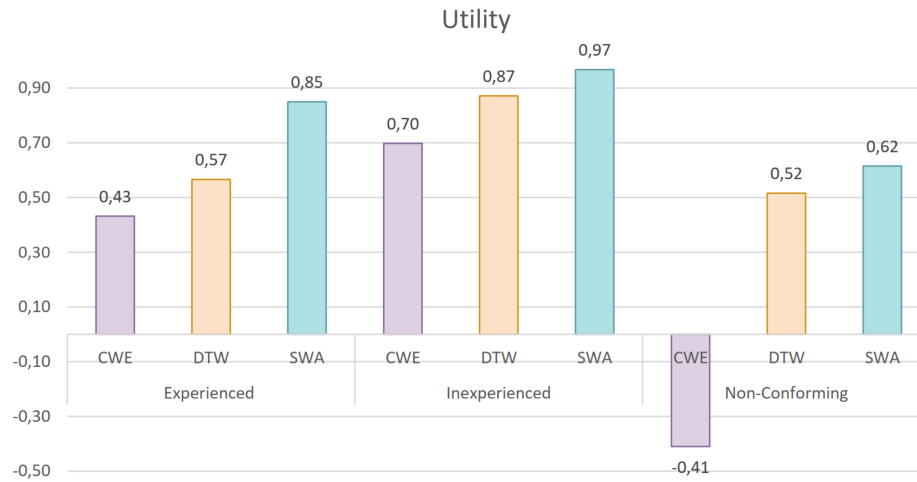


Fig. 8. Average Utility Values for the Different Types of Process Participant and for Different Modes of the Workflow Engine (CWE = Constraint-Based Workflow Engine)

Considering the resulting traces of the experienced process participant, the number of miscontrols ranges from 0 to 11, with an average of four miscontrols. Referring to the derived utility values, the null adaptation based on a retrieval with the SWA shows the most conformance with the traces with an average value of 85 % of tasks that were proposed by the workflow engine, whereas this holds for only 43 % on average when computing the worklist on the basis of the constraint-based workflow engine.⁴ The utility value for the null adaptation with DTW lies in between. Hence, hypothesis **H1** is confirmed for both null

⁴ The poor performance of the constraint-based workflow engine can be explained due to the differing inputs of the methods. The only input for the constraint-based approach is the simplified workflow model with a reduced set of tasks compared to the extended model. Consequently, work items are only derived from this reduced

adaptation variants, as the workflow traces are largely compliant ($> 50\%$) with the proposed work items, which verifies a great support for the experienced process participants.

Considering the results for the inexperienced process participant, the performance is similar. With an average value of 0,87 and 0,97 for both null adaptation variants, the utility is high and consequently, hypothesis **H2** is confirmed for those two methods. In contrast, the constraint-based workflow engine performs not as good with an average value of 0,7.

For the non-conforming process participants the results differ significantly, foremost of the constraint-based workflow engine. Both null adaptation variants with either the SWA or DTW yield overall high utility values around 0,5 and higher. The constraint-based workflow engine mostly leads to a negative utility value, which means it is tending to increase the edit distance slightly, resulting in an impairment of the support for the non-conforming process participant, as the original trace is more similar to the most similar valid workflow. Consequently, hypothesis **H3** can be confirmed for both null adaptation methods, but is denied for the constraint-based workflow engine.

The case-based modes of the workflow engine outperform the constraint-based one for each type of process participant, confirming hypothesis **H4**.

5 Conclusion

We presented an approach for workflow flexibility that allows for supporting alternatives when facing deviations during workflow execution. On the one hand the workflow can be controlled through a constraint-based workflow engine that uses several domain-independent strategies for conflict resolution, on the other hand the case-based deviation management on the basis of a time-series similarity measure and two null adaptation variants integrated experiential knowledge for the determination of work items. The experimental evaluation showed promising results and an improvement of utility when applying the null adaptation methods in the exemplary domain and simulated environment.

In future work, the null adaptation variants could be enhanced through considering the edit steps similar to trace-based reasoning [10]. Moreover, we developed an approach based on generative adaptation that includes the constraint net in the reuse phase in order to improve flexibility through a transfer of relational task dependencies instead of fixed task positions [5]. However, the proposed approach did not perform well and needs a further elaboration and revision. Furthermore, the evaluation needs to be extended to other domains and real-world scenarios to show the general applicability.

Acknowledgements. This work is funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK) under grant no. 22973.

set, while all additional tasks are not part of the build CSP and therefore cannot be part of the solution. In contrast, both null adaptations rely on the experiential knowledge in form of workflow traces, which includes all tasks of the extended model.

References

1. van der Aalst, W.M.P.: Business process management: a comprehensive survey. *ISRN Software Engineering* pp. 1–37 (01 2012)
2. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. *Data Knowl. Eng.* **53**(2), 129–162 (2005)
3. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* **40**, 115–127 (2014)
4. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In: Fayyad, U.M., Uthurusamy, R. (eds.) *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, Seattle, Washington, USA, July 1994. Technical Report WS-94-03. pp. 359–370. AAAI Press (1994)
5. Grumbach, L.: *Flexible Workflows - A Constraint- and Case-Based Approach*. Ph.D. thesis, University of Trier, Germany (2023)
6. Grumbach, L., Bergmann, R.: SEMAFLEX: A novel approach for implementing workflow flexibility by deviation based on constraint satisfaction problem solving. *Expert Syst. J. Knowl. Eng.* **38**(7) (2021)
7. Gundersen, O.E.: Toward Measuring the Similarity of Complex Event Sequences in Real-Time. In: Díaz-Agudo, B., Watson, I. (eds.) *Case-Based Reasoning Research and Development - 20th International Conference, ICCBR 2012, Lyon, France, September 3-6, 2012. Proceedings. Lecture Notes in Computer Science*, vol. 7466, pp. 107–121. Springer (2012)
8. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady* **10**, 707–710 (1965)
9. Malburg, L., Brand, F., Bergmann, R.: Adaptive management of cyber-physical workflows by means of case-based reasoning and automated planning. In: Sales, T.P., Proper, H.A., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) *Enterprise Design, Operations, and Computing. EDOC 2022 Workshops - IDAMS, SoEA4EE, TEAR, EDOC Forum, Demonstrations Track and Doctoral Consortium, Bozen-Bolzano, Italy, October 4-7, 2022, Revised Selected Papers. Lecture Notes in Business Information Processing*, vol. 466, pp. 79–95. Springer (2022)
10. Mille, A.: From case-based reasoning to traces-based reasoning. *Annu. Rev. Control.* **30**(2), 223–232 (2006)
11. Rietzke, E., Maletzki, C., Bergmann, R., Kuhn, N.: Execution of knowledge-intensive processes by utilizing ontology-based reasoning. *J. Data Semant.* **10**(1-2), 3–18 (2021)
12. 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)
13. Schake, E., Grumbach, L., Bergmann, R.: A Time-Series Similarity Measure for Case-Based Deviation Management to Support Flexible Workflow Execution. In: Watson, I., Weber, R.O. (eds.) *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings. Lecture Notes in Computer Science*, vol. 12311, pp. 33–48. Springer (2020)
14. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Process Flexibility: A Survey of Contemporary Approaches. In: Dietz, J.L.G., Albani, A., Barjis, J. (eds.) *Advances in Enterprise Engineering I, 4th International Workshop CIAO! and 4th International Workshop EOMAS, held at CAiSE 2008, Montpellier, France, June 16-17, 2008. Proceedings. Lecture Notes in Business Information Processing*, vol. 10, pp. 16–30. Springer (2008)

15. 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)
16. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. *Journal of molecular biology* **147**(1), 195–197 (1981)