Supporting Architectural Design Process with FLEA

A Distributed AI Methodology for Retrieval, Suggestion, Adaptation, and Explanation of Room Configurations

Viktor Eisenstadt[1], Christoph Lanhgenhan[2], Klaus-Dieter Althoff[1,3]

[1] University of Hildesheim, Institute of Computer Science
Samelsonplatz 1, 31141 Hildesheim, Germany
{ayzensht,althoff}@uni-hildesheim.de

[2] Technical University of Munich, Chair of Architectural Informatics
Arcisstrasse 21, 80333 Munich, Germany
langenhan@tum.de

[3] German Research Center for Artificial Intelligence (DFKI)
Trippstadter Strasse 122, 67663 Kaiserslautern, Germany

Abstract. The artificial intelligence methods, such as case-based reasoning and artificial neural networks were already applied to the task of architectural design support in a multitude of specific approaches and tools. However, modern AI trends, such as Explainable AI (XAI), and additional features, such as providing contextual suggestions for the next step of the design process, were rarely considered an integral part of these approaches or simply not available. In this paper, we present an application of a distributed AI-based methodology FLEA (Find, Learn, Explain, Adapt) to the task of room configuration during the early conceptual phases of architectural design. The implementation of the methodology in the framework MetisCBR applies CBR-based methods for retrieval of similar floor plans to suggest possibly inspirational designs and to explain the returned results with specific explanation patterns. Furthermore, it makes use of a farm of recurrent neural networks to suggest contextually suitable next configuration steps and to present design variations that show how the designs may evolve in the future. The flexibility of FLEA allows for variational use of its components in order to activate the currently required modules only. The methodology was initialized during the basic research project Metis (funded by German Research Foundation) during which the architectural semantic search patterns and a family of corresponding floor plan representations were developed. FLEA uses these patterns and representations as the base for its semantic search, explanation, next step suggestion, and adaptation components. The methodology implementation was iteratively tested during quantitative evaluations and user studies with multiple floor plan datasets.

Keywords: Room configuration, Distributed AI, Case-based reasoning, Neural networks, Explainable AI

1 Introduction

Current trends of information technology established artificial intelligence (AI) as one of the most ubiquitous techniques for decision and creativity support in different business and research areas. Architecture, being an interdisciplinary domain, i.e., active in both business and research, is not an exception: continuously increasing complexity and the industrial digitization of the architectural design process require consistent modernization and consolidation of methods that support creativity in form of digital assistance during the inspiration and exploration phases. Still, many approaches that implement the recent AI trends, such as convolutional and recurrent neural networks (CNN, RNN), are in the process of research and not yet ready to be used in the daily design process. However, many of them showed potential for standalone application or integration in the existing computer-aided architectural design (CAAD) software.

This paper presents a novel AI-based methodology *FLEA* (named after its four main steps/components: *Find*, *Learn*, *Explain*, *Adapt*) and its implementation for the phase of creating a room configuration: an essential process of the initial design phase that is responsible for the basic setup of the building design as it influences the later utilization and interior of the currently designed architectural unit. The goal of FLEA is to inspire and guide the designer during the early conceptual phase by providing her with a collaborative assistance system in order to create a proper room layout for the design task at hand.

The key advantage of FLEA is its high flexibility: each main component can be decoupled and used separately, without being dependent on other modules, provided that the data required for the component to work is available. A combination that consists of a subset of the main components is possible as well. Each main component itself possesses a certain grade of flexibility too, e.g., a conditional choice of the most suitable neural network to suggest the next step. The methodology was implemented as the underlying structure for the mode of operation of *MetisCBR* [7], a framework for support of the early design phases in architecture. MetisCBR uses methods of case-based reasoning (CBR), multi-agent systems (MAS), and artificial neural networks (ANN). The architectural design case in the framework is represented as a *graph-based room configuration* that uses *semantic fingerprints of architecture* [21] for description with well-known architectural concepts (see Fig. 1 and also Fig. 3). This paper describes the current status of FLEA's integration in the framework (see Fig. 2).

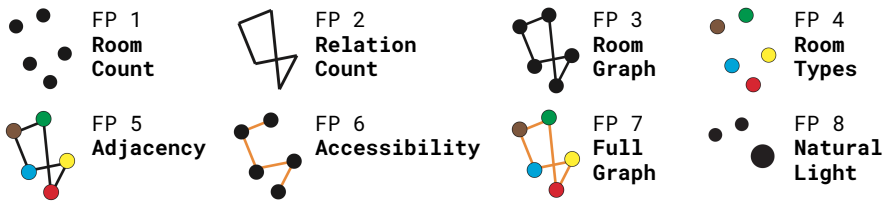

Fig. 1. Semantic fingerprints of architecture currently implemented in MetisCBR.

2 Related Work

In the last decades, many research initiatives from the AI-related domains developed approaches to support the early phases of the architectural design process. During the early years of CBR, a multitude of design-related case-based systems established case-based design (CBD) as an essential part of the entire CBR domain. The approaches, such as ARCHIE [30], FABEL [29], or, later, DIM [19] and CBArch [9], provided different variations of case-based techniques to examine how CBR can improve the design process by recommending, modifying, or explaining the floor plans and their corresponding (semantic) representations. Richter's work [23] summarizes the activity of CBR in the CAAD domain and provides an overview of problems that the case-based approaches need to solve or keep in mind if the acceptance of CBR among the representatives of the architecture domain should be increased. Among these problems are the optimization of retrieval strategies and the lack of variability and flexibility in the systems.

From the distributed AI perspective, the MAS-related research provided the most influential approaches for development and enhancement of the FLEA methodology. The work of Anumba et al. [3] examines research and technical foundations of use of multi-agent systems in construction and architecture domains. Gerber et al. [16] developed a research framework for prototyping of agents for support of generative parametric design process for the tasks of, e.g., façade generation or shell form finding. Simeone et al. [26] present an agent-based simulation and modeling approach for building design construction that enhances the Building Information Modeling (BIM) structure by extending it with agents that take into account the building's future users and use. Chu et al. [11] use agents to create a system for collaborative distributed 3D design: agents run on the server as well as on the user side, negotiate the design parameters, and generate the geometric model based on these parameters.

The research domain of deep learning in form of ANNs extended its activities for the CAAD domain as well and provided a number of related approaches during the last years. One of these approaches, DANIEL (Deep Architecture for fiNdIng alikE Layouts), uses deep learning for extraction of semantic features from floor plan images and applies convolutional neural networks for retrieval of similar building layouts [25]. An approach that combines case-based reasoning strategies with multi-agent systems and artificial networks was developed by González-Briones et al. for optimization of energy management in office buildings [17]. These two approaches are structurally the closest ones to FLEA and its implementation in MetisCBR, however, they are either conceptualized for one phase only, e.g., retrieval, or do not deal with the room configuration phase.

3 FLEA Methodology

Case-based reasoning, being a versatile knowledge-based technique, can be used to cover a number of phases of the design process and many tasks during these phases. For example, CBR can be used in early conceptual phases as well as in

the cost calculation phase. For the former, CBR can offer the designer a collection of structurally similar past designs, pick the features that should be adapted in the current design, or create a variety of design recommendations based on the current design and the similar designs. For the latter, CBR-based approaches can search for past designs with similar features, present the cost for each of these designs, and estimate the budget for development of the current design. The accomplishment of such tasks is possible thanks to the human-alike reasoning structure of CBR's 4R-cycle [1] that includes the phases *Retrieve*, *Reuse*, *Revise*, and *Retain*. By combining CBR with distributed AI, it is also possible to distribute the R-steps among autonomous agents that assume responsibility for one or more tasks and collaborate to solve the given common problem.

However, a completely collaborative process with voluntary participation of autonomous agents is not possible with 4R, as the current step always depends on the results of the previous step. This also means that combinations of some of the steps can't be made flexible: e.g., retrieval is always required. Furthermore, for the design domain, the problem of CBR-specific separation of case in *problem and solution* exists as well: usually, the problem is described as a failure or a (feature-based) search request for which the most similar case from the case base should be retrieved, its solution adapted and implemented, and, if positively revised, saved in the case base. Designs, however, usually do not contain explicitly described failures, and reduction to a set of abstract features does not represent its complex structure and so is not suitable for use as a query for structurally similar designs and the adaptation. The approaches that used the structure of an architectural design as query [2], mostly contain the retrieval step only.

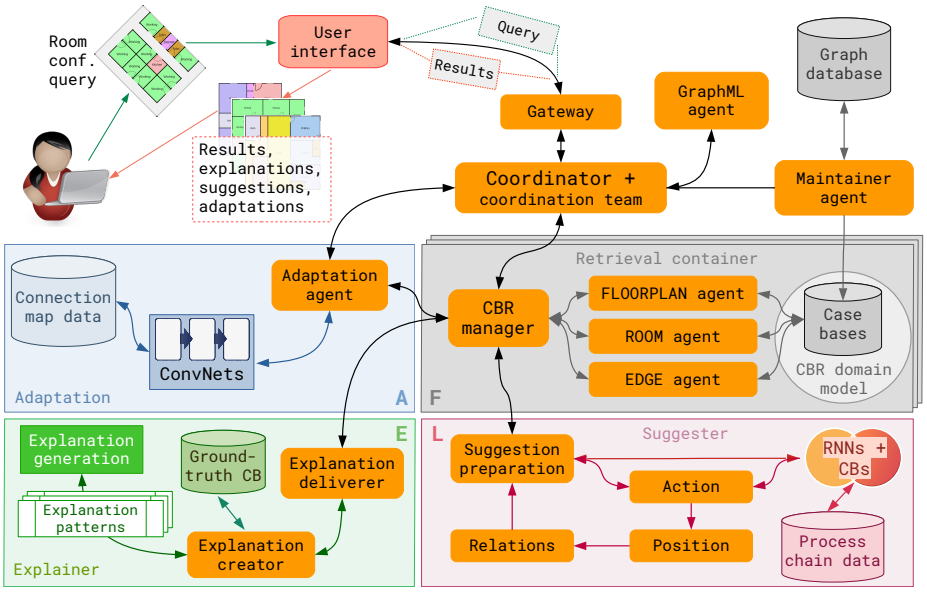

Fig. 2. The current implementation of FLEA in MetisCBR.

To overcome these shortcomings, we developed FLEA, a more flexible alternative to the classic CBR cycle. FLEA was created specifically for the architectural design domain, however, it can also be applied to other domains that provide a similar case structure. FLEA's main task is to provide a flexible assistance methodology to support the early phase of the design process by recommending inspirational similar designs (*Find*, see Sect. 3.1) and contextually suitable next steps of the process (*Learn*, see Sect. 3.2), explaining why and how a design can be helpful (*Explain*, see Sect. 3.3), and adapting the design by generating varieties of alternatives (*Adapt*, see Sect. 3.4). The FLEA methodology deliberately does not provide a strict execution sequence of its steps: unlike the 4R cycle, steps of FLEA can be almost arbitrary combined or used separately, provided that the correct data for the step could be inferred from the previous steps and/or the current state of the design process. Examples of such combinations in the context of the architectural design process are provided in Sect. 4.

3.1 Find

Find is historically the first phase that was developed for FLEA and MetisCBR in order to find similar architectural designs with methods of CBR. *Find*'s *distributed domain model* [6] (see Fig. 3) defines the underlying structure for all room configuration cases of the system. It is based on semantic fingerprints and AGraphML [20], the architectural implementation of the graph description language GraphML. Three main concepts are available to describe the case: *ROOM*, *EDGE* (room connection, relation), and *FLOORPLAN* (metadata). ROOM and EDGE can have multiple instances per case and describe the atomic parts of the floor plans with the corresponding AGraphML attributes and a number of specific additional attributes (such as `source` and `target` for EDGE). FLOORPLAN contains the meta information about the case and is enriched with specific attributes too (e.g., `roomTypes` for the list of room functionalities of the room graph). The main concepts are connected with the `is-part-of` connection. For retrieval, MetisCBR uses the so-called *Basic* strategy (see also Fig. 3 and [6]), which is based on the main premise of CBR: similar problems have similar solutions. *Find*'s *case base of floor plans* contains the cases for the case-based comparison with the query. Each case is constructed with respect to the domain model, incomplete cases (e.g., those without metadata) are not permitted.

Based on the results of the most notable experiments, e.g. [24], performed to evaluate MetisCBR's retrieval coordination component and the retrieval strategy in the context of other methods, the retrieval component of MetisCBR was extended with improvement of the search functionality, making it more flexible and responsive [5]. During a specific study, the representatives of the architecture domain were asked to play the role of the framework with the goal to examine their working and thinking processes during the early conceptualization process, with emphasis on the search for similar references during the room configuration process, in order to transfer a meta model of these processes to the framework.

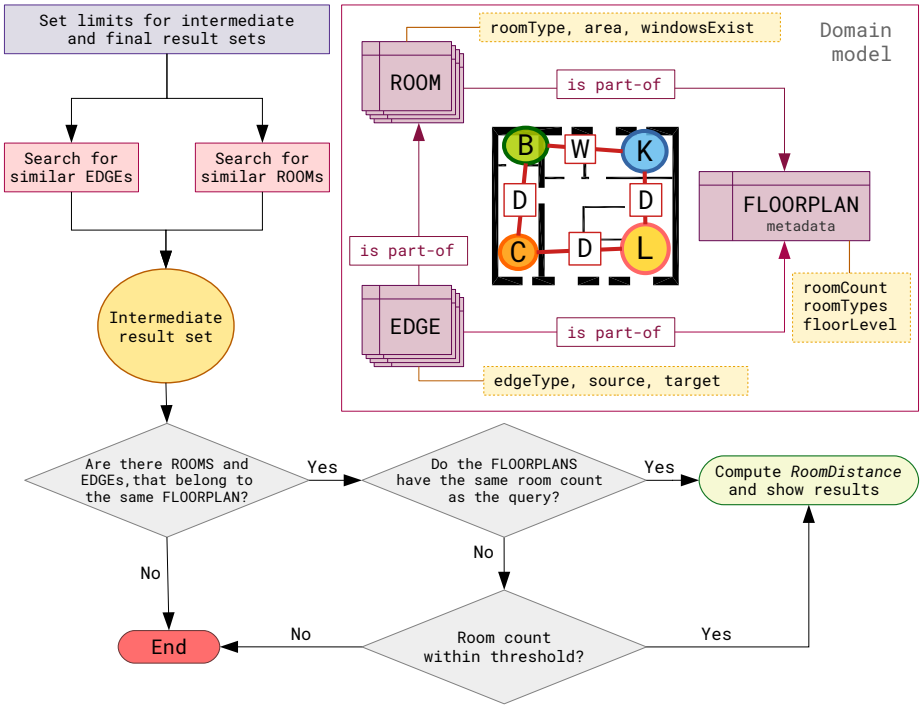

Fig. 3. The current state of the Basic retrieval strategy, together with the distributed domain model. The main concepts of the model, *ROOM*, *EDGE*, and *FLOORPLAN* are accompanied by a selection of their corresponding attributes (yellow rectangles). An example of a room configuration case is placed in the middle of the model. **L** stands for the *Living* room, **K** – *Kitchen*, **C** – *Corridor*, **B** – *Bath*, **D** – *Door*, **W** – *Wall*.

One of the results of this study is the generic definition for retrieval strategy that became a base for creation of new and modification of already available strategies. The most recent research work [28] presents a new strategy, the *Room Type Dominance* (RTD) strategy, as well as the extension of the Basic strategy according to the definition. The RTD strategy is based on the new criterion for floor plan comparison, the *room type dominance*: a measure of dominance of a room type within the room configuration (e.g., in a room configuration {`Living, Living, Living, Sleeping, Kitchen, Bath`} the room type `Living` is highly dominant). The RTD strategy uses it as the main comparison criterion along with the semantic fingerprint-based criteria and the abstraction levels [5] of the floor plans. The extension of the Basic strategy adds a special similarity value, *Neighborhood Similarity Coefficient* to the last step of the strategy. The evaluations of both new strategies have proven their suitability for use in MetisCBR under different scenarios: the RTD strategy for scenarios where specific functionalities of rooms or the floor plan should be detected, the modified Basic strategy for situations where a set of inspirational designs is required, however, with a very high structural similarity to the query [28].

3.2 Learn

The phase *Learn* of the implementation of FLEA represents a collection of machine learning methods to learn the context or purpose of the current room configuration process and to suggest a set of possible next actions for this process. In particular, the system module the *Suggester* (that implements and executes the *Learn* phase, see Fig. 4) analyzes the previous steps of the process recorded by the system, tries to determine which context the process most likely belongs to, and produces a number of possible continuations of the configuration based on the previous processes from this context. The key components of the Suggester module of MetisCBR are the process chain, the contextual clusters, the contextual recurrent neural networks, and the context footprint case bases [12].

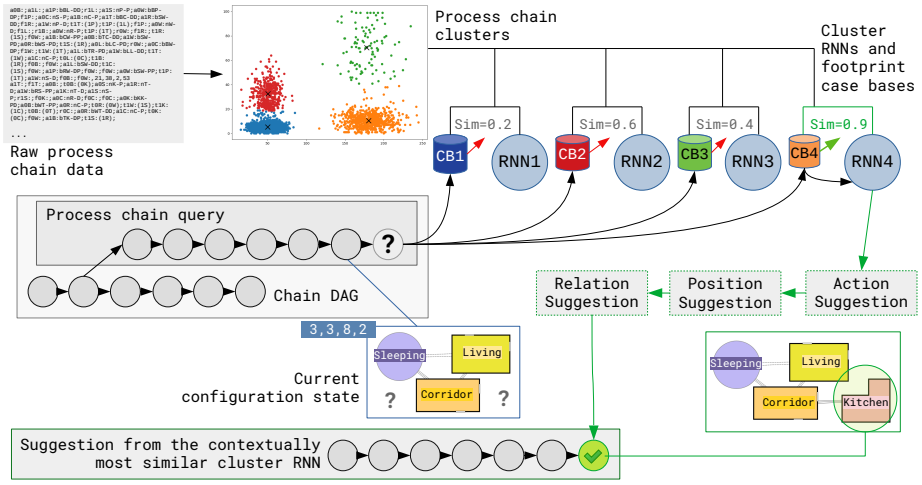

Fig. 4. The mode of operation of the Suggester in MetisCBR. An example of a visualized query with possible similarity values for each context and an example of a visualized suggestion are added for demonstration purposes. Figure adapted from [12].

The *process chain* represents an ordered record of the particular steps of the room configuration process. It is an emblematization of user's actions and is represented by a directed acyclic graph (DAG). For each step of the process, a notation of its action is mandatory. The Suggester can differentiate between room-related actions, such as *Addition*, *Removal*, *Reshaping*, or *Type modification*. Each action also contains specific information about the level of abstraction of the room, and the room type. For the Addition action, a set of additional information can be appended: a position within the configuration and the connections that should be used to connect the new room with rooms from the position suggestion. Each step in the process chain is accompanied by a metadata label (see Fig. 4) that consits of the current room and edge counts, the current amount of actions, and the amount of semantic fingerprints used in the latest search. The latter is the only attribute that connects *Find* and *Learn* in MetisCBR.

The *contextual clusters* represent the process chain dataset divided into a number of clusters based on the metadata described above. That is, each process chain is assigned to a cluster with contextually similar chains. The contexts are determined automatically with the *K-Means* clustering method that takes the metadata of the last step in the chain as the input vector. This contextual separation is a direct connection to our research [13] where the specific contextual classes define relations between the configurations in the retrieval result set.

After the cluster creation has been performed, a *contextual RNN* is created for each cluster as basis for training and the actual suggestion of the most suitable configuration step. In the past, RNNs have provided the best performance for sequential data (e.g., time series) as the mode of operation of their network cells allows for remembering the states of previous time steps in order to predict the next step. That is, RNNs are able to remember dependencies between the past steps over time. The Suggester uses *GRU* (Gated Recurrent Unit) [10] for the cells of cluster RNNs. To determine which cluster RNN is the most suitable one for suggestion of the next step for the current configuration, a special case base is created for each context cluster. This case base represents the cluster with a *footprint* – a selection of cases that provide the most typical metadata values for this cluster. This method is inspired by the footprint similarity-based retrieval [27]. The initial set of footprint cases is a product of random selection. Every new process chain query that is processed by the Suggester gets compared with all footprint cases of all cluster case bases, the RNN of the cluster case base that provides the highest similarity value is then queried and returns a set of possible next actions. The case that provided the highest similarity value remains in the case base, all others are removed and the case base gets filled up with new random cases from the cluster data. This repeats for every new query.

If the context RNN suggests to add a new room to the configuration, then the Suggester needs to find the proper position within the configuration and to determine which connections can be used to connect this new room to the adjacent rooms. The position suggestion uses a histogram of position entries of the corresponding action (for each of the possible addition actions such a histogram exists that was created from the initial process chain dataset) and suggests the positions consecutively starting with the position with the highest number of occurrence. The suggestion of connections is performed in a similar way: an instant histogram of all connections of each room type available in the current configuration is used. The Suggester then tries to find the best connection guess for the given number of the adjacent rooms, starting with the connections set with the highest number of occurrence.

For implementation of *Learn* in MetisCBR, an initial quantitative evaluation was conducted to examine the general suitability of the module [12]. A generated dataset with an amount of 1000000 process chains was used. 100 clusters/RNNs were created from this data with 50 chains per cluster footprint case base. 1000 generated queries were used to perform two suggestion phases. In the first phase 86% of suggestions were valid, i.e., accepted as suitable for the current configuration. In the second phase this value increased to $\approx 97\%$.

3.3 Explain

The current directions of the AI-related research domains emphasize the importance of human-understandable explanations of the internal processes of the AI systems. The research domain of Explainable AI (XAI) gained much attention within the last decade, resulting, for example, in multiple XAI-related workshops at the major AI conferences. The phase *Explain* of FLEA follows the current XAI development and requires an explanation feature for the systems that implement the methodology. To provide MetisCBR with explanation abilities, the *Explainer* module (see Fig. 2) was created as part of the system according to this methodology requirement. The main task of the Explainer is to support the retrieval process by providing explanations for the search results in form of textual expressions that contain information about system's decisions and reasons to include these results in the final result set. The mode of operation of the Explainer is based on the *Explanation Patterns* for CBR-based approaches [8]:

- *Justification* – Answers the question of why a particular result was returned.
- *Relevance* – Provides the context of the question that the system has asked.
- *Transparency* – Explains how exactly the system was able to find results.

Over time, a number of different Explainers have been conceptualized and implemented in MetisCBR. All of them have in common that they implement the explanation patterns and the validation of the produced explanations, i.e., they check the explanation for correlation between its expression or value and the requirements for understandable domain-specific explanations.

The *CBR-Explainer-1* [4] was the first one to implement the concept of explanation patterns and validation. Its task was to detect the patterns within the retrieval results by means of applying a specific ruleset that was conceptualized to cover all possible combinations of semantic fingerprints. After the pattern recognition phase, the corresponding explanation expression was generated, based on detected patterns (see Fig. 5), and evaluated with the case-based validation process based on comparison with ground-truth explanation cases from a case base of 'golden-standard'explanations that were checked for validity by a CAAD expert. During an evaluation with a case base of 225 room configurations, the CBR-Explainer-1 achieved 84.825% of validation processes with a positive outcome and so proved the suitability of its mode of operation.

The *CBR-Explainer-2* [13], an enhanced, however, more restrictive version of the CBR-Explainer-1, provides more deeply in detail going explanations that do not consider the semantic fingerprints only for patterns detection, but also take particular attributes of the domain model's main concepts into account. The detection process of each pattern was reworked, extensively enhanced and assigned a particular agent responsible for this pattern only. The Relevance detection is based on analysis of attributes of all ROOMs and EDGEs available in case and query, where each of them is checked for predefined requirements that determine if this instance is suitable for similarity assessment. If no Relevance was detected in either query or case, then the pattern detection process proceeds with recognition of the Justification pattern. Here, a Justification score that is based on

similarity values of the particular rooms, edges, and the result as a whole, serves as a classification means to categorize the result into a justification class. The detection of the Transparency pattern then depends on similarity assessment history of the attributes of the main concepts. The Transparency agent collects all available data on this history and constructs the assessment summary report. The explanation validation process of the CBR-Explainer-2 is an enhancement of the CBR-Explainer-1's validation, the most notabe modification is the inclusion of *undetected* patterns which provides a more exact detection picture. Besides pattern detection, the CBR-Explainer-2 is also able to detect contextual relations between the results by assigning different context classes to them. These classes represent specific features such as *RoomTypeDominance* (see Sect. 3.1) or *SparseConnections* [13]. The evaluation of the CBR-Explainer-2 with a dataset of 119 room configurations resulted in $\approx 80.4\%$ of validations with a positive outcome. The subsequent user study, during which the understandability of explanations was rated, resulted in satisfactory feedback with comments on visualization of explanations and improvement suggestions from the participants.

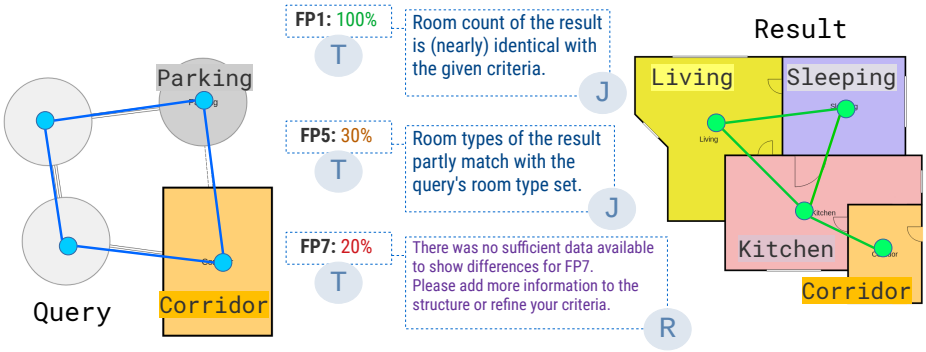

Fig. 5. An example of explanations produced with CBR-Explainer-1. T, J, and R stand for Transparency, Justification, and Relevance respectively. Figure adapted from [4].

The *DA-Explainer* [14] is based on the classification method *discriminant analysis*. This Explainer was conceptualized and implemented to predict the explanation and validation classes of the result. The explanation classes, denoted by the letters A-D and assigned by means of applying the discriminant function and decision trees, represent the detection grade of the corresponding pattern and so can describe how likely a sufficient amount of explanation-related information will be available in both query and result to produce an understandable explanation. The validation classes, denoted by V1-V4, estimate the likelihood of validity of the explanations. The DA-Explainer was compared in a joint evaluation [15] with the CBR-based Explainers. This experiment used a dataset of 120 room configurations and showed that the CBR-Explainer-1 produces the most constant validation values, whereas the DA-Explainer delivered the most inconsistent performance.

The last of the currently existing Explainers is the *BDI-Explainer* [22] that is based on the *Belief-Desire-Intention* architecture well-known from the MAS domain. This Explainer was influenced by the explainable BDI agent concept [18]. The BDI-Explainer provides a high-level structure for construction of other Explainers where each concept of the BDI architecture is connected to a specific task and/or the corresponding knowledge basis. Beliefs describe the domain-specific technical knowledge, i.e., the technical terms and relations of the CAAD domain. Desires represent the goals of the Explainer: detection of explanation patterns and validation of the generated explanation. Intentions stand for the most suitable next steps to achieve the desired goal by means of applying the beliefs knowledge, e.g., to contextually activate one of the available Explainers. The concept of the BDI-Explainer was examined by a group of 4 architects [22]. The results showed a general acceptance rate of 75% for the concept of knowledge- and pattern-based explanation generation. Additionally, it was also determined that the enrichment of results with explanations helps to avoid mistakes during the conceptualization process, but does not stimulate creativity.

3.4 Adapt

Modification of solutions of retrieved cases, emblemaized by the phase Reuse of the 4R CBR cycle, by adapting them (or just the parts of them) to the current situation, is an essential phase that influences the success of the reasoning process as it provides the user with a possible solution to her problem. For design cases, this means that a selection of generated variations of the current design at hand can be presented. FLEA's *Adapt* provides the implementation of the methodology with such functionality to give the designer another source of inspiration that shows how the current design can evolve over time.

The implementation of *Adapt* in MetisCBR is based on *Generative Adversarial Nets* (GAN), an approach for ANN-based generation of data objects. GAN makes use of two neural networks, where the Generator generates objects and the Discriminator decides if this object would appear real to a human. Our approach (see Fig. 6) takes the GAN methodology as a basis and extends it with a pre-generation step, the *adaptation complexity classification* phase implemented in the specific Classificator network that defines how strong the current design should be adapted. The outcome of this classification is the grade on the specific *complexity scale* **1**(slight)-**3**(heaviest) that is then used to select a proper mode of adaptation for the Generator that modifies the current room configuration according to the requirements of this grade. After that, all of the generated room configuration variants are forwarded to the Discriminator whose task is to determine if this configuration can be considered a descendant of the original configuration. The chosen variations are then saved in the case base and marked as descendants of the query design producing its particular case tree. All three steps of our adaptation approach use a convolutional neural network (CNN, ConvNet) as its corresponding underlying technique and work with a connection map as the floor plan representation (see Fig. 6).

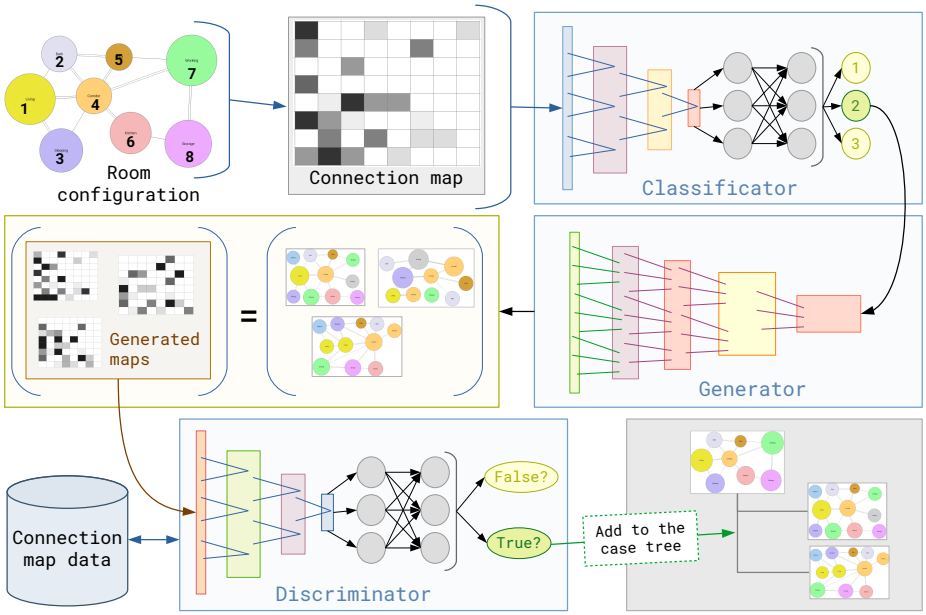

Fig. 6. The main structure of the GAN-based adaptation module of MetisCBR.

A connection map is a *modified adjacency matrix* of the floor plan's graph, however, instead of weights, a special *connection code* is used to mark the connection between two rooms. This code represents a numerical signature of the connection, where the first two numbers represent the connected rooms and the last number stands for the type of connection. For example, the connection code 241 stands for *Living and Kitchen connected by a Door*. Both directions, e.g., `Living`→`Kitchen` and `Kitchen`→`Living` are allowed, e.g., if they are connected with two different connection types. For use in ConvNets, these numbers are then converted into the grayscale values, e.g., 241 to 0.241. In Fig. 7, an example visualization of the original and the corresponding adapted connection maps is shown. The outcomes from the Generator's network are decoded afterwards, if they have been accepted by the Discriminator. The dataset of connection maps is a reflection of the case base for retrieval, with adaptation complexity classes derived from different steps of the process chains described in Sect. 3.3.

Currently the adaptation component is under active development, especially the configurations of different adaptations modes of the Generator are being tested. However, in an initial evaluation of the Classificator, classification accuracy of ≈93% could be achieved, based on 30000 generated connection maps.

4 Example Usages of FLEA Components

The following examples demonstrate how FLEA's components can be combined together or used separately in order to accomplish different tasks that can occur during the room configuration process (see Fig. 7).

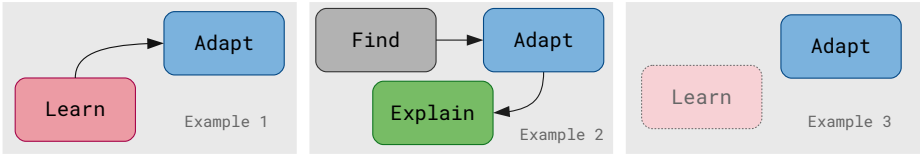

Fig. 7. Examples of FLEA component combinations.

4.1 Example 1: *Learn* and *Adapt*

If the architect does not wish to take a look on floor plans similar to the currently developed one, but instead only to know how this current design can or could evolve, then the system can combine *Learn* and *Adapt* components in order to present variations of the design by using the past design process steps history saved in the process chain (see Sect. 3.2). In this case, the *Adapt* component takes the different, in the most simple case manually or randomly selected, configuration states from the history (including the current state) and produces a number of adaptations that can show how the current state can evolve as well as what would happen if at some other point in the current room configuration history another configuration decision could have been made. This combination can help the designer reconsider her decisions and try to correct them if required.

4.2 Example 2: *Find* - *Adapt* - *Explain*

The most classic case, where the designer looks for the most similar cases in the dataset of past cases in order to find inspiration or take a look how the current configuration is embedded in a similar context can be extended with explanations of why these designs are useful for further design development process and also how they are related to each other (which is already implemented in MetisCBR). Additionally, this combination can be enhanced with an intermediate adaptation functionality that can adapt the current design state as well as a number of the most similar designs found in the case base. Furthermore, these selected most similar designs can themselves be used as the adaptation basis. Subsequently, the Explain component can be used to make retrieval as well as adaptation results more reasonable by showing the similarity-based relationship between them.

4.3 Example 3: *Learn* or *Explain* Separately

If computational ressources are to be considered then only the modules can be active that are currently required most, while other modules can be disabled. For example, *Explain* can be used separately to find out how two designs can be helpful for each other, which is the easiest possible use of *Explain*. In this case, the explanation component does not need much ressources, as pattern detection and validation will only be performed for the evaluation of one design. Other example is the usage of *Learn* as the only active component that concentrates the system performance on suggestions for the next step only. As *Learn* generally requires more computational ressources than other modules, it is recommended to periodically (re)consider how many clusters and RNNs should be created.

5 Future Work and Conclusion

In this paper, we presented FLEA, a methodology for support of the early conceptual design phase in architecture. FLEA is implemented in a framework for room configuration support, where all steps, namely *Find*, *Learn*, *Explain*, and *Adapt*, are implemented in particular system components for retrieval of similar floor plans, explanation of the returned results, suggestions of the next configuration steps, and generation of adapted variations of the current configuration. Future work will be conducted mostly for *Learn* and *Adapt* to improve these components and perform a number of quantitative and qualitative experiments.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications **7**(1), 39–59 (1994)
2. Ahmed, S., Weber, M., Liwicki, M., Langenhan, C., Dengel, A., Petzold, F.: Automatic analysis and sketch-based retrieval of architectural floor plans. Pattern Recognition Letters **35**, 91–100 (2014)
3. Anumba, C., Ren, Z., Ugwu, O.: Agents and multi-agent systems in construction. Routledge (2007)
4. Ayzenshtadt, V., Espinoza-Stapelfeld, C.A., Langenhahn, C., Althoff, K.D.: Multi-agent-based generation of explanations for retrieval results within a case-based support framework for architectural design. In: ICAART-18. Scitepress (2018)
5. Ayzenshtadt, V., Langenhan, C., Bukhari, S., Althoff, K.D., Petzold, F., Dengel, A.: Extending the flexibility of case-based design support tools: A use case in the architectural domain. In: ICCBR 2017, Trondheim, Norway, June 26-28, 2017, Proceedings. Springer (2017)
6. Ayzenshtadt, V., Langenhan, C., Bukhari, S.S., Althoff, K.D., Petzold, F., Dengel, A.: Distributed domain model for the case-based retrieval of architectural building designs. In: Petridis, M., Roth-Berghofer, T., Wiratunga, N. (eds.) UKCBR-2015, December 15-17, Cambridge, United Kingdom (2015)
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.) ICAART-2016, February 24-26, Rome, Italy. SCITEPRESS (2016)
8. Cassens, J., Kofod-Petersen, A.: Designing explanation aware systems: The quest for explanation patterns. In: ExaCt. pp. 20–27 (2007)
9. Cavieres, A., Bhatia, U., Joshi, P., Zhao, F., Ram, A.: CBArch: A Case-Based Reasoning Framework for Conceptual Design of Commercial Buildings. Artificial Intelligence and Sustainable Design – Papers from the AAAI 2011 Spring Symposium (SS-11-02) pp. 19–25 (2011)
10. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: EMNLP-2014. Association for Computational Linguistics (2014)
11. Chu, C.H., Wu, P.H., Hsu, Y.C.: Multi-agent collaborative 3d design with geometric model at different levels of detail. Robotics and Computer-Integrated Manufacturing **25**(2), 334–347 (2009)

12. Eisenstadt, V., Althoff, K.D.: 'what is the next step?' supporting architectural room configuration process with case-based reasoning and recurrent neural networks. In: FLAIRS 2019 (2019)
13. Eisenstadt, V., Espinoza-Stapelfeld, C., Mikyas, A., Althoff, K.D.: Explainable distributed case-based support systems: Patterns for enhancement and validation of design recommendations. In: ICCBR-2018. Springer (2018)
14. Espinoza-Stapelfeld, C.: Case-based classification of explanation expressions in search results of a retrieval system for support of conceptual phase in architecture (2018)
15. 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.) Agents and Artificial Intelligence. pp. 46–63. Springer International Publishing (2019)
16. Gerber, D.J., Pantazis, E., Marcolino, L.S.: Design agency. prototyping multi-agent systems in architecture. In: CAAD Futures 2015. pp. 213–235. Springer (2015)
17. González-Briones, A., Prieto, J., De La Prieta, F., Herrera-Viedma, E., Corchado, J.: Energy optimization using a case-based reasoning strategy. Sensors **18**(3) (2018)
18. Harbers, M., van den Bosch, K., Meyer, J.J.C.: Design and evaluation of explainable bdi agents. 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology **2**, 125–132 (2010)
19. Lai, I.C.: Dynamic idea maps: a framework for linking ideas with cases during brainstorming. International journal of architectural computing **3**(4) (2005)
20. Langenhan, C.: A federated information system for the support of topological bim-based approaches. Forum Bauinformatik Aachen (2015)
21. Langenhan, C., Petzold, F.: The fingerprint of architecture-sketch-based design methods for researching building layouts through the semantic fingerprinting of floor plans. International electronic scientific-educational journal: Architecture and Modern Information Technologies **4**, 13 (2010)
22. Mikyas, A.: Concept for development of an explanation component for bdi agents to support the design phase in architecture (2018)
23. Richter, K.: What a shame-why good ideas can't make it in architecture: A contemporary approach towards the case-based reasoning paradigm in architecture. In: FLAIRS Conference (2013)
24. 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: ICPRAM 2017, February 24-26, Porto, Portugal (2017)
25. Sharma, D., Gupta, N., Chattopadhyay, C., Mehta, S.: Daniel: A deep architecture for automatic analysis and retrieval of building floor plans. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 420–425. IEEE (2017)
26. Simeone, D., Cursi, S., Coraglia, U.M.: Modelling buildings and their use as systems of agents. eCAADe-2017 (2017)
27. Smyth, B., McKenna, E.: Footprint-based retrieval. In: Case-Based Reasoning Research and Development, pp. 343–357. Springer (1999)
28. Standke, S.: Strategical extension of similarity assessment of the retrieval module in a system for support of conceptual design phase in architecture (2018)
29. Voss, A.: Case design specialists in FABEL. Issues and Applications of Case-based Reasoning in Design pp. 301–335 (1997)
30. Zimring, C.M., Pearce, M., Goel, A.K., Kolodner, J.L., Sentosa, L.S., Billington, R.: Case-based decision support: A case study in architectural design (1992)