A Framework for Supporting the Iterative Design of CBR Applications*

Guillermo Jimenez-Diaz[1], Mirko Lenz[2,3], Lukas Malburg[2,3], Belén Díaz-Agudo[1], and Ralph Bergmann[2,3]

[1] Department of Software Engineering and Artificial Intelligence
Instituto de Tecnologías del Conocimiento
Universidad Complutense de Madrid, Spain
{gjimenez,belend}@ucm.es
[2] Artificial Intelligence and Intelligent Information Systems, Trier University,
54296 Trier, Germany, www.wi2.uni-trier.de
info@mirko-lenz.de, {malburgl,bergmann}@uni-trier.de
[3] German Research Center for Artificial Intelligence (DFKI)
Behringstraße 21, 54296 Trier, Germany, ebls.dfki.de
{mirko.lenz,lukas.malburg,ralph.bergmann}@dfki.de

Abstract. Iterative design is a well-known methodology that involves prototyping, testing, analyzing, and refining products or processes. Rapid prototyping and evaluation are crucial, enabling designers to quickly identify and resolve issues and iteratively improve the design. However, effective iterative design relies on tools that accelerate both prototype creation and visual evaluation. This paper aims to address this challenge by presenting a framework specifically designed to enhance the iterative design process for CBR applications. In this context, we emphasize the manual and knowledge-intensive task of defining similarity measures. Furthermore, we introduce a proof-of-concept implementation of this framework based on the CBRkit toolkit and the SimViz visualization tool. We studied the capabilities to support the iterative design of CBR applications through a case study in the prototypical cars domain.

Keywords: Case-Based Reasoning · Iterative Design · CBR Frameworks · Rapid Prototyping · Visualization · Similarity · CBRkit · SimViz

1 Introduction

The Case-Based Reasoning (CBR) literature provides various frameworks designed to streamline the design and implementation of CBR applications (see [20] for a comprehensive review). One of the most challenging aspects of CBR development is gaining a deep understanding of the domain and constructing effective similarity measures that accurately capture its intricacies. Several approaches exist to tackle this problem [18]. A common method is manual programming,

* The Version of Record is available online: doi.org/10.1007/978-3-031-96559-3_17

where domain experts define similarity functions based on predefined heuristics and rules. Another approach involves learning from data, where machine learning techniques are applied to automatically infer similarity measures from case examples. Regardless of the chosen option, modeling similarity measures is a critical and multifaceted task that requires an iterative modeling-testing loop. Visualizations play a key role in overcoming intrinsic modeling challenges by offering intuitive insights and improving the understanding of complex relationships within the data. As evidenced by previous approaches (e.g., [13,3,17]), visual tools facilitate expert-assisted methodologies that seamlessly integrate human expertise with computational techniques. By leveraging expert knowledge, these tools guide the development of similarity measures while iteratively refining them through systematic testing and feedback.

This paper aims to improve CBR development by integrating visualization techniques with CBR frameworks, helping developers and making CBR more accessible, especially to those outside the research community. To address the complexities of building CBR systems and designing effective similarity measures, we apply *iterative design* [6], a well-established methodology involving prototyping, testing, analyzing, and refining a product or process. This approach improves design quality and functionality by using system interaction as a research tool to continuously refine and evolve the project. Developers create a prototype, test and analyze it, refine it based on feedback, and repeat the cycle to converge toward an optimal solution.

Although individual CBR frameworks provide structured methodologies for designing CBR applications and there are some initial approaches to visualizing similarity measures during their construction [21,8], there is currently a lack of flexible solutions that integrate seamlessly into existing CBR frameworks. As a result, assessing the impact of modifications in similarity measures on similarity assessments remains a significant challenge. This issue is particularly critical in CBR, as the retrieval of relevant cases is heavily based on well-designed similarity measures, which, in turn, influence subsequent phases of the reasoning process.

Specifically, we demonstrate how the CBRkit framework [11] can be integrated with the SimViz approach [8] to facilitate the design of CBR systems in this paper. This integration involves identifying their inter dependencies, determining the necessary metadata for similarity measure computation, and exploring methods for visualizing and dynamically recalculating results when similarity measures are modified by the CBR developer. The emphasis of CBRkit on usability and integration capabilities, such as seamless compatibility with visualization tools like SimViz, makes it an ideal choice for researchers and practitioners aiming to build CBR systems. Through its streamlined design, CBRkit reduces the complexity of implementing CBR solutions while maintaining adaptability to diverse application scenarios.

The main contribution of this paper is the design and implementation of a framework to support the iterative design process for creating CBR similarity measures in CBR systems. This paper represents the integration effort between CBRkit [11] a flexible and modular framework designed to simplify the devel-

opment and deployment of Case-Based Reasoning systems, and SimViz [8], an exploratory visualization tool aimed at understanding and identifying errors in both data and similarity measures. CBRkit supports customizable similarity measures and allows users to tailor the framework to specific domain requirements, and SimViz visualizations provide insight into the similarity between local and global attributes across different case representations. We propose an applied methodology using the CBRkit-SimViz tandem to ensure systematic development and evaluation for designing, prototyping, and evaluating CBR systems.

The paper is structured as follows: Section 2 reviews several CBR frameworks from a visualization perspective. Based on this literature review, in Section 3 we describe the proposed design framework and identify the requirements that should be met to support the CBR development cycle and enhance understanding of the domain, data, and similarity measures. Section 4 describes the architecture and implementation of the integrated system, focusing on the data flow between CBRkit (design and prototype) and SimViz (evaluate). We use the cars domain, which includes rich attribute diversity and relationships characteristic of many real-world similarity modeling challenges, for demonstrating a step-by-step CBR development cycle with SimViz visualization during the design phase. The cars domain has been extensively used in CBR research as a standard benchmark to illustrate core concepts and methodologies [24,15,12].

Section 5 concludes the paper by reviewing key lessons learned from the integration of these tools and outlining future research lines.

2 State-of-the-Art

Iterative design emerged as a need in the 1980s for the development of user interfaces and with the rise of the discipline of Human-Computer Interaction [6]. It attempts to solve the problems of designing new interfaces and adapting them to users, rather than training users to use them. The foundations of iterative design lie in the development of "cheap" prototypes whose purpose is to test a design idea. The performance of the prototype is evaluated with users, and a new design is proposed based on the evaluation findings. Iterative design methods have become standard in software development and management and are applied in different methodologies such as Spiral Development [5], Scrum [22], or Lean Software Development [19], among others. The iterative design cycle is more effective when the iterations are fast enough to determine whether a solution has value. Rapid prototyping and evaluation are crucial, enabling designers to quickly identify and address issues and iteratively improve the design. This requires tools that accelerate both prototype creation and the analysis of user feedback.

Visualization techniques play a vital role in this process, enabling users to quickly grasp the results of design choices, evaluate prototypes, and facilitate rapid adjustments. Effective evaluation requires the ability to visually access detailed information, connect disparate data points, identify patterns and deviations, and explore information from multiple perspectives [25]. In the context of CBR, the design of effective similarity measures is critical, and existing CBR

frameworks enable the development of tailored CBR prototypes and further support this rapid iteration. Visualizations serve as powerful evaluation tools that help to understand information [1]. Interactive visualizations empower evaluators by allowing them to dynamically select, manipulate, and refine the displayed information, tailoring the visualization to their specific evaluation goals and enhancing its effectiveness.

In the following sections, we explore approaches and methods relevant to integrating visualization techniques for interactive design within CBR systems. This exploration is twofold: First, we examine the interactive design support provided by current CBR frameworks, based on Schultheis et al. [20]. Second, we discuss visualization approaches for interactive design, particularly during the key similarity measure modeling phase in CBR. Finally, we synthesize these contributions and identify the research gaps that motivate this paper.

2.1 Support of Interactive Design in CBR Frameworks

Schultheis et al. [20] present a review of open source CBR frameworks used to develop specific CBR applications. The authors compare the following CBR frameworks: *CloodCBR*, *eXiT*CBR*, *jColibri*, *myCBR*, and *ProCAKE*. In addition, they define a framework as "[...] a generic, domain-independent, and extensible software component that enables the implementation of specific applications." [20, p. 327]. All CBR frameworks have been compared in terms of supported CBR types and case representations, support to implement the knowledge containers, the CBR phases that can be applied, interfaces, and other additional features. In the following, we compare the results of the review, with a special focus on the visualization capabilities of the CBR frameworks and their support for interactive design. Based on this detailed analysis, we are able to identify current gaps in visualization techniques and derive requirements for developing suitable interactive design capabilities for CBR systems.

CloodCBR is a CBR framework that supports the development of textual and structural CBR applications. Regarding visualization capabilities, Schultheis et al. [20] determine that a web-based GUI is provided as a dashboard to visualize the CBR cycle. *eXiT*CBR* supports the development of structural CBR applications. A GUI is provided for configuration, to perform retrievals, data export, and the visualization of the results. *jColibri* is a Java-based CBR framework that supports the development of textual, structural, and conversational CBR applications. The framework provides a GUI for configuration and visualization tools for the case base. *myCBR* supports the development of textual and structural CBR applications. It is the only CBR framework that provides a GUI for modeling similarity measures and thus supports the interactive design of similarity measures. *ProCAKE* is a CBR framework that enables developers to create textual, structural, and process-oriented CBR applications. The framework provides a GUI for visualization and the creation of cases.

2.2 Approaches for Visualization of Similarity Measures

In addition to the support of interactive design to model similarity measures in CBR frameworks, there exist several visualization approaches and methods that can be beneficial for this purpose.

Schultheis et al. [21] present a visualization approach that helps to understand the similarity assessment of complex cases in the context of process-oriented CBR. For this purpose, three adopted visualization techniques have been used with the aim of supporting the design of similarity measures. Similarly, the SimViz approach [8] is a tool for visualizing similarity functions. It offers a web-based GUI for users to assess the similarity between cases in a case base, using various visualization techniques. Changing similarity measures triggers recalculations of similarity values. Bach and Mork [3] propose a similarity visualization method in CBR to aid in visualizing similarity measures during the retrieval phase.

Marin-Veites et al. [13] propose a visual explanation of the similarity scores for the query cases by comparing the global and local similarity measures of the attributes for end users in a medical domain. They use bar charts to compare the case similarity between the top 5 ranks of most similar cases and a query case. The visualization shows not only the global similarity, but also the local similarity scores that intervene in the similarity measure.

The works in [16,17,23] propose visualization methods based on the use of graphs and spring force models for visual exploration of case bases and the similarity measures applied to them. The distance between cases in a two-dimensional space represents their similarity, so similar cases will be drawn close together, and dissimilar cases should be drawn far apart. This visualization is useful to identify regions of similar cases and gaps without representative cases or outliers, but it only works for small case bases.

The work in [9] describes another alternative to provide visual explanations of the similarity scores between a query and the retrieved cases. These visualizations are based on scatter plots and rainbow boxes and are aimed at end users, i.e., medical experts of the CBR system. The complexity of the visualization used led the authors to simplify and reduce the dimensions and information displayed by the tool, which required a short training before using it.

In [14], the authors use visualization techniques to increase the explainability of the results of a CBR system. For this purpose, coordinate diagrams are used that represent the similarities between the cases.

2.3 Summary and Research Gaps

Schultheis et al. [20] note that while most CBR frameworks offer GUIs for visualization and configuration, they rarely support the interactive design of similarity measures. Although visualization techniques exist, they mainly focus on explaining similarities rather than guiding the design process. Moreover, these tools are often standalone, limiting integration with modern CBR frameworks. This lack of interactive design capabilities hinders broader applicability, both within and

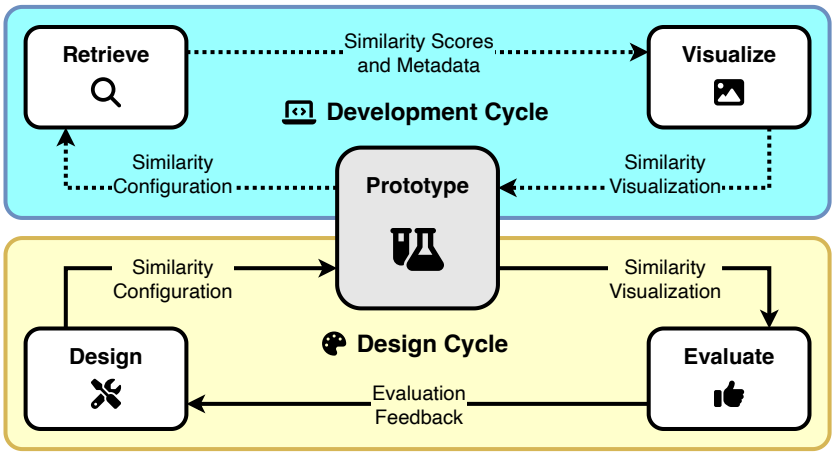

Fig. 1. Iterative CBR design process with integrated visualization.

outside the CBR research community. To address these limitations, we propose a tightly integrated architecture that combines visualization techniques directly within CBR frameworks, enabling interactive similarity measure design tailored to various applications.

3 Iterative Design Framework

The goal of this section is to propose an implementation-agnostic description of the proposed iterative process for designing CBR applications that can be used as a template for other work in this area. A reference implementation of this framework is provided in Section 4. Figure 1 shows an overview of the proposed iterative design process. Compared to the standard process described in Section 2, our approach focuses on the integration of CBR development frameworks and visualization tools to enable rapid prototyping. As a result, practitioners are no longer required to model rather complex similarity measures by hand, but can rely on visual guidance and feedback through the process. In order to enable these interactions, the framework needs to meet certain *constraints*—expressed as *requirements* in the context of our envisioned integration—that will be discussed as part of this framework description. These requirements have been derived based on current CBR frameworks supporting visualizations and additional standalone tools helping CBR developers during modeling similarity measures (see Sect. 2). In addition, the requirements reflect both our expertise as CBR researchers and our experience developing CBR applications, thereby representing the user group of CBR developers. In addition to *mandatory* requirements (MR), we propose a set of *optional* requirements (OR) that can further enhance the user experience, but may be omitted for certain use cases. We divide the proposed framework into three sections: (i) *Designing* a similarity configuration, (ii) visualization-aware *prototyping* as part of the development cycle, and (iii) end-to-end *evaluation*.

We want to emphasize that similarity is useful in different CBR tasks, not just in retrieval. It plays a crucial role in adaptation, reuse, and learning by ensuring that relevant cases are identified, appropriately modified, and integrated into the system effectively. As such, finding optimal similarity measures for a domain is a central task in the overall CBR design process.

3.1 Designing a Similarity Configuration

When starting from scratch, CBR developers face a cold start problem as they cannot rely on existing similarity visualizations or expert feedback. Consequently, they must establish a set of initial similarity measures for the specific case representation based on the data types involved and their prior experience. This bootstrapping issue is beyond the scope of our work, but could be tackled with the help of Large Language Models (LLMs)—for instance, by automatically selecting and parameterizing similarity measures of established CBR frameworks as proposed by Lenz et al. [10]. Having obtained an initial configuration, the CBR developer can draw on two crucial sources of knowledge during this phase: the evaluation of the similarity score and the feedback from the domain expert. Using these resources, they can refine the similarity configuration and initiate the next iteration of the design process.

MR1 (Similarity Configuration)**.** The framework must provide a description of the similarity configuration used by the CBR system prototype. This includes the functions used to determine local similarities and the aggregation method to calculate the global similarity score.

3.2 Visualization-Aware Prototyping

After an initial similarity configuration is established, the CBR developers can evaluate the retrieval performance using a CBR system prototype. Typically, this involves defining a query, executing it against the available case base, and examining the retrieved cases along with their corresponding numeric similarity values. This process can be tedious and time-consuming, with only little or no insight into the underlying computations. Instead, providing visual feedback on the similarity scores and the impact of individual similarity measures can significantly enhance the expert's understanding of the system's behavior. In this context, relying solely on the global aggregated similarity score is insufficient, and further data is needed to generate appropriate visual representations.

MR2 (Case Representation)**.** The framework must define a machine-readable representation of the case base and the queries. This makes it possible to analyze the data types and select appropriate visualizations.

MR3 (Similarity Scores)**.** The framework must generate data on the local similarity scores for each attribute of the retrieved cases in addition to the overall global score. This data will be used to generate visual representations to explore and analyze the similarity functions described in the previous stage.

OR1 (CBR Introspection)**.** The framework should provide information on all the CBR processes required to achieve the final solution. This enables fine-grained analysis of complex patterns, such as retrieval with MAC/FAC [7].

OR2 (Arbitrary Case Representation)**.** The framework should support not only simple attribute-value case representations, but also more arbitrary representations such as object-oriented structures, hierarchical data, graph-based structures, or even images.

OR3 (Interactivity)**.** The framework should allow interactive configuration of the similarity measures. Instead of using hard-coded or built-in metrics, the user should be able to tweak the parameters and observe the impact on the computed scores in real time.

3.3 End-to-end Evaluation

The final step in the iterative design process is the evaluation of the system with domain experts. Having completed the prototyping phase, the CBR developer already has a working implementation at hand together with a corresponding similarity visualization. Contacting domain experts is resource intensive and time-consuming, so the CBR developer may decide to perform multiple rounds of prototyping first. However, actual feedback from people working in the field is invaluable and can help to identify issues that may not be apparent to the CBR developer. The generated visualization can be helpful for this task as it helps the CBR developer to explain the system's behavior to the domain expert, and, thus, provides an introspective view of the system's behavior. Afterward, the CBR developer can incorporate the suggestions into the similarity configuration as part of the next iteration of the design cycle.

MR4 (Interactive Visualization)**.** The evaluation process must be supported with interactive features. Interactive visualization should be preferred to support the evaluation stage because it will ease the engagement of the different users involved in the evaluation process.

OR4 (Automated Evaluation)**.** The framework should provide automated evaluation methods to assess the appropriateness of the computed similarities. This can, for instance, be achieved by comparing a retrieval result with a set of known relevant cases crafted by human experts. This allows a proper assessment of a good number of cases and helps in maintaining the case base.

4 Proof-of-Concept Implementation

Based on the proposed framework for supporting the iterative CBR design process, in this section, we present a proof-of-concept that instantiates the theoretical framework and works as an example of how to implement and use it in a real use case. For this purpose, we propose the integration of CBRkit [11], an easily applicable CBR framework for the prototyping stage, and SimViz [8], a visualization tool for the visual evaluation stage. In the following, we discuss

the concrete integration between these two tools and their data flow to implement the proposed. In addition, we conduct a case study for evaluation and to demonstrate the suitability of their integration. For this purpose, we use the cars domain, and discuss step by step the iterative CBR design process using both tools.

4.1 CBRkit

CBRkit [11] is a Python-based framework for building CBR applications[4]. It is designed to be modular and flexible and uses a declarative approach to build custom pipelines for the different stages of the CBR cycle. The library comes with a set of built-in components such as similarity measures—making it easy to get started—but also allows for the creation of completely custom functions for more advanced use cases. Built-in functions are available for basic data types like strings or numbers, but also more complex ones like attribute-value data or graphs. CBRkit does not introduce a new abstraction layer through some similarity configuration format, instead it uses native Python code for declaratively defining the desired similarity measures. In addition to running CBR pipelines using these, it is also possible to expose them via a REST API, which allows for easy integration with other systems like SimViz.

The first version of CBRkit published in 2024 already met some of the requirements outlined in Section 3: It is possible to define CBR pipelines consisting of multiple retrieval and reuse stages—making it possible to apply MAC/FAC—which are then transparently exposed in the result object (OR1). Its way of defining similarity measures is flexible enough to allow for arbitrary case representations (OR2). In addition, the resulting similarity scores can be *structured* and contain additional metadata—for instance, local similarity scores for each attribute of the retrieved cases (MR3).

However, two mandatory and two optional requirements were still unmet. First, while easy to configure, the similarity measure definitions were not exposed in a machine-readable format in the result object (MR1). These measures are defined in plain Python code, making the extraction of the required information complex and not feasible without changes to the core of the library. This lack of abstraction also means that the similarity configuration cannot be dynamically changed (OR3). In addition, while CBRkit can handle arbitrary case representations, these are not passed to downstream components due to missing serialization techniques. Furthermore, automated evaluation is not possible due to the absence of interfaces for standardized evaluation metrics such as precision/recall or completeness/correctness (OR4). Lastly, the mentioned REST interface was a byproduct of the library, not a primary feature, leading to an overly verbose and difficult-to-consume API specification.

[4] Source code: `https://github.com/wi2trier/cbrkit` (MIT license).

4.2 SimViz

SimViz[5] (**Sim**ilarity **Vi**suali**Z**ation) is a tool that focuses on the interactive visualization of similarity measures and its application to different case bases [8]. SimViz can visualize structured cases based on attribute-value representations. It uses case base metadata with information about attribute data types to choose the appropriate visualization (MR2). In addition to basic datatypes (strings, numbers, and symbols), it provides specific visualizations for visual attributes (like colors and images) and symbols arranged in a taxonomy.

SimViz supports global-local similarity measures based on weighted similarity. It uses similarity configuration metadata about global and local similarity measures to display visual explanations about the case attributes, and the similarity measures employed (MR1). It also provides heatmaps and histograms to explore the distribution of similarity scores over a case base (MR3). More specifically, the global similarity score between two cases can be visually analyzed, showing how it is deconstructed into local similarity values and which functions are employed to compute them.

Since its early design, SimViz supports the interactive exploration of the case base and the similarity scores (MR4). First, users can click on the heatmap to select a particular pair of cases, or users can randomly select a pair of cases by clicking on a particular histogram bar or on a taxonomic concept. Moreover, users can explore the similarity distribution of an individual case against the other cases in the case base using an interactive stripe chart. Finally, SimViz partially supports the interactive configuration of the similarity measures (OR3). Users can modify the predefined weights of the local similarity measures and compare the effect of these changes on the global similarity value, recalculating it without the need of any additional prototyping tool.

Although SimViz meets all mandatory requirements, some optional ones are still missing. Designed primarily for similarity visualization, SimViz does not support introspection of other stages in the CBR process (OR1). Furthermore, requirement OR2 is only partially fulfilled. Although it includes visualization for taxonomies, images, and colors, SimViz cannot handle arbitrary case representations. We are working on supporting hierarchical data, with future plans to include its representation. Finally, automated evaluation is not supported as its interface does not visualize any standardized evaluation metrics (OR4).

4.3 Integration of CBRkit and SimViz

In this section, we detail the integration process of CBRkit and SimViz. CBRkit is used to design similarity measures and prototype the retrieval cycle to generate similarity scores, while SimViz utilizes these data, along with the case base and its case representation, to visualize and evaluate the similarity values and the similarity measures employed. Figure 2 illustrates the use of both tools to implement the proposed theoretical framework.

[5] Source code: `https://github.com/gjimenezUCM/simviz` (Apache 2.0 License). Deployed version: `https://gjimenezucm.github.io/simviz/`.

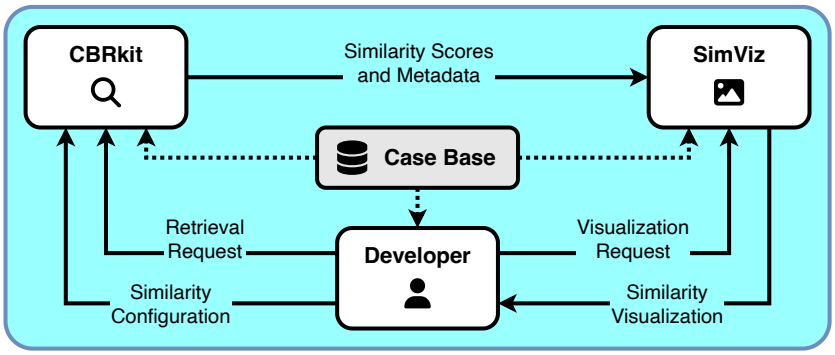

Fig. 2. Architecture of the integration between CBRkit and SimViz.

A significant part of the integration involves defining and adapting the data formats used by both systems. First, we need a machine-readable case representation (MR2) that can be read by both systems. To pass the retrieved cases and the underlying query to SimViz, a set of *dumpers* has been implemented, such as those to convert the data into the widely used JSON format. We intentionally omitted additional metadata, such as descriptions of the data types, as this information can be easily inferred from the available data. For use with SimViz, the exported results have been manually merged with the SimViz data used to load the case base—combining metadata about the case representation and the retrieved cases.

The second change in CBRkit is the introduction of a mechanism to gather information about the similarity configuration provided (MR1). Our goal was to transmit *everything* given by the user in a machine-readable format to downstream applications such as SimViz. To achieve that, *all* built-in similarity measures have been refactored to use a class-based interface that exposes the parameters of the measure set by the user via class attributes. By leveraging Python's magic methods, instances of these classes can still be called as if they were functions, allowing for a seamless transition from the original implementation. Besides the parameters, we automatically determine the name of the function and collect the docstring of the class as a description of the measure. This approach has been implemented in a generic way that works not only for the built-in metrics, but also for custom similarity functions defined by the user. Using these new capabilities, we can generate files containing the similarity configuration (MR1) and similarity scores (MR3) to be ingested by SimViz for visual evaluation. As SimViz expects the data in a specific format, we implemented an adapter to convert between the formats—making it possible for SimViz to directly work with the CBRkit results.

Next, we added an evaluation module (OR4) to CBRkit that is based on the `ranx` library [4]. In an effort to simplify integration with other systems, we also completely overhauled the REST API, which now offers a fully typed OpenAPI specification that can be used to generate client libraries for different programming languages. Currently, the evaluation module is not yet integrated

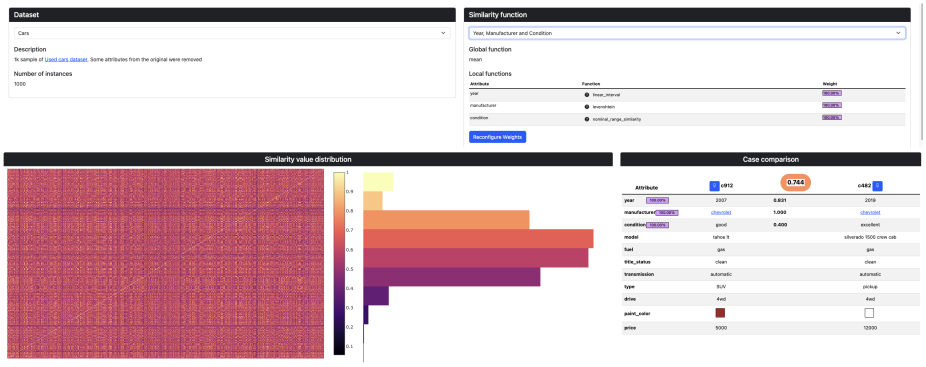

Fig. 3. Visualization of Case Representation With Attribute Weighting (Full image available at `https://tinyurl.com/musaj774`)

into the REST API, but we see this as a first step towards a more automated evaluation of the CBR system.

The last missing piece is an interactive way to define similarity measures in CBRkit (OR3). One way to tackle this aspect would be to define a serialization format manually configuring CBRkit—leading to a high maintenance burden. A more elegant way is to automatically generate a schema from the set of available Python functions based on the available type annotations. Doing this in a generic way is not trivial, since CBRkit supports not only attribute-value representations, but also graphs or fully custom, domain-specific representations. We therefore decided to leave this requirement for future work and focus on the integration with SimViz first.

4.4 Case Study in the Cars Domain

To demonstrate the suitability of the proposed integration between a CBR framework and a visualization tool, we present a case study on the integration of CBRkit and SimViz. For this, we use the cars domain, which includes a dataset of 1,000 cars[6] The cars domain represents an exemplary case study for similarity modeling, embodying challenges common across real-world applications. This dataset requires sophisticated similarity measures that can appropriately handle different data types while capturing their relative importance in determining the overall similarity. Its 11 attributes and their metadata are crucial for the integration between CBRkit and SimViz. Below, we demonstrate how this integration can support the iterative design process in the cars domain.

We started the design process by analyzing the dataset and exploring the catalog of similarity measures provided by CBRkit. After that, we use CBRkit to implement a first version of a similarity measure that combines local similarities using a weighted mean. This measure combines diverse functions such as a linear similarity for the *year* attribute, Levenshtein for the *manufacturer* attribute,

[6] The used cars dataset is available at `https://doi.org/10.5281/zenodo.15006920`.

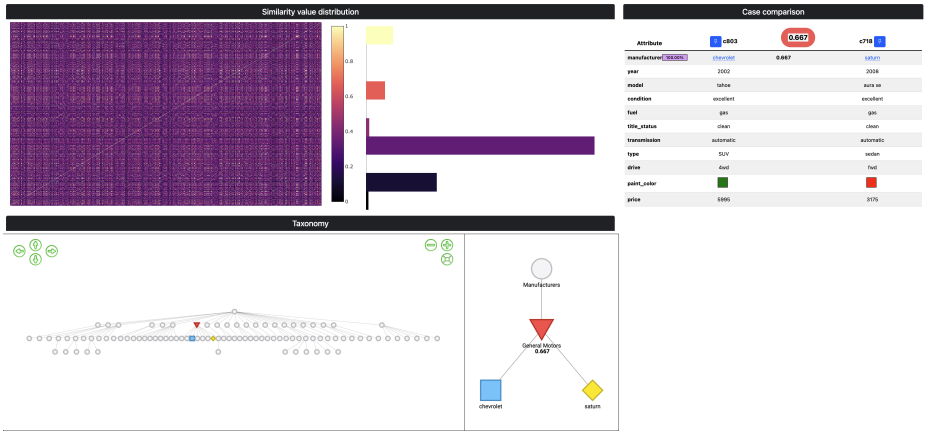

Fig. 4. Explanation of Local Similarities by Visualization (Full image available at `https://tinyurl.com/mr3y4n2e`).

and a custom function for linear similarity in a set of values for the *condition* attribute. The similarity configuration and similarity scores data created using CBRkit are used in SimViz to visualize the designed similarity measures.

Fig. 3 illustrates the dataset used, the case representation, including attribute weighting, and the similarity measures applied. The top section shows the used dataset (left) and the similarity configuration (right). The bottom section is for the interactive visualization of similarity scores data, showing the distribution of the similarity values in the case base (left), and a case comparison between two selected cases (right). In this context, it is visualized with which importance each attribute contributes to the final similarity value. Furthermore, it is also possible to focus on local similarity values and their corresponding similarity measures.

The iterative process continues using the Similarity configuration panel provided by SimViz, where we can modify the predefined weights of the local similarity measures. This feature supports a fine-tuning of the similarities in the case base, visualizing the effect of these changes on the global similarity value.

The iterative process can return to the design stage in order to experiment with more complex similarity measures, using a taxonomy for the *manufacturer* attribute. In this case, we return to CBRkit to use the similarity measures that this framework implements for taxonomies to create new similarity configurations and data scores and visualize these new similarity functions. Back to SimViz to analyze them, Figure 4 shows how SimViz supports the visualization of the taxonomy used for similarity calculation. To better understand the assessment of similarity between the two attribute values, i.e., *Chevrolet* and *Saturn*, the corresponding nodes in the taxonomy are highlighted. This helps both the CBR developer and the domain expert determine how the similarity originates.

5 Lessons Learned, Limitations, and Future Work

Developing effective similarity measures remains a challenge in CBR system design and implementation. Although existing CBR frameworks provide structured methodologies, they often lack the flexibility to integrate with visualization techniques and tools. CBRkit addresses this gap, standing out for their focus on usability and integration. In this paper, we show how the integration of CBRkit with a visualization tool such as SimViz makes it an ideal choice for researchers and practitioners. We demonstrate how CBRkit's streamlined design simplifies CBR solution implementation while maintaining adaptability for various applications. Our approach improves the interpretability of similarity measures, facilitating their iterative refinement, and ultimately making CBR systems more accessible to a broader audience. By visualizing the behavior of similarity measures, developers can gain more profound insight into their impact on case retrieval and system performance, leading to more effective design choices. Based on our experience during the integration of both tools, we identified several paths for future work:

Data-driven process. Currently, similarity measures must be manually defined in CBRkit based on a configuration. A more integrated approach would involve a common interchange format for CBRkit and SimViz, enabling automatic generation of similarity measures.

Orchestration. In its current state, both tools are loosely coupled: CBRkit generates files that are consumed by SimViz. A deeper integration could involve an orchestrated workflow in which SimViz initiates the similarity computation using CBRkit through an integrated user interface to configure the similarity configuration.

Data format standardization. To bridge the different case representation formats, we created an adapter. A better solution would be to standardize the data formats to meet both applications' requirements.

Limitations in data size. Large data files can result in high latency when running as a service. For example, the Cars 1k dataset generates similarity score files of around 200 MB. The API should support selectively querying only the relevant data to mitigate this issue.

Evaluation metrics. While CBRkit includes an initial implementation of an evaluation module, it is not yet part of the integration. To fulfill OR4, it should allow loading a gold standard corpus and comparing the similarity scores with the expected results.

In future work, we will extend this integration by adding more visualization techniques and evaluating their effectiveness through empirical studies. In addition, we want to examine how large language models (LLMs) can be used to support similarity modeling by suitable explanations [2]. We also plan to expand visualization to other key CBR processes, particularly reuse and case base maintenance. Additionally, by providing an OpenAPI specification, we enable easy re-implementation by other libraries, fostering future integration of SimViz with other CBR frameworks and promoting a more collaborative, interoperable CBR ecosystem.

Acknowledgments. Supported by the UCM (Research Group 921330) and the AUDITIA-X project PID2023-150566OB-I00, funded by the Ministry of Science and Innovation of Spain (`https://gaia.fdi.ucm.es/research/auditia-x/`), the association *Eifelkreis Digital* with its project KITEi, and the *Studienstiftung*.

References

1. Azzam, T., et al.: Data Visualization and Evaluation. New Directions for Evaluation **2013**(139), 7–32 (2013)
2. Bach, K., et al.: Case-Based Reasoning Meets Large Language Models: A Research Manifesto For Open Challenges and Research Directions. HAL Science **hal-05006761v1** (2025), `https://hal.science/hal-05006761`, working paper or preprint
3. Bach, K., Mork, P.J.: On the Explanation of Similarity for Developing and Deploying CBR Systems. In: 33rd FLAIRS. pp. 413–416. AAAI Press (2020)
4. Bassani, E.: Ranx: A Blazing-Fast Python Library for Ranking Evaluation and Comparison. In: Adv. in Inf. Ret. pp. 259–264. Springer (2022)
5. Boehm, B.W.: A spiral model of software development and enhancement. Computer **21**(5), 61–72 (May 1988)
6. Buxton, W., Sniderman, R.: Iteration in the Design of the Human-Computer Interface. In: 13th HFAC. pp. 72–81 (1980)
7. Forbus, K.D., Gentner, D., Law, K.: MAC/FAC: A model of similarity-based retrieval. Cogn. Sci. **19**(2), 141–205 (1995)
8. Jiménez-Díaz, G., Díaz-Agudo, B.: Visualization of Similarity Models for CBR Comprehension and Maintenance. In: 32nd ICCBR. vol. 14775, pp. 67–80 (2024)
9. 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)
10. Lenz, M., Hoffmann, M., Bergmann, R.: LLsiM: Large Language Models for Similarity Assessment in Case-Based Reasoning. In: 33rd ICCBR in Biarritz, France. Springer (2025), Accepted for Publication.
11. Lenz, M., Malburg, L., Bergmann, R.: CBRkit: An Intuitive Case-Based Reasoning Toolkit for Python. In: 32nd ICCBR. vol. 14775, pp. 289–304. Springer (2024)
12. Malburg, L., Hoffmann, M., Trumm, S., Bergmann, R.: Improving Similarity-Based Retrieval Efficiency by Using Graphic Processing Units in Case-Based Reasoning. In: 34th FLAIRS (2021)
13. 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)
14. Massie, S., Craw, S., Wiratunga, N.: Visualisation of Case-Base Reasoning for Explanation. In: ECCBR 2004 Workshops. pp. 135–144 (2004)
15. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning Similarity Measures from Data. Prog. Artif. Intell. **9**(2), 129–143 (2020)
16. McArdle, G., Wilson, D.C.: Visualising Case-Base Usage. In: 5th ICCBR Workshops. pp. 105–114. Springer (2003)
17. Namee, B.M., Delany, S.J.: CBTV: Visualising Case Bases for Similarity Measure Design and Selection. In: 18th ICCBR, vol. 6176, pp. 213–227. Springer (2010)
18. Ontañón, S.: An overview of distance and similarity functions for structured data. Artif. Intell. Rev. **53**(7), 5309–5351 (2020)

19. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. The Agile Software Development Series, Addison-Wesley (2010)
20. Schultheis, A., et al: An Overview and Comparison of Case-Based Reasoning Frameworks. In: 31st ICCBR. LNCS, vol. 14141, pp. 327–343. Springer (2023)
21. Schultheis, A., et al.: Explanation of Similarities in Process-Oriented Case-Based Reasoning by Visualization. In: 31st ICCBR. vol. 14141, pp. 53–68. Springer (2023)
22. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press (2004)
23. Smyth, B., Mullins, M., McKenna, E.: Picture perfect - Visualisation Techniques for Case-Based Reasoning. In: 14th ECAI. pp. 65–69. ECAI'00, IOS Press (2000)
24. Verma, D., Bach, K., Mork, P.J.: Similarity Measure Development for Case-Based Reasoning-A Data-Driven Approach. In: 3rd NAIS Symposium. CCIS, vol. 1056, pp. 143–148. Springer (2019)
25. Ward, M.O., Grinstein, G., Keim, D.: Interactive Data Visualization. CRC Press (2015)