# Considering Inter-Case Dependencies During Similarity-Based Retrieval in Process-Oriented Case-Based Reasoning

**Rahol Kumar[1], Alexander Schultheis[1], Lukas Malburg[1,2], Maximilian Hoffmann[1,2], Ralph Bergmann[1,2]**
[1] Artificial Intelligence and Intelligent Information Systems, University of Trier, 54296 Trier, Germany
{s4rakuma,s4axschu,malburgl,hoffmannm,bergmann}@uni-trier.de,
http://www.wi2.uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI)
Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
{lukas.malburg,maximilian.hoffmann,ralph.bergmann}@dfki.de

**Abstract**

In *Case-Based Reasoning (CBR)*, knowledge gained from previously experienced problem-solving situations is stored as cases that can be used to solve similar upcoming problems. Although these cases act as independent knowledge entities, dependencies between cases are common in real-world scenarios, despite being only rarely considered during case retrieval or other CBR phases. In this paper, we introduce so-called inter-case dependencies, which are considered in the context of *Process-Oriented CBR (POCBR)*. Therefore, we 1) derive requirements that must be satisfied for considering dependencies during the retrieval phase, 2) analyze which knowledge representations are suitable for representing dependencies between cases, and, 3) present our approach for *Dependency-Guided Retrieval (DGR)* that considers these dependencies between cases during the retrieval phase. In the experimental evaluation, the proposed DGR approach is compared to a regular CBR approach in case retrieval scenarios from the cooking domain. The results demonstrate that the use of the DGR approach leads to significantly reduced times for human problem-solving compared to regular CBR.

## 1 Introduction

Workflow management is increasingly used in several domains such as in companies, in e-Science, or in smart environments. For this reason, reusing procedural experiential knowledge is gaining importance in workflow and knowledge management (Bergmann et al. 2019). In *Workflow Management Systems (WfMSs)* (Dumas et al. 2018), several workflows, e. g., from different departments in a company, are executed in parallel. These workflows are often interrelated with each other, e. g., tasks are executed in the same or in a similar way or a workflow partially influences tasks in another workflow. For example, multiple recipes in a kitchen scenario can be represented as workflows. These recipes often have dependencies or relations among each other, e. g., some dish is recommended as a starter for some other main dish. This knowledge should consequently be used to improve the selection of workflows for execution. For instance, the exemplary relation can help for recommending pairs of starters and main dish.

Copyright © 2022 for this paper by its authors.

A discipline from artificial intelligence that can be used to support workflow and knowledge management is *Process-Oriented Case-Based Reasoning (POCBR)* (Minor, Montani, and Recio-García 2014; Bergmann and Gil 2014). POCBR combines *Case-Based Reasoning (CBR)* (Aamodt and Plaza 1994) with *Process-Aware Information Systems (PAISs)* such as WfMSs (Dumas et al. 2018) and enables the reuse of best-practice procedural experiential knowledge, i. e., cases represented as a semantic workflow graphs. A key feature of (PO-)CBR is similarity-based retrieval, where problems and their respective solutions – bundled as *cases* in a *case base* – are used to solve upcoming problems (*queries*). This process relies on similarity computations that are harnessed to find the best-matching case w. r. t. a given user query. However, these similarity computations do not sufficiently consider dependencies between workflows (cases) in existing POCBR applications which restricts the utility of the similarity-based retrieval and limits the reuse of experiential knowledge in certain scenarios.

In this paper, we introduce so-called *inter-case dependencies* between different workflow cases. Such a dependency, similar to (Sell et al. 2009), describes a relationship between two entities that can either be a workflow or an element of a workflow. Each dependency has specific features that describe the characteristics of the relationship between the workflows or workflow elements. As a first step to consider inter-case dependencies in POCBR, we present a generic approach that allows to define dependencies within the query definition and the cases of the case base. We derive requirements from literature and investigate which knowledge structures are suitable for the representation of inter-case dependencies based on the requirements. Additionally, our approach covers a novel way of similarity assessment where the defined dependency information is used to find matching cases regarding a query.

The remainder of the paper is structured as follows: In Sect. 2, we present foundations on our semantic graph format and discuss related literature on existing definitions of dependencies in CBR and *Business Process Management (BPM)*. We present our approach for similarity-based retrieval in CBR by considering inter-case dependencies in Sect. 3. The proposed approach is evaluated by using an application scenario from the cooking domain (see Sect. 4). In Sect. 5, the paper is concluded and future work is discussed.

## 2 Foundations

The integration of dependencies in our approach concerns the case base and the queries that are used during a similarity-based retrieval. The cases and the query are both represented in a semantic workflow representation that is described in this section. Furthermore, we discuss related work on integrating dependencies from the research fields of *Case-Based Reasoning (CBR)* and *Business Process Management (BPM)*.

### Semantic Workflow Representation

The approach of representing cases in *Process-Oriented Case-Based Reasoning (POCBR)* considered within the scope of this contribution is that of *NEST* graphs, which are specific representations of a workflow or process (Bergmann et al. 2019). A *NEST* graph is a semantically labeled directed graph (Bergmann and Gil 2014) represented as a quadruple $W = (N, E, S, T)$, whereby $N$ is a set of nodes and $E \subseteq N \times N$ a set of edges. $T : N \cup E \rightarrow \Omega$ assigns a type from $\Omega$ to each node and each edge. $S : N \cup E \rightarrow \Sigma$ assigns a semantic description from a semantic metadata language $\Sigma$ to each node and edge and annotates the graph structure with semantic knowledge. Each NEST graph is part of a case base which is denoted as $W \in CB$. An exemplary NEST graph that represents a cooking recipe is shown in Fig. 1. The figure shows different node and edge types and an exemplary semantic description.

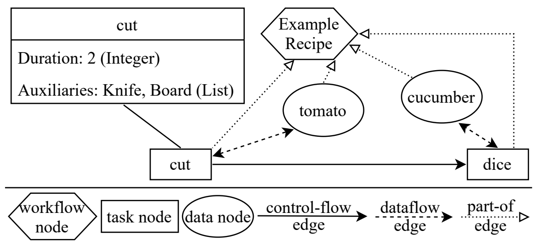

Figure 1: Exemplary Cooking Recipe represented as *NEST* Graph

### Related Work on Workflow Dependencies

Besides CBR literature, related work can also be found in the research area of BPM. That is due to the focus of BPM on *Process-Aware Information Systems (PAISs)*, which overlaps with the focus of POCBR. We present approaches that deal with the application of dependencies, as well as approaches that discuss different representation formats. Weber et al. (2006) address the challenge of case-based maintenance in conversational CBR systems. They build up dependencies between cases of the case base during the progression of a conversation in order to find cases that can be merged. Reuss et al. (2017) use dependencies to represent relationships between different types of knowledge in CBR. This aims at a different application scope than the definition of dependencies between cases. There is also work in CBR regarding the representation of dependencies. A *Memory Organization Packet (MOP)* (Kolodner 1993, pp. 108) is a type of dynamic memory that describes a generalized episode. Organizing dependent cases in a MOP, however, restricts the representation to a single dependency between two cases. Müller and Bergmann (2014) use clusters of cases that are created prior to a retrieval. Although the clustering is originally performed according to the similarity assessment, this could be applied to case dependencies such that dependent cases are organized in clusters. Furthermore, the case retrieval net introduced by Lenz and Burkhard (1996) is a graph structure consisting of nodes (cases) connected by an edge according to their degree of mutual influence. Representing dependencies with case retrieval nets has not yet been explored in the context of POCBR. The representation of dependencies is also discussed in the research area of BPM. Sell et al. (2009) discuss requirements and suitable representations for dependencies between different components of processes. The candidate representations are an OWL ontology and an XML document. Dumas et al. (2018, pp. 428) use a graph format for representing dependencies in the context of automated process discovery based on event data. Thereby, each node represents an event (e. g., a task) from the process and the edges model relationships that indicate the order of events. However, dependencies are only modeled as a temporal observation rather than on a semantic level.

## 3 Inter-Case Dependencies in Similarity-Based Retrieval

The integration of inter-case dependencies into POCBR requires several extensions. In order to model dependencies between cases of the case base, a suitable representation has to be defined. We raise requirements on the representation format and present a suitable format for our purpose, both assisted by analyzing existing literature. Given the dependency representation format, the regular similarity-based retrieval of POCBR is extended to support dependencies in the query definition, the case definition, and the similarity assessment.

### Requirements of Dependencies

The following requirements are imposed on the representation format for integrating dependencies into POCBR applications. They are identified by conducting a literature study regarding applications and use cases of POCBR and *Business Process Management (BPM)*. In particular, several requirements are based on the work of Sell et al. (2009) who also raise requirements on a dependency representation format.

1. **Different types of dependencies** belonging to the

corresponding domain must be distinguished (cf. Sell et al. 2009).

2. Two cases can have **multiple dependencies between each other**. These dependencies can be of different types (among others, cf. Sell et al. 2009).
3. Dependencies can be modeled at **different levels of granularity**. For instance, a complete case can have a relation to another case in the case base. In addition, it is also possible that only parts of a workflow case are related to parts of another workflow case, i. e., one node is dependent on a node from a different workflow.
4. In order to simplify the dependency specification, presentation, and integration in legacy systems, dependencies should be represented using a **well-known and standardized modeling format** (cf. Sell et al. 2009).
5. To allow a seamless integration of dependencies into a POCBR application, it should be possible to add the dependencies **without changes to the existing case representation**. The dependency representation format is intended as a separate component built on top of an existing case base, which ensures that it is applicable to different case representation formats (e. g., XML or relational databases).
6. **Automatically generated dependencies** (e. g., with machine learning methods) should be represented analogously to manually-modeled dependencies by an expert. The representation format should therefore be interpretable by machines as well as understandable and easily readable by humans.

## Knowledge Structure for Representation of Dependencies

Based on a literature study, two candidate dependency representation formats are identified and further described in the following: an OWL-based ontology (Sell et al. 2009) and a dependency graph based on a meta-XML (Dumas et al. 2018). Sell et al. represent Cases as individuals in an OWL ontology. Dependencies can either be modeled as properties or as instances (see Sell et al. 2009 for more details). Different types of dependencies can be modeled in a class structure. However, modeling dependencies in an OWL ontology requires the integration of the case base with the ontology in order to take full advantage of some ontology-specific features such as automatic reasoning. The dependency graph proposed by Dumas et al. is based on a meta-XML. It is very generic and can be configured to individual needs. A node represents a reference to a case from the case base, and an edge between these nodes specifies a dependency. There is a separate edge for each type of dependency and additional meta-information that is required for some types. Both approaches are suitable for representing dependencies in POCBR and meet most of the raised requirements. An exception is the fifth requirement that is not fully met by the ontology-based approach, as it might require a transformation of the cases and their case representation into an ontological representation. This results in a large effort for existing application domain models. Common *Case-Based Reasoning (CBR)* frameworks like ProCAKE (Bergmann et al. 2019), jColibri (Bello-Tomás, González-Calero, and Díaz-Agudo 2004), myCBR (Bach et al. 2014), and eXiT*CBR (López et al. 2011) currently do not support case representations based on ontologies. The dependency graph is a more generic representation for any kind of case representation (e. g., XML or relational database). Therefore, the dependency graph is eventually chosen.

## Definition of Inter-Case Dependencies

The dependencies between components, i. e., either a case or a workflow fragment, are represented by a dependency graph[1] $G = (N_d, E_d, l_G)$. Here, $N_d \in CB \cup N \cup E$ represents the nodes of the dependency graphs that can be a workflow from the case base or a workflow fragment, i. e., a node or an edge. $E_d \subseteq N_d \times N_d$ is the edge set. An edge from $E_d$ represents a dependency between two components where the function $l_G : E_d \to D$ labels the edge with a dependency type (which can be domain dependent) from the set of types $D$.

## Dependency-Guided Retrieval

In CBR, retrieval is important because it acts as the initial phase of the CBR cycle and has a strong influence on all subsequent phases (Aamodt and Plaza 1994). Thus, it is important to properly integrate inter-case dependencies into the similarity assessment of the retrieval phase. Our approach, which takes dependencies into account during query creation and during the similarity assessment, is called *Dependency-Guided Retrieval (DGR)*. It can be divided into three parts: First, the query is defined, which contains several traditional queries and the knowledge about dependencies between them. In the second step, the case base is searched for candidate cases that fulfill the dependency requirements defined in the query. This step can be seen as a preprocessing step prior to the semantic similarity assessment. Finally, the similarity between the query and the candidate cases is computed in order to find the best-fitting retrieval results.

The **query definition** in a DGR (see Fig. 2, item 1) is an extension of conventional retrieval queries (Bergmann 2003, pp. 188). A *dependency query* $Q = (Q_Q, Q_D)$ is a tuple of $m$ conventional queries $Q_Q$ represented as NEST graphs and the dependencies between those queries $Q_D$. $Q_D$ is a dependency graph itself, containing all dependency knowledge among the query components in $Q_Q$.

Since the newly defined query contains several traditional queries, a simple similarity computation between a single case and the query is not applicable and, thus,

[1] The dependency graph is a pseudodigraph that allows multiple edges between two nodes.

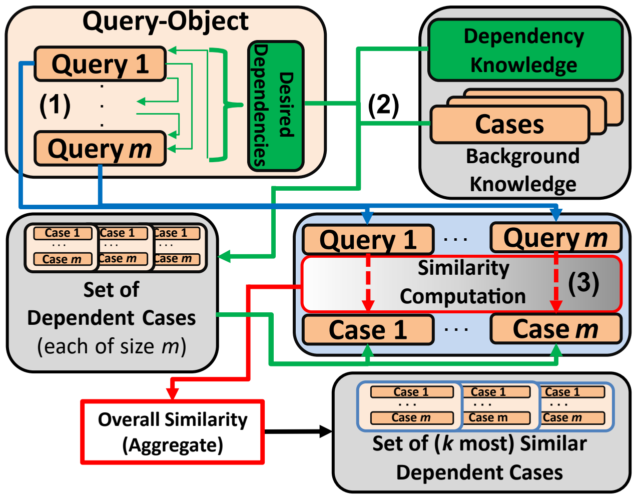

Figure 2: Process of Dependency-Guided Retrieval

the cases must be converted into a compatible structure. Therefore, a set of **dependent cases** $DC$ of size $m$ is created. This set is computed by performing a search in the case base (see Fig. 2, item 2). Each entry in $DC$ is a tuple $(dc, sim_D)$, consisting of a dependent case $dc = (c_1, ..., c_m)$ that contains several cases from the case base and a similarity of the dependency $sim_D$. Algorithm 1 illustrates the search process. The dependency query, the case base, and the dependency graph are given as input. We generally differentiate between *full matches* and *partial matches* of dependent cases regarding the query $Q$. A full match (see loop in lines $3-8$) exists if a dependent case contains exactly the required dependencies from $Q_D$. This means that, for instance, a dependency which points from $q_i$ to $q_j$ of type $d \in D$ must be mapped to a dependency of the same type $d$, which points from $c_i$ to $c_j$. The minimum number of full matches to retrieve can be set by the parameter $ms$. $DC$ is populated by full matches with priority. If there are less than $ms$ full matches, partial matches are also considered. A partial match (see loop in lines $10-14$) exists if a dependent case contains at least as many dependencies as the query, but not all of them are of the required types. For this purpose, dependency similarities can be defined in the similarity model, which indicate how similar dependencies of different types $d \in D$ are. The dependency similarity of a full match is 1 and the one of a partial match is usually below 1. For a partial match, the similarity value is computed by the function *calcMappingMaxSim* that matches the dependency graph of the query onto the dependent case, similar to similarity assessment via graph matching (Bergmann and Gil 2014). This can be done using a brute force approach but heuristic search algorithms can also be used.

After determining the set $DC$, the final **similarity calculation between the case tuples and the dependency query** $Q$ (see Fig. 2, item 3) is performed.

**Algorithm 1** Algorithm to calculate a set of dependent cases with its dependency similarities

**Input:** dependency query $Q = (Q_Q, Q_D)$, case base $CB$, dependency graph of the case base $G$, required minimum size of the set of dependent cases $ms$
**Output:** Set of dependent cases $DC$

```
1:  $DC \leftarrow \emptyset$
2:  $D_Q \leftarrow getDependencies(Q_G, Q_Q)$
3:  for all $(c_1, ..., c_m) \in CB^m$ do
4:      $D_{DC} \leftarrow getDependencies(DG, (c_1, ..., c_m))$
5:      if $isFullmatch(D_Q, D_{DC})$ then
6:          $DC.add(((c_1, ..., c_m), 1.0))$
7:      end if
8:  end for
9:  if $|DC| \leq ms$ then
10:     for all $(c_1, ..., c_m) \in (CB^m \setminus DC)$ do
11:         $D_{DC} \leftarrow getDependencies(DG, (c_1, ..., c_m))$
12:         $sim_D \leftarrow calcMappingMaxSim(D_Q, D_{DC})$
13:         $DC.add(((c_1, ..., c_m), sim_D))$
14:     end for
15: end if
```

The similarity function for the query $Q_Q = (q_1, ..., q_m)$ and for each dependent case $dc = (c_1, ..., c_m) \in DC$ is defined as follows:

$$sim(Q_Q, dc) = \alpha \cdot sim_D(Q_Q, dc) + (1-\alpha) \cdot \sum_{i=1}^{m} sim(q_i, c_i)$$

The dependency similarity of the dependency query and the dependent case $(sim_D(Q_Q, dc))$ is first calculated. If the found case combination is a full match, then $sim_D$ is 1.0. This value is then added to the aggregated similarities between the items of the query and the items of the dependent case $(sim(q_i, c_i))$. $\Sigma$ represents an aggregation function of the individual similarities, e. g., mean, max, min etc. The similarity values of these items are computed with the graph-matching similarity measure presented by Bergmann and Gil (2014). Overall, this results in a similarity value that assesses the dependencies in the query and the cases and the similarities of individual cases. We also include the weighting factor $\alpha$ that enables adjusting both parts of the similarity computation to individual needs of the use case.

## 4 Experimental Evaluation

In our evaluation, we want to examine the suitability of the *Dependency-Guided Retrieval (DGR)* approach regarding the retrieval quality and modeling time in a user study. For this purpose, a usability test of a prototypical implementation of this approach is performed as a comparison to the traditional *Case-Based Reasoning (CBR)* approach. The three usability components effectiveness, efficiency, and satisfaction according to *ISO 9241-11* (The International Organization for Standardization 2018) are determined. The evaluation is performed using the domain of cooking recipes that is in-

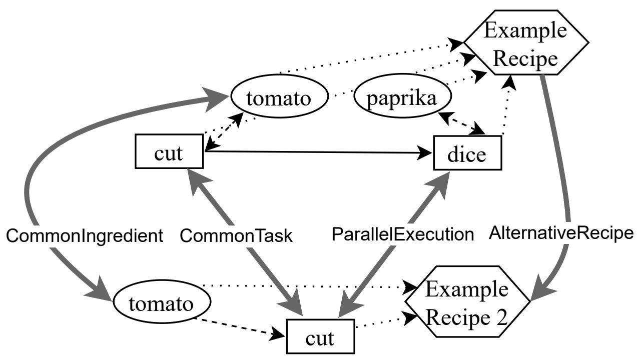

Figure 3: A simple example of dependencies between two workflows represented as *NEST* graphs (Analogous to the representation in Fig. 1)

troduced together with the experimental setup. Afterwards, the results are presented.

## Application Scenario and Experimental Setup

In related work, e. g., Müller and Bergmann (2015), *Process-Oriented Case-Based Reasoning (POCBR)* is applied to the cooking domain where recipes are modeled as semantic graphs. We use this domain, extended by dependencies, as an application scenario of our evaluation (see Fig. 3). Consider the following scenario: A person wants to prepare several dishes and needs corresponding recipes: a starter recipe and a main dish recipe. In addition, several cooking steps should be performed in parallel to save time during cooking. To perform a retrieval with this exemplary query, the integration of dependencies in the retrieval process and the case base is required. This includes an inter-case dependency that represents combinable starter and main dish recipes, as well as dependencies between cooking steps that specify them as executable in parallel. Besides these two, there are many other dependencies that can be applied in the given domain, e. g., an inter-case dependency for the selection of alternative recipes or a dependency between common tasks in different recipes (see Fig. 3).

For the experiments, the presented approach was implemented in the ProCAKE framework (Bergmann et al. 2019) that enables the development of process-oriented CBR applications. Cases are represented as semantic *NEST* graphs (see Sect. 2). A case base with 20 suitable recipes of main and starter dishes has been created and is used as the test domain. The dependencies already described in the application scenario are specified in the case base. We asked experts for the evaluation who already had knowledge in CBR as well as in the use of ProCAKE. The experts are divided into two groups: the first group used ProCAKE without the new approach but with externally provided knowledge about dependencies in tabular form. The second group used ProCAKE with DGR and a given dependency graph containing all required dependency knowledge. There are a total of eight experts that were randomly assigned to the groups. After a short introduction to the topic, we present two retrieval scenarios for the evaluation. In both scenarios, two queries are given, for which the best possible case pair has to be found w. r. t. the given dependencies. In addition, the second group that used our proposed approach also had to add dependencies to the dependency graph. Finally, the experts were interviewed by means of questionnaires using Likert scales. In addition to specific questions about the proposed approach, general questions such as the level of knowledge in CBR have been asked. The questionnaire is used for assessing user satisfaction. Effectiveness is calculated as the product of quantity and quality. Since all experts were able to complete the assignments and found the best possible solutions, quantity and quality are always fully satisfied, i. e., 1, and in turn also the effectiveness. Therefore, it is omitted below.

## Experimental Results

Efficiency is determined by measuring the times required for the assignments. Overall, some outliers were observed, as some experts solved assignments significantly faster or took significantly longer. The first group using CBR without dependency integration took an average of approx. 31 minutes for both retrieval tasks. In the second group that used DGR, it took an average of approx. 19 minutes for both tasks. However, the execution of the retrieval leads to an increase in time of approx. 10 seconds compared to the traditional retrieval, in which only a few milliseconds are needed. It can be seen that the group using the new approach required less training time and was thus able to complete the first task significantly faster. In addition, the experts needed less time to model the dependencies in the query compared to the time needed to manually compare two cases in the other group.

When measuring satisfaction using the questionnaire, all experts indicated that their knowledge of CBR is basic or advanced. In the group that did not use our proposed approach, the experts were all satisfied with their found results and their quality. The representation was judged to be appropriate for the scale of the task. However, opinions differ widely whether this manually created representation is appropriate for larger case bases. Nevertheless, experts agree that representing dependencies in a separate knowledge structure in CBR is useful and also suggest to automatically include them during the retrieval. Half of the experts also formulated suggestions for improvements such as a machine-readable representation and storage of dependencies, as well as queries that contain dependency information. These recommendations can be found in this form in our proposed approach. The second group, which used the new approach of DGR, determined that modeling the dependency graph in the current form is appropriate and preferred to store it as XML. This question was asked to learn about user preferences because, as described, both

XML and an ontology are appropriate representations of the dependency graph. When asked if an ontology would be better suited for storage, half of the experts disagreed, and only one agreed. The approach of DGR was also evaluated positively by the experts. However, the experts were unsure whether their results are optimal, which is probably due to the low consideration of the individual retrieval results. Experts rated the integration of dependencies in the query as a good option and did not find it too complex or incomprehensible. They all considered query construction with multiple problem descriptions to be a good and useful extension of conventional CBR systems. Overall, the experts were able to retrieve the same qualitative results in less time with the new approach of DGR.

## 5 Conclusions and Future Work

This paper presents an approach for integrating dependency knowledge into the retrieval phase of *Case-Based Reasoning (CBR)*. The dependency graph has been identified as a useful knowledge structure for inter-case dependencies compared to an ontology-based approach. In order to include dependencies in the retrieval phase, changes to the retrieval process with the underlying query definition and similarity measures are proposed. The concept of *Dependency-Guided Retrieval (DGR)* has been tested with respect to suitability and accessibility in comparison to traditional CBR by means of a user evaluation and a usability test. The experts that used the dependency-guided approach considered it as suitable for solving the given problem and considered it as a useful extension of traditional CBR systems.

In future work, we plan to optimize the search procedure for the $k$-best case combinations that uses a slow exhaustive search. Thus, approaches from the field of informed search (see Hart et al. 1968) could be explored to achieve a fast computation of the best case combinations. Furthermore, dependencies are currently manually modeled by human experts. We want to examine the use of inductive learning methods to automatically learn dependencies between cases and workflow fragments based on the vocabulary and the case base of the CBR application. In addition, the integration of dependencies into other phases of the CBR cycle (see Aamodt and Plaza 1994) offers potential for future work. Our approach can enhance existing workflow adaptation methods by exploiting dependency knowledge for modifying the solution of the retrieved case in the *reuse* phase.

## References

Aamodt, A., and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Commun.* 7(1):39–59.

Bach, K.; Sauer, C.; Althoff, K.-D.; and Roth-Berghofer, T. 2014. Knowledge Modeling with the Open Source Tool myCBR. In *CEUR Workshop Proc.*, volume 1289.

Bello-Tomás, J. J.; González-Calero, P. A.; and Díaz-Agudo, B. 2004. JColibri: An Object-Oriented Framework for Building CBR Systems. In *7th ECCBR*, volume 3155 of *LNCS*, 32–46. Springer.

Bergmann, R., and Gil, Y. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* 40:115–127.

Bergmann, R.; Grumbach, L.; Malburg, L.; and Zeyen, C. 2019. ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In *27th ICCBR Workshop Proc.*

Bergmann, R. 2003. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*, volume 2432 of *LNCS*. Springer.

Dumas, M.; La Rosa, M.; Mendling, J.; and Reijers, H. A. 2018. *Fundamentals of Business Process Management*. Springer.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 4(2):100–107.

Kolodner, J. L. 1993. *Case-Based Reasoning*. Morgan Kaufmann.

Lenz, M., and Burkhard, H.-D. 1996. Case retrieval nets: Basic ideas and extensions. In *KI-96: Adv. in AI*, 227–239. Springer.

López, B.; Pous, C.; Gay, P.; Pla, A.; Sanz, J.; and Brunet, J. 2011. eXiT*CBR: A framework for case-based medical diagnosis development and experimentation. *Artif. Intell. Medicine* 51(2):81–91.

Minor, M.; Montani, S.; and Recio-García, J. A. 2014. Process-oriented case-based reasoning. *Inf. Syst.* 40:103–105.

Müller, G., and Bergmann, R. 2014. A Cluster-Based Approach to Improve Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In *21st ECAI*, volume 263, 639–644. IOS Press.

Müller, G., and Bergmann, R. 2015. POQL: New Query Language for Process-Oriented Case-Based Reasoning. In *Proc. LWA*, volume 1458 of *CEUR Workshop Proc.*, 247–255.

Reuss, P.; Witzke, C.; and Althoff, K.-D. 2017. Dependency Modeling for Knowledge Maintenance in Distributed CBR Systems. In *25th ICCBR*, volume 10339 of *LNCS*, 302–314. Springer.

Sell, C.; Winkler, M.; Springer, T.; and Schill, A. 2009. Two Dependency Modeling Approaches for Business Process Adaptation. In *3rd KSEM*, volume 5914 of *LNCS*, 418–429. Springer.

The International Organization for Standardization. 2018. Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts.

Weber, B.; Reichert, M.; and Wild, W. 2006. Case-Base Maintenance for CCBR-Based Process Evolution. In *8th ECCBR*, volume 4106 of *LNCS*, 106–120. Springer.