Challenges and Support Potentials in the Development of Case-Based Reasoning Applications*

Lisa Grewenig[1], Christian Zeyen[1], Alexander Schultheis[1,2], Lukas Malburg[1,2], and Ralph Bergmann[1,2]

[1] German Research Center for Artificial Intelligence (DFKI), Branch Trier, Behringstraße 21, 54296 Trier, Germany
{firstname.lastname}@dfki.de
[2] Artificial Intelligence and Intelligent Information Systems, Trier University, 54296 Trier, Germany, https://www.wi2.uni-trier.de
{Alexander.Schultheis,malburgl,bergmann}@uni-trier.de

Abstract Developing *Case-Based Reasoning* (CBR) applications is a knowledge-intensive task that consists of numerous demanding and time-consuming development steps. Major challenges are posed by knowledge engineering since CBR systems depend on capturing, validating, and maintaining domain knowledge in an explicit, machine-interpretable form. This usually can be done only in a close interdisciplinary collaboration between CBR and domain experts. To support development, previous works put forth a broad spectrum of support functions but mostly have not yet addressed the synergies of support functions or the potentials for involving domain experts, particularly those without knowledge-engineering experience. To further improve the accessibility of CBR frameworks and to reduce the effort of development, this paper consolidates development challenges along with corresponding support methods. In addition, we address the potentials for involving domain experts in the development with the help of support functions. We derived development challenges and support potentials from literature and conducted an explorative survey in the CBR research community.

Keywords: Case-Based Reasoning · CBR Applications · Knowledge Engineering · Domain Experts · Development Support

1 Introduction

Developing CBR applications requires both knowledge engineering and specific CBR-related development steps. Regarding knowledge engineering, one has to elaborate, populate, validate, and maintain the so-called knowledge containers [31], i.e., vocabulary, case base, similarity model, and adaptation knowledge.

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-032-33865-5_36

CBR-related development steps comprise application- and domain-dependent design decisions and their implementation to ensure effective and efficient knowledge acquisition, user interfaces, retrieval, adaptation, maintenance, and deployment. Domain experts play a key role in the development process but are usually inexperienced in knowledge engineering. Ideally, they can express their knowledge like knowledge engineers in terms of formal attributes, categories, and relationships rather than narrative descriptions or holistic judgments. With the rapid development of *Generative AI* (GenAI) methods and, particularly *Large Language Models* (LLMs), new support potentials for various aspects such as knowledge acquisition, maintainability, and explainability are opening up [3]. The CBR research community regularly presents new approaches to support the development of CBR applications. These efforts are also reflected in the ongoing development of CBR frameworks [37]. However, related works have not yet addressed the synergies of support functions and potentials for involving domain experts.

The objective of this paper is to form a basis to further improve the accessibility of CBR frameworks or specific applications and to reduce the effort and, consequently, the costs for development in future work. To this end, we investigate the challenges of typically required development steps along with the potentials for support methods. The contribution of this work is (1) a literature-based synthesis of existing support for development steps in CBR, (2) a survey-based assessment of CBR experts on involvement and support of domain experts regarding the identified development steps, and (3) a synthesis of challenges in each development step along with potentials for support methods.

In the remainder of this paper, related work on existing support for developing CBR applications is presented in Section 2. Section 3 presents the results of an explorative survey conducted in the CBR research community with a focus on involving domain experts. Section 4 describes the derived challenges and support potentials for each identified development step. Finally, Section 5 summarizes the main findings, limitations, and proposes future work.

2 Existing Support for Developing CBR Applications

This section investigates related work along the development steps that we derived from CBR literature. Fig. 1 illustrates how the development steps can be mapped to the traditional CBR model, i.e., knowledge containers [31] and CBR cycle [1]. Please note that the steps are organized in an ideal-typical sequence. In practice, it can happen that steps are skipped or that there are iterations. In complement to this work, the INRECA methodology [9] has been proposed to provide practical guidelines from collected CBR development experiences, represented as software process models. In [37], we provided an overview[3] of the features of existing CBR frameworks.

[3] Please refer to `https://git.opendfki.de/easy/overview-and-comparison-of-cbr-frameworks/-/raw/main/Overview-Table.pdf` for an overview table (from 2023).

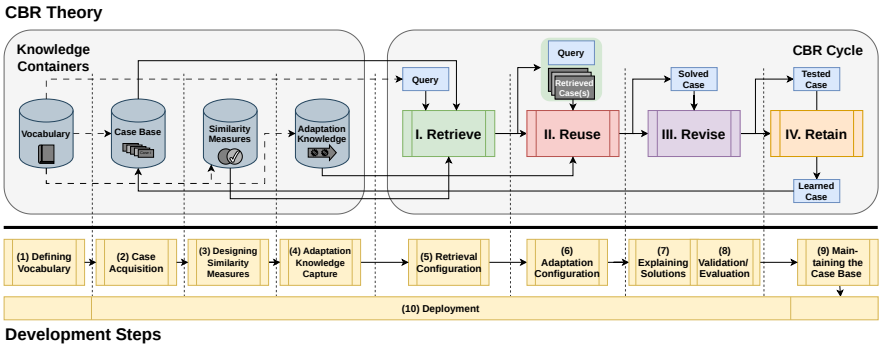

Figure 1. Mapping of Elements of the CBR Model to Development Steps

Defining Vocabulary is supported in the myCBR framework through interactive knowledge modeling. It also provides extensions such as integrating linked open data for taxonomy construction [6,37,40]. The ProCAKE framework provides a GUI-based object editor for system and custom data classes and automatic derivation of data models from input data [10,37].

Case Acquisition is supported by ProCAKE through visual case editing for structural and process-oriented cases and automated imports from CSV and JSON [10,37]. myCBR also supports importing content from unstructured documents into an existing vocabulary [6]. The proposed competence modeling method makes coverage and competence regions visible to guide case acquisition and support maintainers in targeting under-covered regions [39].

Designing Similarity Measures is frequently addressed by related work with a focus on the design, tuning, and explanation of similarity measures. myCBR supports interactive similarity modeling and testing through a workbench with integrated execution support and immediate feedback [37,40]. Bach and Mork [5] provide visualizations of global similarity and local contributions. Marín-Veites and Bach [24] add ranking summaries and per-case-pair views with global similarity, local similarities, and weighted contributions. For process-oriented CBR, visualization-based similarity explanations provide mapping-oriented graph views and heatmaps [36]. Building on this, point-mapping methods are investigated as visualization techniques for similarities in temporal CBR [34]. The tool SimViz provides coordinated views for case comparison and similarity distributions, including taxonomy-based inspection for semantic similarities, and supports identifying modeling errors and data quality issues [17,18]. A combination of SimViz with the CBR framework CBRkit [21] supports iterative similarity design with immediate feedback, partial interactive reconfiguration, and recomputation-based inspection, and explicitly targets CBR designers rather than end users [19].

Adaptation Knowledge Capture is less supported by existing CBR frameworks [37]. Frameworks mainly provide implementations of adaptation mecha-

nisms. myCBR 3 supports rule-based completion and adaptation through integration with a rule engine [2]. Several approaches for learning adaptation knowledge exist in research (e.g., [12,13]), but no concrete support function for accessibility of such approaches is provided. ProCAKE includes an integrated rule engine that enhances the representation, maintainability, and runtime behavior of adaptation rules [23].

Retrieval Configuration is supported by workflow-oriented tooling regarding retrieval strategy configuration and evaluation through guided steps and visual inspection, including scatter plots and clustering [15].

Adaptation Configuration is also less supported by existing CBR frameworks [37]. To optimize adaptation chains, Hotz et al. [16] investigate search algorithms for determining optimal sequences in rule-based adaptation.

Explaining Solutions is an important aspect of *Explainable CBR* (XCBR) where dedicated tools support transparency and communication [35]. The similarity visualizations described above can also be used for user explanation. The myCBR tool provides explanation support for understanding retrieval outcomes during development, and myCBR reports provenance information in concept explanations [2,7]. For end-user-facing explanation in the medical domain, Lamy et al. propose a visual CBR interface combining a polar multidimensional scaling scatter plot with rainbow boxes [20].

Validation and Evaluation is supported by the eXiT*CBR framework through an executable environment with a GUI for navigating experiments and comparing alternative configurations through visual analytics such as receiver operating characteristic curves and cost curves [22]. It also supports experiment replication by managing and backing up experiment artifacts. Workflow-oriented tooling further supports retrieval strategy configuration and evaluation through guided steps and visual inspection [15].

Maintaining Knowledge is supported by eXiT*CBR through human approval workflows for adding cases and instance selection techniques for pruning [22]. The framework CloodCBR supports retain-side functions such as forgetting cases and recomputing similarities [37]. The tool SimViz supports maintenance-oriented inspection [17,18]. Competence modeling makes coverage and competence regions visible to guide case acquisition and maintenance decisions [39]. Interactive visualization supports redundancy and outlier detection and makes case base evolution transparent [25].

Deployment is supported by the CloodCBR framework, which exposes CBR functionality through a microservice-oriented REST architecture [26,37]. myCBR provides a REST interface for modeling and runtime functionality [4]. ProCAKE exposes its functionality via REST and provides a Docker-based setup, facilitating its use as a service in external applications. CBRkit supports embedding and deployment through a REST interface, a command-line interface, and a Docker image [21].

General Support Functions span multiple development steps. Frameworks typically provide support by dedicated GUI, API, and deployment-oriented infrastructure [37]. jColibri provides guided configuration and generation, includ-

ing interactive testing of partially configured systems and generation of deployable applications such as standalone CBR systems and web interfaces [14,37]. Within the COLIBRI ecosystem, system design itself is treated as a CBR problem by retrieving and recommending templates based on project requirements [27]. Semantic templates add a graphical template editor, multi-level template libraries, and semantic annotations to compute template similarity and guide adaptation under dependency constraints [28]. COLIBRI Studio stores architectures as templates, tries them as executable implementations, and adapts them through component selection [30,33]. The COLIBRI Open Platform adds publishing support and a central repository for data, executable systems, and experimental protocols, thereby strengthening reuse and reproducibility of complete designs [29]. myCBR further supports rapid prototyping and provides practical modeling guidance and documented workflows in accompanying documentation rather than tool functionality [6,40]. CloodCBR adds dashboard-based inspection and transparency, including parallel plots for reviewing outcomes and supporting revision through interactive analysis [26].

3 Explorative Survey in the CBR Research Community

To ensure that important development steps and support methods are not missed and to gather practical insights from the CBR research community regarding the involvement of domain experts, we conducted an explorative survey [42].

Setup - In three internal workshops of the authors, the goal and purpose of the survey have been determined, and we iteratively refined the set of questions, items, and measurement scales. The survey consists of seven questions, which are divided into four categories: (1) Background of the participant, (2) rating of the difficulty of development steps for domain experts, (3) rating of the utility of support functions for domain experts, and (4) the experience of the participant in developing CBR applications in collaboration with domain experts. The development steps and support functions have been derived from the literature, complemented with our experience in developing CBR applications. We used the mailing list of the *International Conference on Case-Based Reasoning* (ICCBR) to invite all subscribers to participate in our survey. The survey ran for two weeks in Q1 2026 and we received 22 responses. In the following, we summarize the results.

(1) Background of the Participants - Fig. 2 depicts the profile of survey participants. The survey mainly reflects the viewpoints of academic researchers (16 out of 22). Most responses came from people with more than three years of experience with CBR (16 out of 22) and most people have practical experience in developing CBR applications together with domain experts in the past (15 out of 22).

(2) Rating of the Difficulty of Development Steps for Domain Experts - In this category, we asked participants to rate the difficulty of development

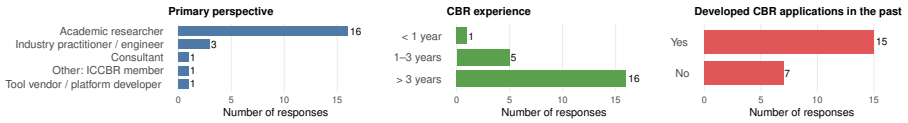

Figure 2. Profile of Survey Participants.

steps for domain experts without knowledge engineering experience on a 5-point Likert scale (1 – not difficult to 5 – extremely difficult). Participants also had the option to name one additional development step along with a rating. The phrasing of the question and development steps, along with the participants ratings, are depicted in Fig. 3. The diagram shows aggregated percentages for

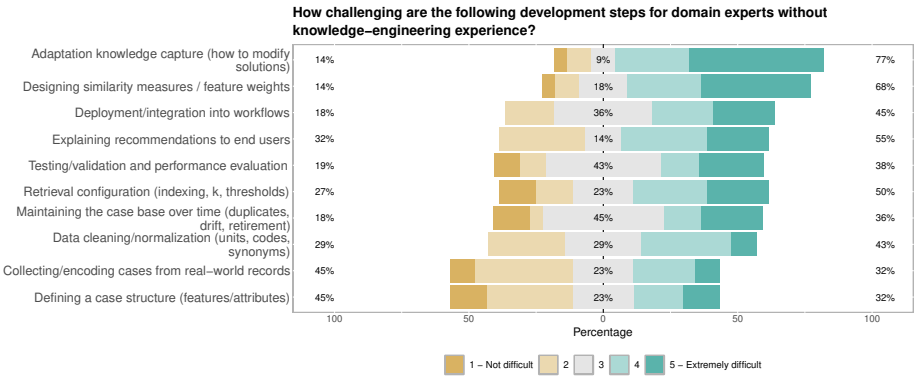

Figure 3. Difficulty Rating of Development Steps.

positive, neutral, and negative assessments. The items are sorted by the average Likert scores in descending order. The steps *capturing adaptation knowledge* and *designing similarity measures* stand out with an assessment of mostly being extremely difficult. Regarding the other end of the scale, the picture is not so clear. There is a tendency that *defining case structures* and *collecting cases* are rather rated as less or not difficult. There is not a single task that was not rated as extremely difficult by at least one participant. Other development steps along with a difficulty rating (in parentheses if available) specified by the participants are *fixing incompatibilities in the knowledge base* (4), *unit testing* (3), *training domain experts* (-), *deployment and support in an industrial context* (4), *licensing* (5), and *building a deep learning CBR* (5).

(3) Rating of the Utility of Support Functions for Domain Experts - In this category, we asked participants to rate the utility of support functions for domain experts on a 5-point Likert scale (1 - not useful to 5 - extremely

useful). Participants also had the option to name one additional support function along with a rating. The phrasing of the question and support functions, along with the participants ratings, are depicted in Fig. 4 analogous to the previous diagram. One can see that there is a tendency for participants to rate all features towards (extremely) useful. The support by *interactive tuning with immediate*

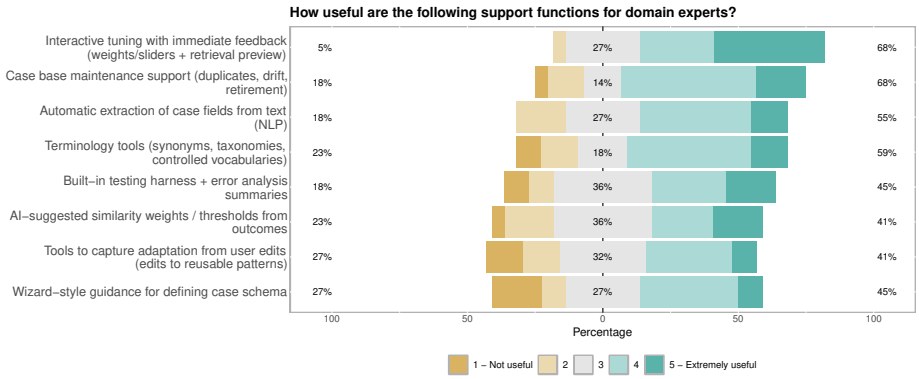

Figure 4. Utility Rating of Support Functions.

feedback stands out as being considered the most useful. Other support functions along with a utility rating specified by the participants are *automatic vocabulary extraction* (3) and *AI-based generation of an initial complete CBR application based on a data set and a given objective in natural language* (5).

(4) Experience of the Participants in Developing CBR Applications in Collaboration with Domain Experts - In this category, 15 out of 22 participants stated that they have such development experience. In this case, we asked follow-up questions concerning the used CBR tools, role of the domain experts, and the most challenging aspects of the collaboration. Participants stated that they used open-source CBR frameworks (six different frameworks in total), proprietary implementations, and their implementations. The most commonly mentioned roles of domain experts are *providing data and knowledge* (11 mentions), *modeling knowledge* (3 mentions), and *testing and evaluation* (5 mentions). The most challenging aspects of working with domain experts with at least two mentions are *finding a common vocabulary* (6 mentions) and *explaining the CBR paradigm and similarity* (3 mentions). It is noticeable that although the survey participants rated defining a case structure and collecting/encoding cases as least difficult for domain experts, it is mentioned as most challenging to find a common vocabulary when collaborating. The participants rated similarity modeling as the second most difficult challenge for domain experts, which is in line with the mentioned aspect that they need to understand the notion of similarity.

4 Development Challenges and Support Potentials

After analyzing the existing support functions proposed by related work and the practical insights gained through the survey results, we refined the development challenges and conceptualized corresponding support potentials. In the following section, we describe them for each development step.

4.1 Defining Vocabulary

A central challenge in the development of a CBR system is defining an appropriate case structure and vocabulary to describe cases [31]. This task requires experts to translate often tacit and experience-based domain knowledge into an explicit, structured representation that can be processed by a CBR system. In practice, this involves determining which attributes are relevant for describing a case, deciding on appropriate levels of abstraction, defining data types and value ranges, and ensuring that the resulting structure captures the aspects of a problem that are important for similarity assessment and retrieval. This step requires a combination of domain expertise and knowledge engineering skills. Consequently, this step typically requires interdisciplinary collaboration.

Schema Definition can be supported by analyzing existing data sources (e.g., CSV, JSON, Excel, databases, or unstructured text) representing past problem–solution situations and suggesting candidate attributes together with inferred data types, value ranges, units, and default values. E.g., automatic vocabulary extraction can help identifying relevant terms and relationships as a basis for further processing with terminology tools. Complementary acquisition interfaces, such as structured input forms, data import tools, or guided interview prompts (e.g., chatbots), can help transform real-world examples into well-formed cases while simultaneously refining the underlying case schema.

Terminology Management can support consistent use of domain terms when defining attributes and values, e.g., with visual editors. Such tools may provide mechanisms for managing synonyms, aliases, taxonomies, and controlled vocabularies [2], while automatic normalization harmonizes heterogeneous value representations (e.g., units or codes).

Consistency Checking - To reduce modeling errors, maintaining vocabulary and case structure as a single source of truth can ensure consistency across the case base and related knowledge artifacts. Changes made to the case schema should therefore be propagated automatically to other knowledge containers. This can be supported with visual editors that show the relationships between knowledge artifacts.

4.2 Case Acquisition

Case acquisition relies on past problem–solution situations that are typically contained in historical data sources or exist as implicit experiential knowledge. However, such experiences are rarely available as ready-to-use cases and must

instead be constructed from recorded information. Consequently, relevant information about problems, decisions, and outcomes is often embedded in narrative descriptions or distributed across multiple records. Identifying suitable cases therefore requires collecting and interpreting these records and transforming them according to the defined case structure into explicit cases.

Semi-Automatic Case Extraction can assist in identifying candidate cases within existing data. By detecting recurring problem–solution patterns or suggesting potential cases from historical data, these tools can reduce manual effort while keeping domain experts involved in validating and refining the extracted cases. Natural language processing (NLP) techniques can further support this process by automatically extracting relevant case fields from unstructured text and mapping them to elements of the case representation.

Interactive Case Acquisition – Adaptive input forms or conversational interfaces (e.g., chatbots) can support users in capturing problem situations and solutions while dynamically adapting inputs based on the case schema and previously entered information. Such environments can ensure that newly created cases conform to the defined structure [10] and can reveal gaps or inconsistencies in the schema or vocabulary, linking acquisition with data analysis in 4.1. In addition, elements of case base maintenance (see Section 4.9) can already be incorporated during acquisition by enforcing quality constraints, for example, by preventing the inclusion of incomplete or inconsistent cases.

Case Base Exploration – Visualization and analysis techniques can support case acquisition by enabling users to explore and interpret existing cases while assessing the coverage of the problem space. Adaptive visualizations based on the case schema can support clustering or filtering of cases according to user-defined criteria, helping users identify patterns and relationships. Complementary analysis methods, such as competence- or coverage-based approaches, can reveal underrepresented regions and boundary areas where additional cases are needed [39]. Based on these insights, systems can guide domain experts to contribute counterexamples or boundary cases that capture rare but critical scenarios, which is particularly important in safety-critical domains.

Support for case base maintenance may already be integrated into the acquisition process to ensure that newly created cases satisfy basic quality criteria and do not introduce avoidable inconsistencies or redundancies.

4.3 Designing Similarity Measures

Similarity assessment is central to CBR, as retrieval relies on identifying cases that are most similar to a given problem [1]. However, defining similarity is often non-trivial because the relative importance of different attributes may vary across domains and contexts [31]. Domain experts typically possess intuitive knowledge about which aspects of a case are more relevant for comparison, but translating this knowledge into formal similarity functions and numerical weights requires knowledge engineering expertise. Furthermore, different attribute types may require different similarity models, such as numerical distance functions, symbolic comparisons, or taxonomy-based similarities [8]. Poorly designed simi-

larity measures or inappropriate feature weights can significantly affect retrieval results and reduce the usefulness of the CBR system [39].

Configuration Recommendation can assist in configuring similarity models by suggesting both appropriate similarity functions and parameter settings, such as feature weights or thresholds. Based on attribute types (e.g., numeric, symbolic, hierarchical), such assistants can guide domain experts in selecting suitable similarity functions while also proposing reasonable default configurations. Machine learning or optimization techniques can further refine these suggestions by analyzing historical cases and retrieval outcomes to improve retrieval performance. In addition, such tools can highlight potential modeling issues and provide transparent explanations to help users understand and validate the proposed similarity model.

Interactive Similarity Tuning – Graphical editors can enable users to adjust feature weights and select similarity functions for individual attributes through sliders or parameter controls. Immediate previews of retrieval results allow users to observe how configuration changes affect case ranking. Such interactive feedback supports what-if analyses by showing how retrieval outcomes change when feature values or weights are modified, allowing experts to explore the sensitivity of the similarity model and identify attributes that dominate or have little influence on retrieval results (see Section 2).

Efficiency and Scalability Analysis – As similarity computations directly influence retrieval performance, tools can support domain experts in assessing the computational efficiency of similarity models. This may include estimating computational costs, identifying expensive similarity functions, or recommending optimized configurations that balance retrieval quality and runtime performance.

4.4 Adaptation Knowledge Capture

Adaptation knowledge denotes the knowledge required to transform the solution of a retrieved case into one that fits a new problem [1,31]. Although CBR frameworks may provide technical adaptation mechanisms [2,37], the underlying adaptation knowledge is usually not available explicitly. Instead, it has to be elicited from domain experts, formalized, and configured for use in the system. As a result, the acquisition and validation of adaptation knowledge typically require substantial effort.

Adaptation Capture – One promising support potential is the capture of adaptation knowledge from user edits. In such a setting, domain experts can modify proposed solutions directly, while the system records these changes and generalizes recurring edits into reusable adaptation patterns or candidate rules. Similarly, adaptation knowledge may be learned from pairs of original and adapted cases, reducing the effort of making such knowledge explicit. Machine learning methods can further support this process by identifying regularities across repeated corrections [13].

Visual Editing - Visual adaptation knowledge interfaces can lower the barrier for specifying rules, operators, or constraints by providing intuitive authoring environments instead of purely symbolic representations. Such environments may

also support the structured validation and refinement of adaptation knowledge in an interdisciplinary and collaborative fashion.

4.5 Retrieval Configuration

Retrieval determines which cases are considered relevant for a given problem specification and therefore directly influences the quality of the proposed solutions [1]. It may be challenging for users to translate a current problem into a formal query according to the defined case structure. Thus, it is important to provide suitable query interfaces. Designing an effective retrieval configuration involves selecting appropriate retrieval strategies, defining indexing structures, and tuning parameters such as the number of retrieved k-nearest neighbors, similarity thresholds, or ranking mechanisms. These decisions depend on domain-specific similarity notions, case base characteristics, and task-specific retrieval requirements. Poorly configured retrieval mechanisms may lead to irrelevant cases being retrieved or useful cases being overlooked.

Visual Retrieval Pipeline Configuration – Dedicated configuration tools can support designers in defining the retrieval workflow as a sequence of retrieval and filtering steps [21]. A graphical retrieval pipeline creator can allow users to compose and configure stages such as candidate selection, similarity-based ranking, and post-filtering. Such tools help make the retrieval process explicit and easier to adjust, especially in systems with multi-stage retrieval strategies. Furthermore, such a pipeline should also include a feature for editing the query—based on the methods of *Interactive Case Acquisition* — which also allows for configurations related to retrieval.

Indexing Support – As case bases grow, efficient indexing becomes increasingly important for maintaining acceptable retrieval performance. Tools can automatically suggest suitable indexing strategies or partitions based on key attributes or data distributions. Such assistance can help balance retrieval accuracy and efficiency without requiring deep expertise in indexing techniques.

Parameter Recommendation can assist in selecting appropriate retrieval parameters, such as the number of retrieved cases (k) or similarity thresholds. By providing default values, parameter suggestions, or feedback based on retrieval behavior, such mechanisms can help iteratively refine the retrieval configuration.

4.6 Adaptation Configuration

The Reuse step focuses on applying adaptation knowledge to tailor solutions. The configuration can be challenging, for example, when multiple adaptation steps need to be chained together and a (runtime) optimized sequence is required.

Visual Adaptation Pipeline Configuration – Support can also be provided during the practical application of adaptation knowledge. Interactive adaptation environments may suggest similar adaptation steps, potentially in a case-based manner, from which users can select, revise, or refine appropriate actions. Such environments may also support chained adaptations by suggesting

and managing sequences of dependent adaptation steps. In addition, LLMs may support the formulation, proposal, or explanation of adaptation steps.

4.7 Explaining Solutions

Explaining solutions is relevant within the Revise phase, where users need to assess, understand, and potentially modify the proposed solution. The difficulty lies both in explaining why a particular case is considered similar to the query case and in explaining how the final proposed solution emerged from the retrieved case and its adaptation [5,24,36].

Retrieval Explanation describes why a particular case is considered similar to the query. Support functions include visualizations of global similarity, local attribute contributions, and ranking behavior, which make the influence of individual features and weights more transparent [5,17,24,36]. In addition, interactive exploration with side-by-side comparisons of past retrieval results can further support understanding.

Adaptation Explanation complements similarity explanation and describes how the final solution emerged from the retrieved case. Adaptation trace explanations and side-by-side comparisons of the query, retrieved case, and adapted solution can help users understand which parts were reused, modified, or newly introduced. Natural-language explanations, potentially supported by LLMs, may further help communicate adaptation decisions in a more understandable way.

4.8 Validation and Evaluation

This step is part of the Revise phase, where the proposed and, if applicable, adapted solution has to be assessed regarding its usefulness for the target problem. Depending on the application, this may require real-world testing, expert assessment, user feedback, or simulation-based evaluation. The challenge is highly domain-dependent, since suitable evaluation criteria vary with the task and the extent to which outcomes can be observed directly. It is also often difficult to distinguish between weaknesses in retrieval and weaknesses in adaptation.

Evaluation Environments enable testing of proposed or adapted solutions under controlled conditions before being applied in practice. This may include simulation sandboxes, replay settings, or other domain-specific test environments.

Feedback Capture supports a structured feedback process. Interactive interfaces can guide domain experts in assessing the usefulness of proposed solutions based on ratings, qualitative feedback, and domain-specific criteria.

Evaluation Analysis supports a systematic interpretation of evaluation results. This may include performance metrics, comparisons of alternative configurations, and error analyses that distinguish between weaknesses in retrieval and weaknesses in adaptation. This may also include (technical) malfunctions, which are subject to testing.

4.9 Maintaining Knowledge

This step can be performed in the Retain phase to ensure the long-term efficiency and effectiveness of the case base. In practice, maintenance has to address duplicates, near-duplicates, obsolete or invalid cases, and domain drift, i.e., changes in the application context that make previously useful cases less representative over time [11,41]. At the same time, reducing the case base should not lead to a loss of competence. In addition to the case base, all knowledge containers are subject to maintenance activities and their close interdependencies have to be considered and coordinated [32].

Continuous Case Base Inspection supports identifying problematic or maintenance relevant cases. This includes the detection of duplicates, near-duplicates, outliers, obsolete cases, and possible signs of domain drift [18,25]. Such support can help domain experts review the current state of the case base more systematically and identify cases that may require further attention.

Remembering to Forget Analysis goes beyond general inspection and supports deciding which cases should be removed, merged, or retained without unnecessarily reducing the competence of the case base. Such support can build on competence-oriented analyses, e.g., by considering coverage or the contribution of individual cases to future problem solving, and can provide recommendations for competence-preserving deletion or consolidation decisions [8,38].

Maintenance Dashboards complementarily support the accessibility of inspection results and maintenance recommendations. Such dashboards can highlight likely deletion, merge, or retention candidates with explanations and thereby support informed maintenance decisions. In addition, they can visualize dependencies between knowledge containers, helping users identify required updates in related components and avoid inconsistencies across the overall knowledge base.

4.10 Deployment

The difficulty of this step does not only lie in making a CBR system technically available but also in integrating it into the software context. This may require integrating CBR functionality with existing process steps, user roles, data flows, and surrounding software systems, as well as onboarding and supporting users.

Interface Generation - To provide easy access to the functions of the CBR system, interfaces and reusable connectors can be provided, e.g., based on REST or MCP, as well as support for common data exchange formats [4,21,26].

Onboarding and Runtime Support supports the introduction and practical use of a CBR system in its target context. This includes onboarding materials such as documentation, tutorials, or explanatory videos that help users understand the workflow of the application and the principles of CBR. Conversational support, potentially based on LLMs, may complement such functionality by providing application-specific assistance across the broader software environment. During runtime, it may further assist users by explaining system functions, clarifying process steps, and guiding them through typical usage situations.

4.11 General Support Functions

Beyond support functions tailored to individual development steps, our analysis revealed several mechanisms that provide benefits across all challenges.

Implementation Recommendation provides guidance for selecting a suitable technological basis for a given application scenario. This includes choosing appropriate frameworks [37] and considering aspects such as extensibility, deployment options, integration requirements, and licensing constraints.

Guided Modeling Support in the form of wizard-style guidance can provide step-by-step assistance across multiple stages of the CBR development process, addressing all previously introduced challenges. Such guidance can be realized through intuitive, low-code graphical interfaces that reduce technical complexity (see e.g., jColibri [14], myCBR [40], eXiT [22], CloodCBR [26]) and enable domain experts to actively participate in system development. Error-tolerant interaction design, including validation and immediate feedback, can help prevent modeling errors and support incremental refinement. Interactive or conversational components, such as chatbot-like interfaces, can further facilitate knowledge elicitation and user engagement.

Template-Based System Design can support the development of CBR systems by enabling the reuse of previously defined system designs across multiple stages of the development process. Instead of constructing models from scratch, designers can retrieve and adapt existing templates that capture case structures, similarity models, or retrieval strategies [15], effectively treating system design itself as a case-based reasoning task. The COLIBRI ecosystem exemplifies this approach by providing reusable and adaptable CBR system templates that can be selected and refined based on project requirements [27,28,30,33]. This idea can be further extended by intelligent assistants that generate initial CBR system configurations from data and high-level user objectives, for example expressed in natural language, as proposed in our survey by a participant.

5 Conclusion, Limitations, and Future Work

In this paper, we investigated the development challenges for CBR applications with corresponding support potentials. We explored the current landscape of support functions and elicited practical insights from the literature and the CBR community on involving domain experts in the development. To summarize, there is a wide range of support functions already available but mostly considered for specific tasks. There is a potential to leverage synergies by integrating different support functions to provide support across different development steps. This work could be a first step in this direction. There might also be a potential to increase involvement of domain experts for various steps, such as defining a case structure or collecting cases. But support is needed, particularly for explaining the CBR paradigm to novices and for supporting the interdisciplinary collaboration to efficiently find a common, shared vocabulary.

Our study has some limitations that should be considered. In the literature review, we investigated related works with a strong focus on CBR that typi-

cally do not focus on domain experts and rarely distinguish between different stakeholders. The survey has been conducted with a limited number of participants who are mainly academic researchers and experienced in the development of CBR applications. Furthermore, the survey posed rather high-level questions with a selection of items derived from literature. To reduce effort for participants, we did not distinguish between types of CBR applications or domains, although CBR covers a broad application field with varying complexity. Consequently, the results should be considered as a first indication of the potential of domain expert involvement and support. Moreover, a prioritization for future framework development heavily depends on various factors such as CBR tasks, domain, and user knowledge.

In future work, a more extensive survey is needed that includes perspectives beyond academia and, in particular, more direct input from domain experts. Based on the identified challenges and support potentials, future research should prioritize relevant support functions for a selected CBR framework, implement them in a user-friendly environment, and evaluate their usefulness for involving domain experts. Existing support functions should be reused and connected through suitable interfaces where possible. In addition, the identified challenges and support potentials could be investigated in a more domain- and application-specific manner to address specific requirements.

Furthermore, recent advances in LLMs and agent-based approaches offer additional opportunities for supporting CBR development, in particular, assisting in generating an initial CBR system configuration and coordinating continuous evolutions over time across all knowledge containers. While current limitations, such as the limited availability of CBR-specific training data, may restrict their immediate applicability, leveraging generative and agent-based AI represents a promising direction for future research.

Acknowledgments. We would like to thank all people from the CBR research community who participated in our survey. This work is partly funded by Rhineland-Palatinate and the EU as part of the IMPACT project under grant no. 86003522 and by Trier University and DFKI as part of the development of the ProCAKE framework.

Declaration on Generative AI During the preparation of this work, the authors used ChatGPT and LanguageTool to: paraphrase and reword, improve writing style, Grammar and spelling check, peer review simulation, and content enhancement. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Commun. **7**(1), 39–59 (1994)
2. Bach, K., Althoff, K.: Developing Case-Based Reasoning Applications Using myCBR 3. In: 20th ICCBR Proceedings. LNCS, vol. 7466, pp. 17–31. Springer (2012). https://doi.org/10.1007/978-3-642-32986-9_4

3. Bach, K., Bergmann, R., Brand, F., Caro-Martínez, M., Eisenstadt, V., Floyd, M., Jayawardena, L., Leake, D., Lenz, M., Malburg, L., Ménager, D., Minor, M., Schack, B., Watson, I., Wilkerson, K., Wiratunga, N.: Case-Based Reasoning Meets Large Language Models: A Research Manifesto For Open Challenges and Research Directions (Mar 2025), https://hal.science/hal-05006761, working paper or preprint
4. Bach, K., Mathisen, B.M., Jaiswal, A.: Demonstrating the myCBR Rest API. In: 27th ICCBR Workshops Proceedings. CEUR Workshop Proceedings, vol. 2567, pp. 144–155. CEUR-WS.org (2019)
5. Bach, K., Mork, P.J.: On the Explanation of Similarity for Developing and Deploying CBR Systems. In: 33rd FLAIRS. pp. 413–416. FloridaOJ (2020)
6. Bach, K., Sauer, C.S., Althoff, K., Roth-Berghofer, T.: Knowledge Modeling with the Open Source Tool myCBR. In: 10th KESE Workshop Proceedings. CEUR Workshop Proceedings, vol. 1289. CEUR-WS.org (2014)
7. Bahls, D., Roth-Berghofer, T.: Explanation Support for the Case-Based Reasoning Tool myCBR. In: 22nd AAAI Proceedings. pp. 1844–1845. AAAI Press (2007)
8. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, LNCS, vol. 2432. Springer (2003). https://doi.org/10.1007/3-540-45759-3
9. Bergmann, R., Althoff, K., Breen, S., Göker, M.H., Manago, M., Traphöner, R., Wess, S.: Developing Industrial Case-Based Reasoning Applications: The INRECA-Methodology, LNCS, vol. 1612. Springer (1999). https://doi.org/10.1007/B94998
10. Bergmann, R., Grumbach, L., Malburg, L., Zeyen, C.: ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In: 27th ICCBR Workshop Proc. vol. 2567, pp. 156–161. CEUR-WS.org (2019)
11. Chebel-Morello, B., Haouchine, M.K., Zerhouni, N.: Case-based maintenance: Structuring and incrementing the case base. Knowl. Based Syst. **88**, 165–183 (2015). https://doi.org/10.1016/j.knosys.2015.07.034
12. Cordier, A., Fuchs, B., Mille, A.: Engineering and Learning of Adaptation Knowledge in Case-Based Reasoning. In: 15th EKAW Proc. LNCS, vol. 4248, pp. 303–317. Springer (2006). https://doi.org/10.1007/11891451_27
13. Craw, S., Wiratunga, N., Rowe, R.: Learning adaptation knowledge to improve case-based reasoning. Artif. Intell. **170**(16-17), 1175–1192 (2006). https://doi.org/10.1016/J.ARTINT.2006.09.001
14. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building CBR systems with jColibri. Sci. Comput. Program. **69**(1-3), 68–75 (2007). https://doi.org/10.1016/J.SCICO.2007.02.004
15. Fornells, A., Recio-García, J.A., Díaz-Agudo, B., Golobardes, E., Fornells, E.: Integration of a Methodology for Cluster-Based Retrieval in jColibri. In: 8th ICCBR Proceedings. LNCS, vol. 5650, pp. 418–433. Springer (2009). https://doi.org/10.1007/978-3-642-02998-1_30
16. Hotz, M., Malburg, L., Bergmann, R.: Advanced Search Techniques for Determining Optimal Sequences of Adaptation Rules in Process-Oriented Case-Based Reasoning. In: 33rd ICCBR Proc. LNCS, vol. 15662, pp. 236–251. Springer (2025). https://doi.org/10.1007/978-3-031-96559-3_16
17. Jiménez-Díaz, G., Díaz-Agudo, B.: Visualization of Similarity Models for CBR Comprehension and Maintenance. In: 32nd ICCBR Proceedings. LNCS, vol. 14775, pp. 67–80. Springer (2024). https://doi.org/10.1007/978-3-031-63646-2_5

18. Jiménez-Díaz, G., Díaz-Agudo, B.: SimViz: Interactive visualizations to analyze similarity measures. SoftwareX **32**, 102454 (2025). https://doi.org/10.1016/J.SOFTX.2025.102454
19. Jiménez-Díaz, G., Lenz, M., Malburg, L., Díaz-Agudo, B., Bergmann, R.: A Framework for Supporting the Iterative Design of CBR Applications. In: 33rd ICCBR Proceedings. LNCS, vol. 15662, pp. 252–266. Springer (2025). https://doi.org/10.1007/978-3-031-96559-3_17
20. Lamy, J., Sekar, B.D., Guézennec, G., Bouaud, J., Séroussi, B.: Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. Artif. Intell. Medicine **94**, 42–53 (2019). https://doi.org/10.1016/J.ARTMED.2019.01.001
21. Lenz, M., Malburg, L., Bergmann, R.: CBRkit: An Intuitive Case-Based Reasoning Toolkit for Python. In: 32nd ICCBR Proceedings. LNCS, vol. 14775, pp. 289–304. Springer (2024). https://doi.org/10.1007/978-3-031-63646-2_19
22. López, B., Pous, C., Gay, P., Pla, A., Sanz, J., Brunet, J.: eXiT*CBR: A framework for case-based medical diagnosis development and experimentation. Artif. Intell. Medicine **51**(2), 81–91 (2011). https://doi.org/10.1016/J.ARTMED.2010.09.002
23. Malburg, L., Hotz, M., Bergmann, R.: Improving Complex Adaptations in Process-Oriented Case-Based Reasoning by Applying Rule-Based Adaptation. In: 32nd ICCBR Proc. LNCS, vol. 14775, pp. 50–66. Springer (2024). https://doi.org/10.1007/978-3-031-63646-2_4
24. Marín-Veites, P., Bach, K.: Explaining CBR Systems Through Retrieval and Similarity Measure Visualizations: A Case Study. In: 30th ICCBR. LNCS, vol. 13405, pp. 111–124. Springer (2022). https://doi.org/10.1007/978-3-031-14923-8_8
25. McKenna, E., Smyth, B.: An Interactive Visualization Tool for Case-Based Reasoners. Applied Intelligence **14**(1), 95–114 (2001)
26. Nkisi-Orji, I., Wiratunga, N., Palihawadana, C., Recio-García, J.A., Corsar, D.: Clood CBR: Towards Microservices Oriented Case-Based Reasoning. In: 28th ICCBR Proc. pp. 129–143. Springer (2020). https://doi.org/10.1007/978-3-030-58342-2_9
27. Recio-García, J.A., Bridge, D.G., Díaz-Agudo, B., González-Calero, P.A.: CBR for CBR: A Case-Based Template Recommender System for Building Case-Based Systems. In: 9th ECCBR Proceedings. LNCS, vol. 5239, pp. 459–473. Springer (2008). https://doi.org/10.1007/978-3-540-85502-6_31
28. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Semantic templates for case-based reasoning systems. Knowl. Eng. Rev. **24**(3), 245–264 (2009). https://doi.org/10.1017/S0269888909990051
29. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: The COLIBRI Open Platform for the Reproducibility of CBR Applications. In: 21st ICCBR Proceedings. LNCS, vol. 7969, pp. 255–269. Springer (2013). https://doi.org/10.1007/978-3-642-39056-2_19
30. Recio-García, J.A., González-Calero, P.A., Díaz-Agudo, B.: Template-Based Design in COLIBRI Studio. Inf. Syst. **40**, 168–178 (2014). https://doi.org/10.1016/J.IS.2012.11.003
31. Richter, M.M.: Knowledge Containers. In: Readings in Case-Based Reasoning. Morgan Kaufmann Publishers (2003)
32. Roth-Berghofer, T.: Knowledge maintenance of case-based reasoning systems - the SIAM methodology. Ph.D. thesis, Kaiserslautern University of Technology, Germany (2003), https://d-nb.info/967388635

33. Roth-Berghofer, T., Sauer, C., Garcia, R.A.J., Bach, K., Althoff, K.D., Agudo, B.D.: Building case-based reasoning applications with myCBR and COLIBRI studio. Expert Update **13**(1) (2013)
34. Schander, R., Schultheis, A., Bergmann, R.: Explanation of Similarities in Temporal Case-Based Reasoning by Visualization. In: 34th ICCBR Proc. LNCS, vol. 16780. Springer (2026). https://doi.org/10.1007/978-3-032-33865-5_11
35. Schoenborn, J.M., Weber, R.O., Aha, D.W., Cassens, J., Althoff, K.D.: Explainable Case-Based Reasoning: A Survey. In: AAAI-21 Workshop Proc. (2021)
36. Schultheis, A., Hoffmann, M., Malburg, L., Bergmann, R.: Explanation of Similarities in Process-Oriented Case-Based Reasoning by Visualization. In: 31st ICCBR Proc. LNCS, vol. 14141, pp. 53–68. Springer (2023). https://doi.org/10.1007/978-3-031-40177-0_4
37. Schultheis, A., Zeyen, C., Bergmann, R.: An Overview and Comparison of Case-Based Reasoning Frameworks. In: 31st ICCBR Proc. LNCS, vol. 14141, pp. 327–343. Springer (2023). https://doi.org/10.1007/978-3-031-40177-0_21
38. Smyth, B., Keane, M.T.: Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In: 14th IJCAI Proc. pp. 377–383. Morgan Kaufmann (1995)
39. Smyth, B., McKenna, E.: Modelling the Competence of Case-Bases. In: 4th EWCBR Proceedings. LNCS, vol. 1488, pp. 208–220. Springer (1998). https://doi.org/10.1007/BFB0056334
40. Stahl, A., Roth-Berghofer, T.: Rapid Prototyping of CBR Applications with the Open Source Tool myCBR. In: 9th ECCBR Proceedings. LNCS, vol. 5239, pp. 615–629. Springer (2008). https://doi.org/10.1007/978-3-540-85502-6_42
41. Wilson, D.C., Leake, D.B.: Maintaining Case-Based Reasoners: Dimensions and Directions. Comput. Intell. **17**(2), 196–213 (2001). https://doi.org/10.1111/0824-7935.00140
42. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, 2nd edn. (2024). https://doi.org/10.1007/978-3-662-69306-3