FLEA-CBR – A Flexible Alternative to the Classic 4R Cycle of Case-Based Reasoning

Viktor Eisenstadt[1(✉)], Christoph Langenhan[2], and Klaus-Dieter Althoff[1,3]

[1] Institute of Computer Science, University of Hildesheim, Samelsonplatz 1, 31141 Hildesheim, Germany
ayzensht@uni-hildesheim.de, klaus-dieter.althoff@dfki.de
[2] Chair of Architectural Informatics, Technical University of Munich, Arcisstrasse 21, 80333 Munich, Germany
langenhan@tum.de
[3] German Research Center for Artificial Intelligence (DFKI), Trippstadter Strasse 122, 67663 Kaiserslautern, Germany

Abstract. This paper introduces FLEA-CBR, an alternative approach for composition of case-based reasoning (CBR) processes. FLEA-CBR extends the original 4R *(Retrieve, Reuse, Revise, Retain)* CBR cycle with a flexible order of execution of its main steps. Additionally, a number of combinatorial features for a more comprehensive and enhanced composition can be used. FLEA is an acronym for *Find, Learn, Explain, Adapt* and was initially created to solve the restrictiveness issues of case-based design (CBD) where many existing approaches consist of the retrieval phase only. However, the methodology can be transferred to other CBR domains too, as its flexibility allows for convenient adaptation to the given requirements and constraints. The main advantages of FLEA-CBR over the classic 4R cycle are the ability to combine and activate the main steps in desired or arbitrary order and the use of the explainability feature together with each of the steps as well as a standalone component, providing a deep integration of Explainable AI (XAI) into the CBR cycle. Besides the CBR methods, the methodology was also conceptualized to make use of the currently popular machine learning methods, such as recurrent and convolutional neural networks (RNN, ConvNet) or general adversarial nets (GAN), for all of its steps. It is also compatible with different case representations, such as graph- or attribute-based. Being a template for a distributed software architecture, FLEA-CBR relies on the autonomy of implemented components, making the methodology more stable and suitable for use in modern container-based environments. Along with the detailed description of the methodology, this paper also provides two examples of its usage: for the domain of CBR-based creativity and library service optimization.

Keywords: CBR cycle · Distributed CBR · Explainable AI · Adaptation · Case-based design · Artificial neural network · Software architecture

K. Bach and C. Marling (Eds.): ICCBR 2019, LNAI 11680, pp. 49–63, 2019.
https://doi.org/10.1007/978-3-030-29249-2_4

1 Introduction

In case-based reasoning (CBR), the historical development of the research area established the 4R cycle [2], that consists of the consecutive steps *Retrieve*, *Reuse*, *Revise*, and *Retain*, as the main underlying structure for implemented and theoretical approaches that explicitly identify themselves or were designated by researchers as CBR-based. Quickly after its introduction in 1994, the 4R cycle became a widely accepted synonym for case-based reasoning and is referenced nowadays in nearly every CBR-related research work.

Generally, two types of such CBR research works exist nowadays: *one-step-specific* and *generative application*. The works of the first type deal with only one of the R-steps (where Retrieve and Reuse make up the majority) and are rarely created for one specific domain, providing improvement of a universally applicable method. In contrast to the first type, the generative application of the CBR cycle is usually conceptualized for a specific domain (often including the related domains as well), taking the CBR cycle as a whole and using all of the R-steps to build a CBR-based solution for a specific problem of the domain. Sometimes this type is combined with other AI methods providing the so-called *hybrid* approaches. Both types have in common that they do not try to modify the complete CBR cycle and adapt it to own purposes, instead the 4R cycle is used "as is". These modifications, however, exist and make up the third, very rare, type of CBR-related research: the *explicit modifications* of the classic CBR cycle. The reasons for development of such modifications are different, but can mostly be narrowed to the (partial) incompatibility of the application domain in question to the original order of execution of 4R and/or to the restrictions of the domain that allow for execution of a subset of the four R-steps only.

In this work, we present *FLEA-CBR*, a methodology for modification of the original CBR cycle, whose aim is to bring more flexibility to the cycle structure and to offer an alternative underlying approach for systems where the four R-steps may not or cannot be applied subsequently in the original order. FLEA is an acronym for *Find*, *Learn*, *Explain*, and *Adapt*. These four components represent the main functionalities of the methodology. Main features of FLEA are the possibility of standalone usage of the components as well as the arbitrary combination of them and the deep integration of Explainable AI (XAI) into the cycle making it an equivalent step among the already existing ones. FLEA was conceptualized for the domain of case-based design (CBD), however, its flexibility allows for application to other domains as well. The only requirement is that the case representation types of the domain(s) are compatible with FLEA-CBR.

This paper is structured as follows: first, we present work related to the research topic of this paper, i.e., the modifications of the 4R CBR cycle. We show differences between hybrid CBR approaches and cycle modifications and present an overview of selected modifications. In the following Sect. 3, FLEA-CBR's core components and features are presented in detail. In Sect. 4, existing and theoretical example usages of FLEA-CBR are presented for the domains of CBR-based creativity and library service optimization respectively. Finally, an outlook to the future of the methodology concludes this work in Sect. 5.

2 Related Work

The explicit modifications of the original order of execution of the four R-steps have a long tradition in CBR, being present after the 4R cycle started to gain its popularity. However, it is important to mention that this rare type of CBR-related research should not be confused with the hybrid CBR approaches, i.e., those that combine CBR with other machine learning techniques or related methods. The examples of such hybrid approaches can be found in combination with deep learning [6], genetic algorithm optimization [21], or support vectors [11]. Additionally, an overview of such combinations [28] provides an entry point to start research into these approaches. The hybrid approaches mostly do not change or adapt the cycle itself, whereas the CBR cycle modifications explicitly edit the 4R cycle: for example, an additional step can be appended or prepended to the cycle, or one or more of the existing steps can be replaced by other steps, combined into a single step, or managed by an additional non-CBR module, such as an artificial neural network (ANN). The following Table 1 provides a comparison of selected CBR cycle modifications that either explicitly identify themselves as a 'modification' or provide enough evidence to count as such, i.e., edit the cycle as described above. The approaches that were proposed as modified CBR cycles, but in fact are hybrid systems, were not considered. Besides the descriptions of the respective approaches and the corresponding cycle modifications, this comparison also provides information on suitability of the approach for universal use and if it implements explainability features.

Table 1. An overview of selected CBR cycle modifications. *U* stands for the universal type of use, *DB* for domain-bound use. XAI stands for explainability/explainable AI.

Description	Publication	Modification	U/DB	XAI
CBR-based recipe recommender system	Skjold and Øynes [36]	Second Retrieve in ephemeral case base betw. Reuse and Revise	DB	No
Recipe generation with CBR and deep learning	Grace et al. [16]	Dual-cycle CBR	DB	No
CBR real-time planning for industrial systems	Navarro et al. [25]	Starts with learning (Revise + Retain)	DB	No
Decision support with neocortex imitation	Hohimer et al. [18]	Exec. order: *Reason*, Retain, *Review*, Revise	U	No
Tagging + retrieval of similar code passages	Roth-Berghofer and Bahls [34]	*Explain* + *Customize* replace Reuse + Revise	DB	Yes
Reorganization of the case bases	Finnie and Sun [14]	*Repartition* betw. Reuse and Revise	U	No
Maintenance of CBR systems	Reinartz et al. [29]	*Review* and *Restore* after Retain	U	No
Oceanography forecasts with CBR and ANN	Lees and Corchado [23]	ANN governs Reuse and Revise	DB	No

Interpreting the Table 1, we can conclude that certain similarities between hybrid approaches and cycle modifications exist, in some cases it is also hard to tell if the given approach is of hybrid type or an explicit modification. Both types are mostly used for a specific domain or task. The universal approaches contained in Table 1, such as the neocortex imitation [18], the intermediate reorganization of case bases [14], or the 6R cycle [29] that adds the maintenance steps Review and Restore to 4R, build the minority. Another example is the real-time intelligent decision support [12], which, however, was not included in the comparison, as it makes the human operator responsible for Retain and provides no adaptation, restricting the CBR process to the retrieval phase only.

The obvious problem of cycle modifications is that no methodology exists that is universal as well as flexible in terms of execution order. Other problem is the lack of XAI features. The only exception is the approach [34] that, however, provides explanations only for results of the retrieval phase.

3 FLEA-CBR

This section describes the FLEA-CBR methodology in detail and compares it to the 4R cycle in terms of flexibility and universality of use. First, the problem of non-flexibility of the 4R cycle will be described, after that an overview and each of the R-steps will be compared to its approximate counterpart of FLEA-CBR.

3.1 Problem Description

During the last decades, the 4R cycle (see Fig. 1) became an ubiquitous part of case-based approaches that mostly improve one of the R-steps or apply the cycle to the given domain to produce a complete CBR-based solution to the given problem space (see also Sect. 1). However, one of the main problems of the 4R cycle is that it requires the consecutive execution of the R-steps in order to work properly and to be used for the problem space as a suitable solution approach. This problem also precludes the 4R cycle from being more universal and forces the applications to follow the only available order (Retrieve → Reuse → Revise → Retain) and to be non-flexible in terms of selection of the currently required step. This issue is also valid for the contemporary alternatives to the 4R cycle, such as the cycles presented by Hunt [20] or in Leake's work [22]. Changing the order of execution might result in reassessment of the complete approach and in the subsequent decision of the developers of the system to use methods other than case-based. A number of modified execution orders and the nevertheless existing problems and issues were already listed in Sect. 2.

A major technical flexibility problem of the 4R cycle and the majority of its modifications is that the retrieval step is required in all application cases and all other steps depend on its results. While many problems of many domains can be solved with this order and restrictions (for example, in the classic domains of CBR, such as mechanical engineering tools diagnosis or medical applications [19]), other domains, especially those that make use of creativity (for example,

architecture or game development), may require other, more complex and custom orders that may not look reasonable to humans but are completely understandable and executable by the machines.

Another main structure-related issue is that, while being simulated from human thinking and reasoning abilities, the 4R cycle and its contemporary alternative cycles also follow the human methods of experience-based decision making. This is reasonable if the computer should make decisions that the humans can relate to, but computers are generally more flexible in that regard and should not be restricted to one order only and should be able to decide autonomously which order is the most suitable for the current problem space. Therefore, especially case-based reasoning should provide more high-level features, for example to mix the steps and allow for repeating of some, if necessary.

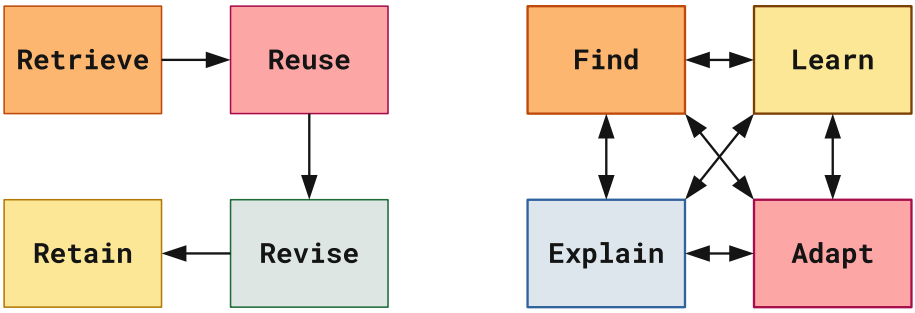

Fig. 1. FLEA-CBR in direct comparison with the 4R cycle.

3.2 Overview and Background

To overcome the issues described above and to make it possible for affected domains to make more use of CBR, we developed the methodology FLEA-CBR that introduces flexibility to the usage of the CBR features and allows for *mixing*, *repeating*, *sequencing*, or *cycling* (further referred as FLEA's *core features*) of the steps *Find*, *Learn*, *Explain*, and *Adapt* (further referred as FLEA's *core components*). FLEA-CBR was created in research context of MetisCBR [7], a framework for distributed case-based decision support during the early conceptual phases in architecture, that can find similar spatial configurations and contextually explain the retrieval results, suggest the next design step, and evolve the current design state to show how it can look in the future.

In the following sections the four core features of FLEA-CBR are described first, after that the four core components are presented. In Fig. 1, a high-level overview of FLEA-CBR and its direct comparison to the 4R cycle are available. Figure 2 demonstrates examples of core feature usages. Figures 3 and 4 contain feature usages for real domains.

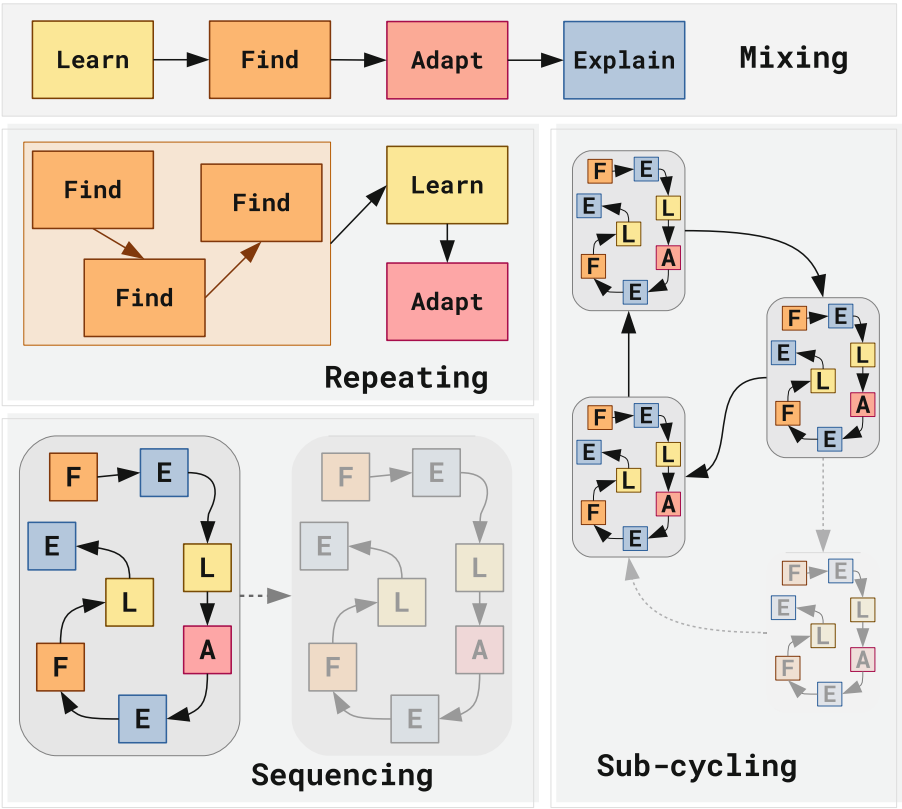

Fig. 2. Core features of FLEA-CBR.

3.3 Core Features

Mixing. This feature is crucial for FLEA-CBR's flexibility advantage as it allows for breaking the original 4R order and mix the mains steps in the order that is necessary for the domain or just for trying out and find the best option. Mixing can be done by a human operator as well as by a machine, e.g., a 'parent' CBR cycle. In Sect. 4, all examples are mixed and demonstrate how this action provides the FLEA-CBR implementation with the flexibility advantage.

Repeating. The goal of this functionality is to allow for continuous re-execution of a step with a predefined or arbitrary count of executions, e.g., to try other configuration options or different methods within one cycle run. This feature makes the CBR process with FLEA more fine-grained and exact, but also versatile in terms of selection of the most suitable method. This core feature can be executed with specific high-level frameworks, such as Keras [9] for deep learning.

Sequencing. This feature introduces the possibility of summarizing of (repeated) steps into sequences that can be used for similar domains or tasks. Unlike repeating, the sequences are more symbolic, i.e., do not contain information about exact methods or approaches applied, and can consist of different steps. This gives the system designers freedom to decide which methods are the most suitable, but preclude them from changing the execution order, transferring only the structure of the sequence. Along with Mixing, Sequencing bears a close resemblance to the original 4R cycle.

Sub-cycling. This feature introduces FLEA-CBR sub-cycles that can be used to encapsulate certain sequences and use the final outcome only in the parent cycle. Sub-cycles' goal is to increase the likelihood of possibly useful outcomes that, however, might be helpful or reused *later* in the parent process. That is, the sub-cycles can be skipped or executed asynchronously. Furthermore, sub-cycles can also be configured by the parent cycle. Technically, sub-cycles were added to the FLEA-CBR features in order to add a higher compatibility to the modern microservices-based software architectures. They can be executed in a dedicated container with methods that totally differ from other sub-cycles in terms of execution time or system resources usage.

3.4 Find

FLEA-CBR's *Find* step is the approximate equivalent of the *Retrieve* phase of the 4R cycle. Its main task is to search in the given collection of the domain cases (the case base) for entries with the highest similarity to the received problem description (query). However, generally, Find differs from Retrieve in the way that it can receive queries or requests from other FLEA-CBR modules or sub-cycles and forward them to modules other than Reuse in the format suitable for this module or sub-cycle.

Beyond the retrieval of the most similar cases, Find can also be used for looking for contextual connections between a subset of the data collection entries in order to provide the user or a FLEA-CBR application with information about how these entries, i.e., cases, are related to each other. In theory, it is also possible to integrate Find as a simple full text or image search engine to perform a common precision-recall-based information retrieval process, however, the modules that receive the results should be able to work with them, e.g., infer similarity values when necessary. All the retrieval methods described above can be combined and/or executed consecutively using the Repeating feature.

Context of Usage. When used as an intermediate step between other steps or modules, it is advisable to provide the FLEA-CBR sequence that contains Find with information on the execution context, e.g., the domain, in order to select the most suitable retrieval or matching method. This information can be received from the Learn or Explain component. Similarly to Retrieve, Find can also be used as an entry point to the case-based reasoning process.

3.5 Learn

The *Learn* step of the methodology was developed to enrich the FLEA-CBR-based approaches and applications with abilities of making reasonable conclusions based on features of the cases and predicting contextual and relational connections between them, but also to implement the retaining techniques that govern how and why the cases should be saved in the application's case collections. Therefore, the Learn step can also be seen as an approximate further development of 4R cycle's Retain and a high-level interface for gathering and combining the learning strategies and case preservation methods. To accomplish these tasks, Learn is conceptualized to make use of the modern deep learning methods, such as recurrent neural networks based on LSTM [17] or GRU [8], convolutional neural networks, or Generative Adversarial Networks (GAN) [15]. However, the established CBR learning methods such as Instance-Based Learning [4] can be used as well. A combination by using a subset of these technologies is possible too, e.g., in the Repeating feature (see Sect. 3.2).

Context of Usage. Learn can be used in context of classification of cases into different categories for later saving in this category's case base (if the application or approach make use of distributed architecture with multiple case bases). It can also be used in the CBRs usual context of judging if the case is save-worthy or not. On application's entry point level Learn can separate the initial case set into different clusters based on separations conducted for similar case sets. Intermediately, Learn can function as a bridge between other modules (i.e., activated each time after Find, Explain, or Adapt) to draw conclusions from the results of the respective step and inform the subsequent module about its findings in order to influence its behavior and produce a more reasonable outcome.

3.6 Explain

The current development of the research area of artificial intelligence emphasizes the increasing requirement of the AI systems to include a component or module that implements an algorithm that is able to explain the behavior and decisions of the system during the current operation. Based on this increasing need, a research area of Explainable AI (XAI) has emerged and resulted in a number of corresponding research initiatives, such as the well-known DARPA Explainable AI program and a multitude of thematic workshops, such as IJCAI XAI workshops [5] or the XCI workshop [27]. One of the most recent research works, a comprehensive guide to XAI in machine learning applications by Molnar [24], shows the future directions of XAI and provides instructions and best practices for its implementation. Furthermore, a survey on XAI [3] can be examined.

In case-based reasoning, the requirement for proper explanations was examined before the 4R cycle was first introduced [1]. However, the development of the XAI area in CBR concentrated mostly on simplistic explanations (i.e., the why- and how-explanations), as pointed out in foundational issues of explanations in CBR [33]. The cognitive explanations, that extend their explanation space

beyond the simplistic questions and whose task is, inter alia, to provide answers to the questions of relations between results of operations (e.g., retrieval) or contextual connections between concepts, were rarely implemented. This problem extends to the most systems and approaches from other AI domains. Being very complex systems (which especially relates to the decision support systems), they are not always able to explain their behavior to the user. The reasons for this are either the non-transparent reasoning process, which could be implemented on purpose this way (e.g., in a black-box API), or a system design that does not allow for explanation interfaces to be implemented and/or connected. The latter problem is more common: many systems were constructed without having in mind the importance of an explicit explanation module, i.e., conceptualized and implemented before XAI emphasized the importance of such functionalities.

With FLEA-CBR's *Explain*, we intend to provide the systems that use the methodology as their underlying structure, with a high-level approach that can combine, select, and apply different types of explanations in order to compose an algorithm that can give insights into the behavior of this system. *As a standalone component, Explain is the main structural difference between 4R and FLEA-CBR.* Its inclusion in the methodology as a particular step of the reasoning process is based on the XAI importance described above. The main goal of Explain is to build connections that can provide a structure for exchange of explainable data between the modules and sequences of a FLEA-CBR implementation and the user. In particular, we differentiate between the *internal* and *external* types of explanations that can be handled and produced by Explain:

- *Internal* – This type of explanations is intended for internal use between the components. The data transferred within these explanations does not have to be human-readable, as its goal is to provide the receiving module with a foundation for its reasoning process.
- *External* – The explanation data of this type is delivered to the user providing her with the corresponding information. It is up to the system designer as well as depends on the domain how the explanations will be presented. Many of the systems use textual explanations (e.g., Roth-Berghofer and Bahls [34], see Sect. 2), however, graphical or audiovisual explanations can be produced too. The most important fact to have in mind is that XAI is a matter of user experience and system usability too, therefore, explanations should be designed in a way that is most familiar to the target group.

Both explanation types can be *final* as well as *temporal*. The final explanations provide a finite state where the expression or data cannot be modified anymore and represents the final output of the *Explain* module implementation. This type can be used for static modules that do not update their state with new results, i.e., do not run iteratively to improve the results, e.g., use only the Mixing feature (see Sect. 3.3). In contrast to the final ones, the temporal explanations can be changed over time and are more suitable for systems that are able to run iteratively or apply a number of concurrent processes. Hence, the temporal type allows for continuous update of explanations making the systems

more dynamic and their mode of operation more asynchronous. Such approaches would most likely use either Sequencing or Repeating features for their systems.

Context of Usage. The necessity to use Explain can be found at any point in the runtime of the system where a component needs to send or receive data required for justification or transparency of its actions. In structural CBR systems, for example, local similarities on the attribute level can be used to produce a detailed transparency report that makes clear how the overall similarity value for the case has been calculated. This report can be filled iteratively with temporal explanations constructed for different similarity measures. In textual CBR systems, Explain can be used for labeling of the most important text passages providing, for example, the Learn module with information about those passages in order to learn the class of the text or transfer the labeling information to the current domain. As a standalone component, Explain can also be used separately, e.g., to explain differences between selected cases.

3.7 Adapt

Adaptation of selected cases, being a core feature of case-based reasoning, has also found its way to FLEA-CBR. Similarly to Find and Learn, the step *Adapt* inherits the original task of its approximate 4R equivalent Reuse, i.e., the modification of cases according to the adaptation rules, and extends it with additional operations to catch up with the currents trends in AI. Adapt unifies methods for both transformation-based and generative adaptation [37] together with the data augmentation, completion, and generation approaches.

Additionally, Adapt is responsible for automatic revision of transformed cases, i.e., it imitates the Revise step of the 4R cycle in order to rate the helpfulness of adaptation and to decide if the transformation has a potential to solve the given problem. For this special case, a combination of Adapt and Explain seems reasonable: with Explain's abilities to construct a human-readable expression an explanation can be delivered to the user informing her about possible solution transformation and application.

Context of Usage. Besides the classic usages of Reuse named above, Adapt can be used for the currently very frequently executed tasks of data augmentation and data generation. Both tasks are required in applications where the available amount of data is small and can be extended and/or enhanced with augmentation or generation to produce the amount required for application to work properly or to be able to conduct experiments with a prototype of the approach.

4 Example Usages

The following examples intend to demonstrate how the dynamic structure of FLEA-CBR would (in theory and practice) outperform the static structure of 4R on problem spaces that are suitable for solving with the CBR methods.

4.1 CBR and Creativity

The first example and at the same time the showcase of FLEA-CBR is the problem space of the *CBR and creativity* domain. Among the CBR community, this notation normally subsumes the application of CBR to the human creativity-related tasks, such as architecture [31] and computer-aided design (CAD) [26] (both are usually summarized as case-based design, CBD), cooking [10], knitting [30] or similar domains. The common problem space of all these sub-domains of CBR-based creativity can be narrowed to the following characteristics:

1. *The complexity of cases* – Mostly, the cases available in the case bases of the creativity domain applications and approaches have a complex, very often nested, attribute-value-based structure. Knowledge described with these cases is comprehensive and many cases differ only in small details. An example of such case is a CAD model that consists of many parts combined together or a floor plan with many differently shaped rooms. As a result of such complexity, the 4R cycle may take a very long time to finish its process.
2. *Adaptation is not available* – The use of CBR knowledge containers (case bases, adaptation knowledge, similarity measures, vocabulary) [32] qualifies an approach as CBR-based. However, quite often the adaptation knowledge is not available in CBR creativity applications due to the lack of the adaptation feature. Such approaches are often restricted to retrieval only leaving the user without creative solution recommendations or an adaptation to the styles of other designers (style transfer).

FLEA-CBR's ability to run tasks in parallel and partially can solve such problems without being modified or adapted specifically, the only important fact to have in mind is that a suitable configuration of FLEA-CBR is required.

For example, using a configuration with the Sub-cycling feature, the FLEA-CBR-based application can divide the execution of the whole cycle into separate sub-cycles in order to provide each of these sub-cycles with a part or a layer of the complex case. The sub-cycles would run concurrently and asynchronously and speed up the process, without waiting for the previous process to finish. For each part or layer a modification of the case into other representation can be provided as well using it as input for an adaptation approach that can work with this type of data. More specifically, the Generative Adversarial Nets (GAN) [15], as an artificial neural network type that was specifically created for generation of objects that would appear real to a human and also accepted as such, can be used as an adaptation approach for creativity cases, provided that it receives input data in the proper format. As adaptation knowledge the data transferred with Learn from other domains and accepted generated objects can be used. Additionally, Learn can suggest the proper next actions of the current creativity process. In Fig. 3, an example implementation of FLEA-CBR for the creativity domain is shown. This example is based on the only existing and evaluated [13,35] implementation of FLEA-CBR: namely, in the framework MetisCBR for support of floor plan design process (see also Sect. 3.2). The figure demonstrates the current state and the planned further development of the framework.

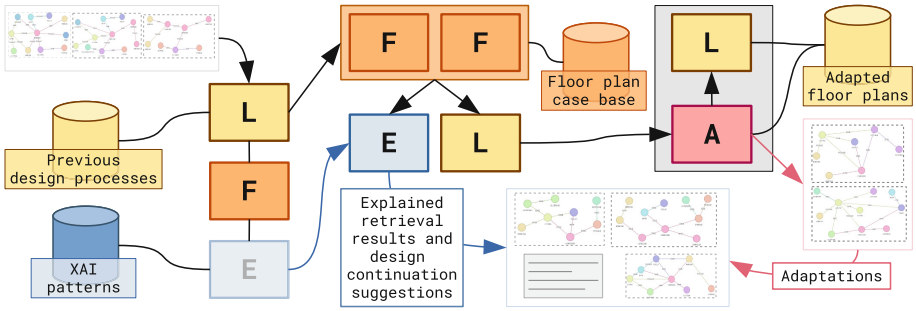

Fig. 3. FLEA-CBR implementation in MetisCBR [7]. The modules that are currently in early stage of development are represented with a slight transparency. The dark gray group with Learn and Adapt, that produces possible design evolutions with GAN, represents the Sub-cycling feature. The double Find group represents Repeating.

4.2 Library Service Optimization

The second example demonstrates how the optimization of the scientific library service process can be achieved with a FLEA-CBR-based software architecture. For demonstration purposes, an everyday process of recording of bibliographic items, e.g., scientific monographies, and the following item borrowing process from the library's stock will be used. We do not assume a specific case representation, but the case itself, i.e., the bibliographic item to be recorded, can be incomplete or in foreign language. The general tasks of the planned system in this particular case can be summarized as follows:

1. *Automatic translation of the item* – If the item is not in the language(s) of library's country then it should be translated for correct keyword extraction and, if required, transliterated to be readable by the library staff and users.
2. *Looking for similar items in case bases of items* – This task subsumes the search for similar items and the subsequent completion of the given item, if some information is missing.
3. *Entry and search in the knowledge graph* – Usually, a catalogue search would be used instead, however, taking the current scholarly information processing trends into account, we replace it with a knowledge graph.

Additionally, the XAI features, that provide the results of the tasks named above with human-understandable justifications to receive confirmations or corrections from the user and to proceed with the next step, can also be used.

The task of the FLEA-CBR implementation is to guide the human operator during the item recording process: suggest the best way of transliteration based on the item's language, fill up the missing information based on similar previous cases, insert information into the knowledge graph, suggest catalogue categories and a signature, and track the item's status (borrowed, not returned etc.). Figure 4 demonstrates the possible FLEA-CBR configuration for this use case.

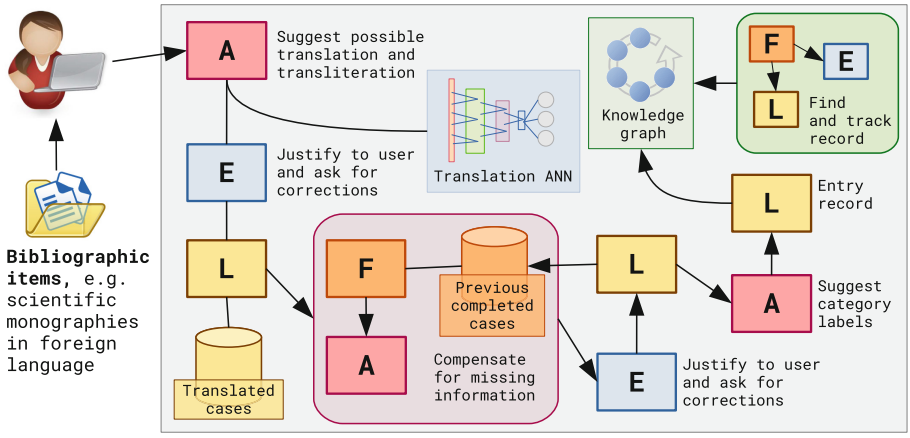

Fig. 4. FLEA-CBR configuration example for a library service optimization. The pink group represents the Sequencing feature where information about incomplete parts can be transferred from other librarian records categories (e.g., periodical publications). The green group represents Sub-cycling, i.e., searching and status tracking of the record. (Color figure online)

5 Conclusion and Future Work

In this paper, we presented FLEA-CBR, an alternative approach to the well-established 4R (Retrieve, Reuse, Revise, Retain) cycle. FLEA-CBR's core components Find, Learn, Explain, and Adapt can be executed in order desired or suitable for the current domain or task, unlike the 4R steps that require the execution in the predetermined order, starting with Retrieve. FLEA-CBR's goal is to solve this flexibility issue, that the 4R itself and its contemporary alternatives and modifications possess. Furthermore, the FLEA methodology makes the XAI component an equivalent and ubiquitous part of the CBR cycle. Additionally, FLEA-CBR offers a number of features that extend the flexibility of the cycle: Mixing, Sub-cycling, Sequencing, and Repeating. The suitability of FLEA-CBR for use as the underlying structure for CBR approaches was demonstrated in an evaluated existing approach for CBR-based creativity and a theoretical application for the library service domain. Future plans for FLEA-CBR include, for example, an implementation of a symbolic programming framework for prototyping of applications that are based on the methodology and conducting of new experiments for existing implementations to further confirm the methodology's suitability for use as the underlying software architecture.

References

1. Aamodt, A.: Explanation-driven case-based reasoning. In: Wess, S., Althoff, K.-D., Richter, M.M. (eds.) EWCBR 1993. LNCS, vol. 837, pp. 274–288. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58330-0_93

2. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–59 (1994)
3. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
4. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6**(1), 37–66 (1991)
5. Aha, D., Darrell, T., Pazzani, M., Reid, D., Sammut, C., Stone, P.: IJCAI-17 workshop on explainable AI (XAI). In: IJCAI-17 Workshop on Explainable AI (XAI) (2017)
6. Amin, K., Kapetanakis, S., Althoff, K.-D., Dengel, A., Petridis, M.: Answering with cases: a CBR approach to deep learning. In: Cox, M.T., Funk, P., Begum, S. (eds.) ICCBR 2018. LNCS (LNAI), vol. 11156, pp. 15–27. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01081-2_2
7. Ayzenshtadt, V., Langenhan, C., Bukhari, S.S., Althoff, K.D., Petzold, F., Dengel, A.: Thinking with containers: a multi-agent retrieval approach for the case-based semantic search of architectural designs. In: Filipe, J., van den Herik, J. (eds.) 8th International Conference on Agents and Artificial Intelligence (ICAART-2016), Rome, Italy, 24–26 February. SCITEPRESS (2016)
8. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
9. Chollet, F., et al.: Keras: Deep learning library for theano and TensorFlow (2015). https://keras.io
10. Cordier, A., et al.: Taaable: a case-based system for personalized cooking. In: Montani, S., Jain, L. (eds.) Successful Case-Based Reasoning Applications-2, pp. 121–162. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-38736-4_7
11. De Paz, J.F., Bajo, J., González, A., Rodríguez, S., Corchado, J.M.: Combining case-based reasoning systems and support vector regression to evaluate the atmosphere-ocean interaction. Knowl. Inf. Syst. **30**(1), 155–177 (2012)
12. Eremeev, A.P., Vagin, V.N.: Common sense reasoning in diagnostic systems. In: Efficient Decision Support Systems-Practice and Challenges From Current to Future. IntechOpen (2011)
13. Espinoza-Stapelfeld, C., Eisenstadt, V., Althoff, K.-D.: Comparative quantitative evaluation of distributed methods for explanation generation and validation of floor plan recommendations. In: van den Herik, J., Rocha, A.P. (eds.) ICAART 2018. LNCS (LNAI), vol. 11352, pp. 46–63. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05453-3_3
14. Finnie, G., Sun, Z.: R5 model for case-based reasoning. Knowl.-Based Syst. **16**(1), 59–65 (2003)
15. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
16. Grace, K., Maher, M.L., Wilson, D.C., Najjar, N.A.: Combining CBR and deep learning to generate surprising recipe designs. In: Goel, A., Díaz-Agudo, M.B., Roth-Berghofer, T. (eds.) ICCBR 2016. LNCS (LNAI), vol. 9969, pp. 154–169. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47096-2_11
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
18. Hohimer, R., Greitzer, F.L., Noonan, C.F., Strasburg, J.D.: Champion: intelligent hierarchical reasoning agents for enhanced decision support. In: STIDS, pp. 36–43 (2011)
19. Holt, A., Bichindaritz, I., Schmidt, R., Perner, P.: Medical applications in case-based reasoning. Knowl. Eng. Rev. **20**(3), 289–292 (2005)

20. Hunt, J.: Evolutionary case based design. In: Watson, I.D. (ed.) UK CBR 1995. LNCS, vol. 1020, pp. 17–31. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60654-8_19
21. Kim, S., Shim, J.H.: Combining case-based reasoning with genetic algorithm optimization for preliminary cost estimation in construction industry. Can. J. Civ. Eng. **41**(1), 65–73 (2013)
22. Leake, D.B.: Case-Based Reasoning: Experiences, Lessons and Future Directions. MIT Press, Cambridge (1996)
23. Lees, B., Corchado, J.: Neural network support in a hybrid case-based forecasting system. In: Case-Based Reasoning Integrations, pp. 85–90 (1998)
24. Molnar, C.: Interpretable Machine Learning (2019). https://christophm.github.io/interpretable-ml-book/
25. Navarro, M., De Paz, J.F., Julián, V., Rodríguez, S., Bajo, J., Corchado, J.M.: Temporal bounded reasoning in a dynamic case based planning agent for industrial environments. Expert Syst. Appl. **39**(9), 7887–7894 (2012)
26. Ociepka, P., Herbuś, K.: Application of the CBR method for adding the process of cutting tools and parameters selection. In: IOP Conference Series: Materials Science and Engineering, vol. 145, p. 022029. IOP Publishing (2016)
27. Pereira-Fariña, M., Reed, C.: Proceedings of the 1st workshop on explainable computational intelligence (XCI 2017). In: Proceedings of the 1st Workshop on Explainable Computational Intelligence (XCI 2017) (2017)
28. Prentzas, J., Hatzilygeroudis, I.: Combinations of case-based reasoning with other intelligent methods. Int. J. Hybrid Intell. Syst. **6**(4), 189–209 (2009)
29. Reinartz, T., Iglezakis, I., Roth-Berghofer, T.: Review and restore for case-base maintenance. Comput. Intell. **17**(2), 214–234 (2001)
30. Richards, P., Ekárt, A.: Supporting knitwear design using case-based reasoning. In: Proceedings of the 19th CIRP Design Conference-Competitive Design. Cranfield University Press (2009)
31. Richter, K.: Augmenting Designers' Memory: Case-based Reasoning in Architecture. Logos-Verlag, Berlin (2011)
32. Richter, M.M.: Knowledge containers. In: Readings in Case-Based Reasoning. Morgan Kaufmann Publishers, San Francisco (2003)
33. Roth-Berghofer, T.R.: Explanations and case-based reasoning: foundational issues. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 389–403. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28631-8_29
34. Roth-Berghofer, T.R., Bahls, D.: Code tagging and similarity-based retrieval with myCBR. In: Bramer, M., Petridis, M., Coenen, F. (eds.) International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 19–32. Springer, London (2008). https://doi.org/10.1007/978-1-84882-171-2_2
35. Sabri, Q.U., Bayer, J., Ayzenshtadt, V., Bukhari, S.S., Althoff, K.D., Dengel, A.: Semantic pattern-based retrieval of architectural floor plans with case-based and graph-based searching techniques and their evaluation and visualization. In: 6th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2017), Porto, Portugal, 24–26 February 2017 (2017)
36. Skjold, K., Øynes, M.S.: Case-based reasoning and computational creativity in a recipe recommender system. Master's thesis, NTNU (2017)
37. Wilke, W., Bergmann, R.: Techniques and knowledge used for adaptation during case-based problem solving. In: Pasqual del Pobil, A., Mira, J., Ali, M. (eds.) IEA/AIE 1998. LNCS, vol. 1416, pp. 497–506. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-64574-8_435