Explainable Distributed Case-Based Support Systems: Patterns for Enhancement and Validation of Design Recommendations

Viktor Eisenstadt[1(✉)], Christian Espinoza-Stapelfeld[1], Ada Mikyas[1], and Klaus-Dieter Althoff[1,2]

[1] Institute of Computer Science Intelligent Information Systems Lab (IIS), University of Hildesheim, Samelsonplatz 1, 31141 Hildesheim, Germany
{viktor.eisenstadt,christian.espinoza-stapelfeld,ada.mikyas}@uni-hildesheim.de

[2] German Research Center for Artificial Intelligence, Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
klaus-dieter.althoff@dfki.de

Abstract. This paper addresses the issues of explainability of case-based support systems, particularly structural CBR systems dominated by knowledge-rich comprehensive cases and domain models. We show how explanation patterns and contextually enriched explanations of retrieval results can provide human-understandable insights on the system behavior, justify the shown results, and recommend the best cases to be considered for further use. We applied and implemented our approach as an agent-based system module within a case-based assistance framework for support of the early conceptual phases in architectural design, taking a single floor plan as a case with a high number of attributes. For the retrieval phase, a semantic search pattern structure, Semantic Fingerprint, was applied, whereas the explanation generation phase is controlled by a number of explanation patterns adapted from already existing explanation goals. Rulesets, case bases, and natural language generation are used for construction and automatic revision of explanation expressions. A contextualization feature categorizes the results into different context classes and includes this information into the explanation. A user study we conducted after the implementation of the explanation algorithm resulted in good acceptance by the representatives of the architectural domain, a quantitative experiment revealed a high rate of valid generated explanations and a reasonable distribution of patterns and contexts.

Keywords: Case-based design · Knowledge-supported design
Explainable artificial intelligence · Pattern recognition
Contextualization · Human-computer interaction

1 Introduction

Users of modern AI-based decision support systems are usually not provided with a sufficient amount of information about the system's inner processes that lead to the presented result, solution, or recommendation. However, often, a particular requirement of users of such intelligent information systems is to have a possibility to understand and even reconstruct the system's behavior in order to follow its reasoning process. This is helpful for both, users and the system, if it is planned by developers that interaction with the system should be based on users' trust in the system. Some systems try to satisfy this requirement by including a feature to track the system's behavior, e.g., by providing the users and/or developers with comprehensive documentation or a special API endpoint that can return some explanation data with a certain grade of transparency. In many cases, however, an additional action from the user is required, which is not always desirable from the usability and user experience point of view.

In this paper, we present an automatic approach for explaining of design support system's actions, based on special explanation patterns that can be detected in the user query and the corresponding retrieval results. Our approach consists of three main steps – *pattern recognition*, *validation*, and *contextualization* (PRVC) – and is implemented in *MetisCBR* [6], a case-based framework for support of early design phases in architectutre, as the underlying technique for its results explanation module, the *Explainer*. This work is a continuation, further development, and enhancement of our pilot research into explanation of case-based design recommendations [3] (see also Sect. 4.1). In this paper, we describe a much more detailed approach that deals with deep properties of cases and builds a contextual relation between result sets from the same user session.

This paper is structured as follows: in Sect. 2, we describe our previous work in the domain of case-based support of the early conceptual phases in architecture and other related research in the domain of explainable AI. In Sect. 3, we shortly describe the MetisCBR framework, followed by the Sect. 4, where we provide a detailed description of our new explanation approach and its application within the framework. Finally, in the last two sections, we present the results of the user study and the quantitative experiment that evaluated the approach, and conclude the paper with a short review of this work and our future research.

2 Related Work

Foundations of explainability of CBR-based software were defined in a seminal work [24] that described current and future issues of this domain in relation to other CBR tasks, such as retrieval and retention. Some of these issues are highly related to our work presented in this paper and are described later in this section. However, before being summarized by Roth-Berghofer [24], earlier work on explanations in CBR has been published, e.g., Aamodt described explanation-driven case-based reasoning [1]. Explanations were also mentioned as an important feature of CBR systems in a survey [9] of functions of all four steps (Retrieve, Reuse, Revise, Retain) of the CBR cycle. In a series of work about explanation-aware systems [8,25,27,28], explanation patterns were presented that formalize the explainable system knowledge by means of applying a control structure with a collection of patterns that represent *explanation problem frame*. Problem frames themselves are a core concept presented by Jackson [13]. For recommender systems, explainability-themed research was conducted as well [19,29,30].

In the last decades, multiple research initiatives were started to support design process in creative engineering domains; issues of *case-based design* (CBD) were examined [18]. Especially in architecture, methods of CBD made a big progress with seminal projects such as PRECEDENTS [20], ARCHIE [32] SEED [11], or FABEL [31]. Later, a number of approaches continued research into this topic: DIM [14], VAT [17], or CaseBook [12]. Some of them had an explicit explanation facility implemented and established this fucntionality in CBR-CAAD approaches: ARCHIE contains explanations in the cases as 'outcomes' (goal satisfaction summary) and 'lessons to be learned' (contextual performance of the cases), CaseBook contains a *similarity explanation report* (no information is available about its concrete functionality or algorithms). One of the most recent research projects that worked on further research into the topic of (distributed) CBR-based design support, is *Metis* (funded by German Research Foundation). The focus of Metis were the graph-based and case-based retrieval methods. MetisCBR, the framework for which the explanation component presented in this paper was developed, was initially one of these methods. Others were the adapted VF2 graph matching method and the index-based retrieval in a graph database [26].

Explainability of AI systems has become an emerging topic during the last years, based on the wide distribution of such systems in a multitude of research/application domains. Initiatives, such as workshops [2,16,21–23] collect the current trends and newest approaches. In contrast to many other AI fields, case-based reasoning, as mentioned earlier in this section, explored and emphasized the importance of explanations even before the most well-known 4R-structure (Retrieve, Reuse, Revise, Retain) was presented. Roth-Berghofer's work on foundational issues of explainability in CBR [24] argued that the (commercial) CBR systems provide, if implemented, only simplistic types of explanations (why-, how-, and purpose-explanations). What these systems often do not provide are the so-called *cognitive explanations* that, inter alia, aim at answering the question of how the results are related to each other in different dimensions (*contexts*, in our case).

Our approach in this paper is an effort to combine all of these types of explanations to provide a versatile and universal algorithm for construction of reasonable explanations in structural CBR systems and to establish further the tradition of explanation facility in CBR-CAAD systems.

3 MetisCBR

MetisCBR[1] is a distributed system for case-based support of the early conceptual phases in archtecture. A case in MetisCBR is a floor plan that has attributes according to the Rooms+Edges+Metadata domain model [5] (see Fig. 2). The system's core functionality is the *case-based retrieval with semantic fingerprints*, where the fingerprints (FPs), based on a hierarchical description structure [15], represent a collection of attributes according to structural/relational floor plan abstractions, thus acting as semantic search patterns. For each query, the FPs selected by the user are distributed among the *retrieval containers* to decrease the complexity of search. The currently implemented FPs are shown in Fig. 1.

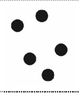
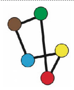
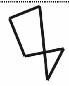
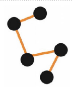
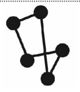
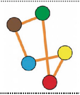
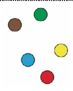
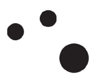
Fingerprint	Label / Specifics	Fingerprint	Label / Specifics
FP1 	**Room Count** No connections between rooms and no labels specified	FP5 	**Adjacency** Rooms information is complete, no edge labels
FP2 	**Relation Count** No room information specified	FP6 	**Accessibility** Edge information is complete, no room labels
FP3 	**Room Graph** Anonymous representation (no labels) of rooms & edges	FP7 	**Full Graph** All information about rooms and edges available
FP4 	**Room Types** No room connections, only room labels are specified	FP8 	**Natural Light** Light condition attributes

Fig. 1. Current semantic fingerprints of MetisCBR. FP1, FP2, FP4, FP8 are metadata-based (non-graph-based), FP3, FP5-7 are graph-based. Figure from [3].

4 PRVC Methodology for Explanation Generation

In this section, we present our methodology for creation of explanations for design recommendations. We think that this methodology might be of use not only for floor plan cases, but also for cases from other domains and other structural CBR systems. The only requirement to adapt this methodology for other domains and systems is a domain model that can hierarchically differentiate between cases, concepts, and attributes, or similar structures. Firstly, however, we give a definition of explanation as we use it in our support system.

Definition 1. *Explanation is a quadruple $E = (P, V, C, R)$, where P is the set of explanation patterns, V is the vocabulary for explanation expressions, C is the set of case contexts, and R is the set of mapping rules between P, V, and C.*

[1] http://veisen.de/metiscbr/.

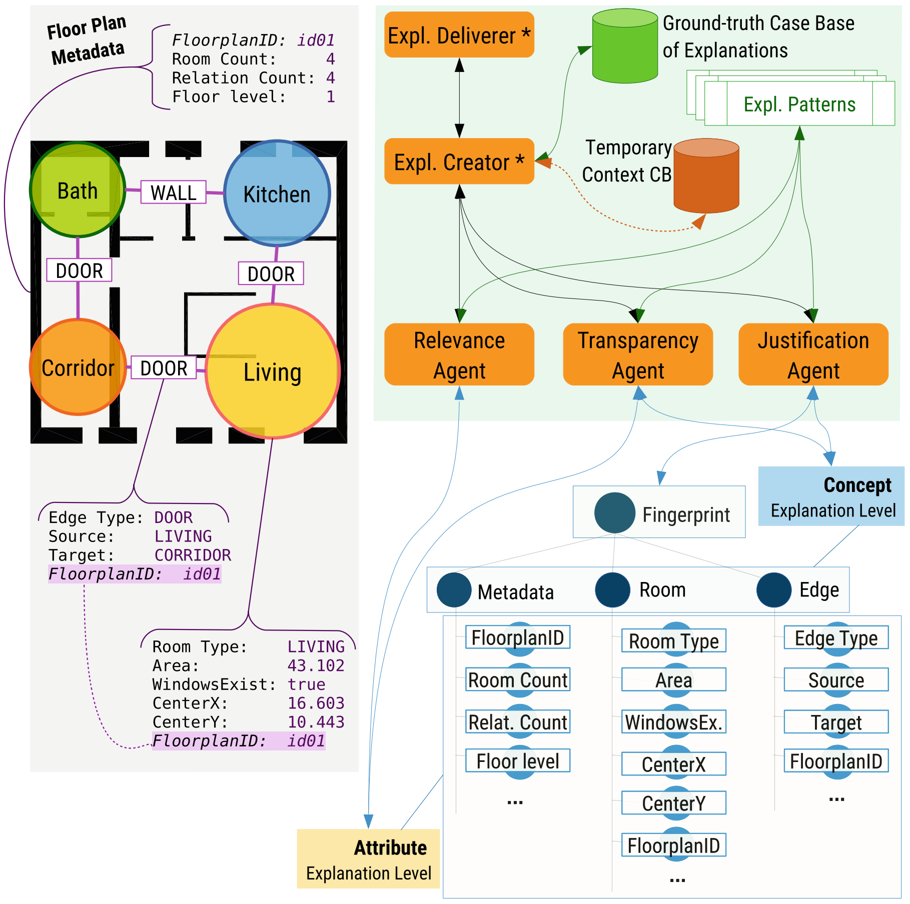


Fig. 2. Left: the domain model *Rooms+Edges+Metadata* and an exemplary case (floor plan) that consists of 4 rooms and 4 edges connected with a common floor plan ID. Bottom right: the general structure of an explanation tree with the explanation levels for a single retrieval result. Top right: the current Explainer module (asterisk marks the agents from the previous version of the module).

4.1 Previous Explainability Function in MetisCBR

The previous explainability function in MetisCBR is described in detail in our previous work [3]. In this section, only a brief overview of this previous/first version of the explanation module, the *Explainer*, is given to present its features relevant for the purposes of this paper. As shown in Fig. 2, two agents govern the process of generation of explanations. The first one is the *Explanation Deliverer* agent whose task is to receive the explanation request for a result and to forward it for further processing, and to receive the results enriched with explanations for forwarding them for displaying in the user interface. The second agent is the *Explanation Creator* agent who is responsible for actual creation/generation of explanations for the results forwarded by the Deliverer. The Creator tries to detect explanation patterns within the query-result object, that consists of the user query and the corresponding results, and generates an *explanation expression* based on the patterns detected (more on explanation patterns is provided in the next sections). After that, the explanation expression is validated against a set of ground-truth expressions, and, if valid, is added to the result.

4.2 Pattern Recognition Phase

The first phase of the PRVC methodology deals with detection of explanation patterns in the previously mentioned *query-result object*. This part of the explanation generation process works with a so-called *explanation tree*, which is created for each single result, and where each *explanation level* of the tree corresponds to an abstraction level of the domain model (see also Fig. 2):

1. *Fingerprint Level* – represents the highest possible abstraction level of the hierarchy and corresponds to the semantic fingerprint selected by the user.
2. *Concept Level* – this level stands for the core structural concepts of the domain model: Floor plan metadata, Room, and Edge.
3. *Attribute Level* – contains attributes of the core concepts according to the fingerprint (i.e., only the attributes of the FP are considered for explanation).

As shown in Fig. 2, the explanation levels are distributed among the *explanation agents* which in turn represent an explanation pattern. In the next sections, these patterns and the procedures for their detection and analysis are presented more in detail. First, however, we give a short description of the theory behind the explanation problem frames and patterns.

Explanation Problem Frames and Patterns. The *explanation patterns* [8] provide a means for abstracted description of explanation-aware computing problems when it comes to dealing with the question of how the reasoning process of an intelligent information system should be made understandable to the user. The concept of explanation patterns is a derivation from the original concept of generic software engineering problem frames [13], and thus an adaptation of these frames for the explanation-aware computing domain. Therefore, the explanation patterns can also be considered *explanation problem frames*. The general structure of an explanation pattern consists of a machine (representation of the software component that creates the explanation), domain (representation of the application area), and requirements (criteria of the proper solution space for a problem, in this case a space of possible explanations). Our rationale for use of explanation patterns for the explanation component was the similarity of their concept to the concept of semantic fingerprints, so that our knowledge and expertise in work with patterns could be transferred. As a result of our work on this transfer, a new *Fingerprint* machine was created [3] that connects the semantic fingerprints of architecture to the explanation patterns of *Relevance*, *Justification*, and *Transparency* [10].

Relevance Pattern. The *Relevance* pattern was conceptualized to justify the questions that system asks the user if certain requirements have not been met. For example, if not enough relevant information was provided to properly answer the query, the system may ask the user for more relevant information and display a message why it needs this information, e.g., why the answer/result may be inexact or incorrect in the current context (the *purpose*-question [24]).

To provide the new version of the Explainer with abilities to detect queries and cases that could not be considered for a proper similarity assessment because of their incompleteness or inexactness, we implemented a rule-based proving mechanism that checks each structural entity of the query and of the result, i.e., each room (node) and room connection (edge), for the *structural completeness* requirement. Currently, the structural completeness requirement for rooms is considered met if the room is *not isolated* – that is, each node has to be connected to *at least one* other node, and if the room label is in the list of enabled labels, such as `Working`, `Living`, `Sleeping`, or `Corridor`. For edges, this requirement employs a ruleset that checks their *source* and *target* rooms for the same label availability requirement – i.e., each edge has to have both source and target to be considered structurally correct. Depending on the outcome of this check, each entity in the result gets an additional *relevance label* that corresponds to its structural correctness (i.e., `RelevanceQuery`, `RelevanceResult`, or `RelevanceNone`). The relevance score *relScore* for rooms ($relScore_r$) or edges ($relScore_e$) is then:

$$relScore = \frac{|R_Q| + |R_R|}{|R_N| + |R_Q| + |R_R| + e} \tag{1}$$

where R_Q is the set of entities labeled with `RelevanceQuery`, the same for R_R (`RelevanceResult`), and R_N (`RelevanceNone`). e is the error rate for entities whose relevance could not be determined. The entire result floor plan then gets its own relevance label depending on condition resolving shown in Algorithm 1.

```
Data: $relScore_r$, $relScore_e$, Relevance threshold $t_{rel}$, Floor plan $f$, Floor
      plan relevance $rel_f$, Expression vocabulary $E$, Relevance
      expression set $E_{rel} \subseteq E$
if $relScore_r > t_{rel}$ or $relScore_e > t_{rel}$ then
    $rel_f$ = true
    if $relScore_r >= relScore_e$ then
        $e_{rel} = e_{rel}^{rooms} \in E_{rel}$
    else
        $e_{rel} = e_{rel}^{edges} \in E_{rel}$
    end
else
    $rel_f$ = false
end
```

Algorithm 1: Rule-based relevance determination for a result floor plan.

The outcome of this algorithm, if the relevance pattern has been detected for the entire result (i.e., $rel_f = \texttt{true}$), is an expression e_{rel} from the corresponding subset of expressions that should help the user understand why more data is needed by the system to ensure proper similarity assessment for this query or result. Some examples of such expressions are:

– *'This database floor plan may not have enough structural information about room connections for proper similarity assessment.'*
– *'Not enough information has been provided about room configuration to properly assess similarity for this query. Please provide more structural information for the room configuration.'*

Justification Pattern. Reasoning of why a result might be good/helfpul is the task of the *Justification* pattern (*why*-question [24]). For the proper implementation of this pattern in the new Explainer, we relied on our previously applied reasoning premise: a possibly helfpul result is a result whose overall similarity should be at least over the threshold of a sufficient similarity grade. Like in the first version of the Explainer [3] (and for results of the retrieval phase [5]), we applied the following similarity grades: *very similar* if result's overall similarity $Sim \geq 0.75$, *similar* if $0.75 > Sim \geq 0.5$, *sufficiently similar* if $0.5 > Sim \geq 0.25$, and *unsimilar* if $Sim < 0.25$. For the extension and more detailed recognition of possibly helpful designs we introduced an additional justification score $jstScore$:

$$jstScore = \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n} s_{e_i} + \frac{1}{m}\sum_{i=1}^{m} s_{r_i}\right) \tag{2}$$

where $s_e \in S_E$ are the similarity values of edge entities of a result, the same for rooms with $s_r \in S_R$. It is important to notice, however, that the $jstScore$ computation and all other justification operations are only executed if the *Relevance* pattern has not been detected (i.e., if $rel_f = \texttt{false}$, see Algorithm 1). After the computation of $jstScore$, the justification expression is added to the explanation text of the result, depending on conditions shown in Algorithm 2:

```
Data: $jstScore$, Threshold set $T_{jst}$, Floor plan $f$, Floor plan justification
      $jst_f$, Expression vocabulary $E$, Justification expression set
      $E_{jst} \subseteq E$, Similarity grades $G$, Justification classes
      $C_{jst} = \{0 : high, 1 : middle, 2 : low\}$
if $jstScore > t_{jst}^{high} \in T_{jst}$ then
    $jst_f = \texttt{true}$; $c_{jst} = 0$
else if $jstScore > t_{jst}^{middle} \in T_{jst}$ then
    $jst_f = \texttt{true}$; $c_{jst} = 1$
else
    $jst_f = \texttt{false}$; $c_{jst} = 2$
end
$e_{jst} = mapping(c_{jst}, g_f \in G, E_{jst})$
```
Algorithm 2: Justification expression determination for a result floor plan.

Where T_{jst} is a set of threshold values for classification of justification depending on its $jstScore$. Currently, following values are used: $t_{jst}^{high} = 0.6$, $t_{jst}^{middle} = 0.3$. The mapping function assigns the proper justification expression for the given justification class c_{jst} and the similarity grade $g_f \in G$ of the result floor plan.

Transparency Pattern. The last implemented explanation pattern is the *Transparency* pattern (the *how*-question [24]), whose task is to provide a means for decoding of the system's pathways to find a result. We think that the users of our system should be informed in as much detail as possible, but at the same time we are aware of the fact that they know how the system works in general. Therefore, we came to a conclusion that a summary of similarity assessment on the attribute level is the best way to provide the users with sufficient information about this procedure. To ensure this, a completely new approach has been implemented for the *Transparency* pattern that takes all the local similarity values from the entity pre-selection step of the retrieval strategy [5] into account. Generally, the transparency reasoning process consists of the following tasks:

1. *Collect* all information of the entity comparison history, i.e., how often and for which entity of the query the entity of the result has been object of comparison, and include the corresponding similarity values.
2. *Reason* about this data, that is, produce an understandable, human-readable summary of this data according to the user requirements and techical terms.

From the collected data, the transparency agent tries to derive the relevant similarity data for each of the attributes for the currently selected fingerprint and summarizes this data by grouping the attributes with the same overall similarity grade. The outcome of the reasoning process is a summarized statistical expression about the attributes' local similarity assessment (*local transparency*) followed by a detailed list of the mean similarity grades for each attribute for each entity. The same procedure is also conducted for the complete result set and is handled as the *global transparency*, and placed on top of the result set. Some examples of such outcomes are provided below (attribute overview omitted):

– '*This floor plan provides a sufficient grade of similarity in terms of passages, room functionalities, and light.*' (local transparency, FP 6 `Accessibility`)
– '*This result set has an overall high value of similarity for room area, and light condition. Low similarity has been determined for door connections.*' (global)

4.3 Validation

The basic principles of the validation process already introduced in [3] remained unchanged, especially because of its good performance: the complete explanation undergoes a feature extraction process, these features build together a case, i.e., become the attributes of this case which in turn becomes a query and is compared with the cases from the ground-truth (i.e., 'golden standard') explanation case base. The value of similarity with the most similar 'golden standard' case then becomes the *validation similarity* v_{max}. The explanation is considered *valid* if

v_{max} exceeds the specified threshold. For the new version of the Explainer, we modified the attributes of the explanation case and replaced the explanation text with the *fingerprint label* (see Fig. 1) and the overall FP similarity value. The detected explanation patterns remained the main similarity assessment feature for the validation. However, to provide a more exact comparison for the patterns as well, we now take into account the *undetected patterns* too, and do not use the detected ones exclusively. The following weighted sum is now in use for validation:

$$v = \omega_l l + \omega_o o + \sum_{i=1}^{n} \omega_{p_i} p_i \tag{3}$$

Where $v \in V$ ($v_{max} = max(V)$), and $\omega \in \Omega$ are the weights, where ω_l is the weight of the FP label similarity l, ω_o is the weight of the overall FP similarity value o, and ω_p are the weights for the similarity values of the patterns.

4.4 Contextualization

The last step of the PRVC methodology, and the completely new feature of the Explainer, is *Contextualization*, which is responsible for classification of the results into different contexts of the user session (see Fig. 3), and is intended to provide a means for cognitive explanations mentioned in Sect. 2. The inspiration for this feature came from the experience with different internet services such as Flickr or Netflix, where the automatic tagging of pictures and categorization of movies/series are the well-known features. To adapt this feature for our cases, we consider each user request to the system an action of a conceptualization process according to the definition of *Process* defined in our previous work on transfer of cognitive processes of architectural domain representatives into the system [4].

Definition 2. *Process is a triple $P = (S, t, A)$, where S is a set of retrieval strategies, t is the type of the process (e.g., sequential, semi-sequential, enclosing iteration), and A is the set of actions. $A = A_s \cup A_i \cup A_e$ (actions can be of starting, ending, and intermediate type), where $A_s \wedge A_e \neq \varnothing$. Strategies are linked to actions with a surjective mapping $S \twoheadrightarrow A$, i.e., $\forall a \in A \exists s \in S$.*

The basic mode of operation of the contextualization process is based on *feature extraction* from a single result: the main semantic and structural properties, such as room and edge count, room types, or a number of identical room types between query and result, are extracted from the result data and mapped with different contextual classes. These classes represent an abstract expression about the floor plan, some examples are:

- `SparseConnection` - represents floor plans where the number of connections is in the interval from lower to marginally higher as the number of rooms.
- `RoomCount, EdgeCount` - number of rooms and/or room connections is equal to the corresponding average value of the complete result set.

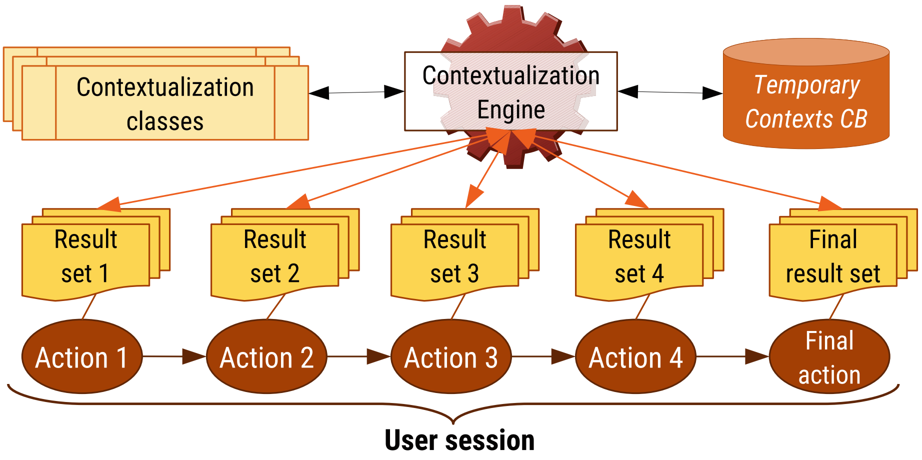


Fig. 3. Contextualization of the results of a sequential process.

– `RoomTypeDominance` - floor plans where a certain room type dominates the room configuration (e.g., {`Living, Living, Living, Kitchen, Sleeping`}).

Additionally to the classification step, for each user session a special temporary case base is created that contains cases where the attributes represent the extracted features named above, with corresponding values. For each unique floor plan result from a session, such a case with a count as label is created (if this case is already available then its count gets increased). For each new single result, a context similarity value $sim_{context}$ is then calculated as follows and categorized into a similarity grade (f is a feature, F is the set of all available features):

$$sim_{context} = \frac{1}{n}\sum_{i=1}^{n} sim_{f_i} \quad f \in F \tag{4}$$

The result of the contextualization process is a contextual expression that contains information about contexts available in the single result, providing the user with additional information about differences and commonalities regarding the configuration of all results. Exemplary contextual expressions are:

– '*This result has a high grade of contextual connection to the previous retrieval results of this session. Available contexts for this result are:* `Room Type Dominance`, `Sparse Connections`.'
– '*This result has a very low grade of contextual connection within this session. No contexts could be determined.*'

The $sim_{context}$ value is then combined with the overall similarity of the result, thus influencing its position in the overall result ranking within the result set.

Other influence is the previously mentioned *case label*, that works as a *boost value* for results whose final similarity values are identical.

4.5 Explanation Algorithm

Summarizing all of the steps described above we present the algorithm for generation of explanations within MetisCBR, that can be transferred or adapted for other systems. The concrete implementation depends on the domain and the corresponding domain model, however, the adaptation should not be difficult, as many structural CBR systems use the attribute-value-based structure for cases (Fig. 4).

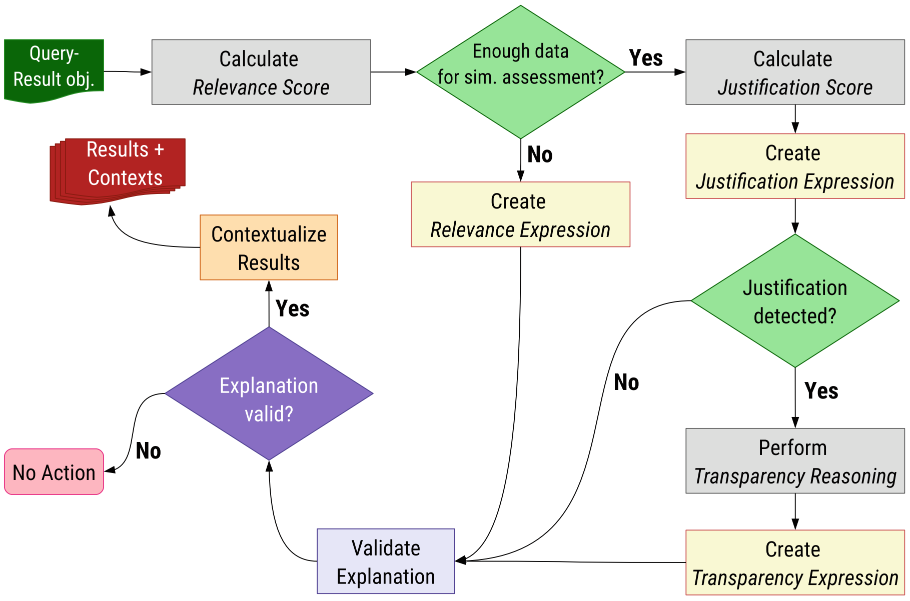

Query-Result obj.
Calculate *Relevance Score*
Enough data for sim. assessment?
Yes
Calculate *Justification Score*
No
Create *Justification Expression*
Results + Contexts
Create *Relevance Expression*
Contextualize Results
Justification detected?
Yes
No
Explanation valid?
Yes
No
No Action
Perform *Transparency Reasoning*
Validate Explanation
Create *Transparency Expression*

Fig. 4. Algorithm for generation, validation, and contextualization of explanations.

5 User Study and Quantitative Evaluation

To evaluate the new, PRVC-based, Explainer module, we conducted two studies: the first one was a quantitative experiment that aimed at examination of the new Explainer for validity of explanations and distribution of detected patterns. Additionally, we examined the context similarity and contexts distribution of the result floor plans. In the subsequent user study, we presented the representatives of the architectural domain with the explanation module and asked them to rate the produced explanation expressions and context classes.

5.1 Quantitative Experiment

The quantitative test was performed on a case base of 119 floor plans. We used 18 different queries and produced 47 requests (≈3 randomly selected FPs per query) that returned an overall number of 5189 results. 14 ground-truth explanations were used for validation, i.e., the explanation of each single result was validated 14 times. A threshold of 0.5 was applied to determine the validity. Overall, the constructed explanations were validated 72646 times for these results. 58408 ($80,4\%$) of them resulted in a valid outcome – an expectable slightly lower value than in the previous version [3] (84.825% for 225 cases), considering the more restricted handling of patterns (inclusion of undetected patterns) and addition of a new validation attribute. For the theoretical maximum number of validity determinations for this part of the experiment (78302), overall ≈74,6% of valid explanations were produced, i.e., ≈10 valid explanations per single result. Figure 5 shows the distribution of detected and absent patterns among the results.

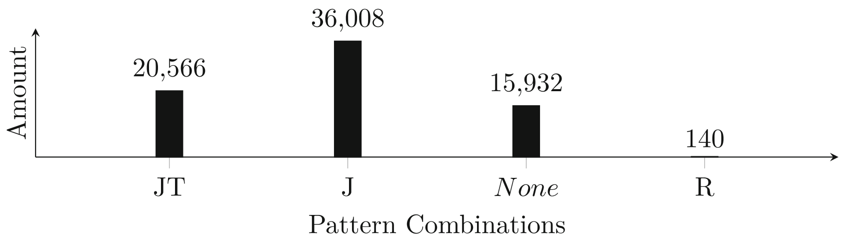


Fig. 5. Pattern combinations recognized in the results. J, T, and R stand for Justification, Transparency, and Relevance respectively. JT is the best case, R is the worst case (not enough data available), *None* (no R, but also no J or T) is detected when the similarity grade of the result is not sufficient. For non-graph-based FPs (see Fig. 1), transparency reasoning was not performed.

For the second part, the quantitative evaluation of the contextualization process, we first examined the overall context similarity $sim_{context}$ of the results, which revealed an average similarity value of ≈0.31, where from the total number of 5189, 2280 results had $sim_{context}$ of 0.35, 1934 results had $sim_{context}$ of 0.15, and 944 results had $sim_{context}$ of 0.55. Further, we examined the overall count of the detected classes during contextualization. Similar to the patterns examination, we show how the contexts were distributed among the results (Fig. 6).

5.2 User Study

For the subsequent user study, we used the same floor plan data set as for the quantitative experiment. The participants, $n = 5$, were asked to answer a specific questionnaire that contained questions regarding the understandability of explanations and their opinion on the contextualization feature. Before the

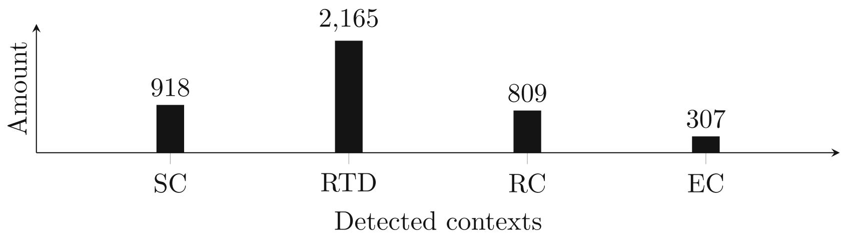


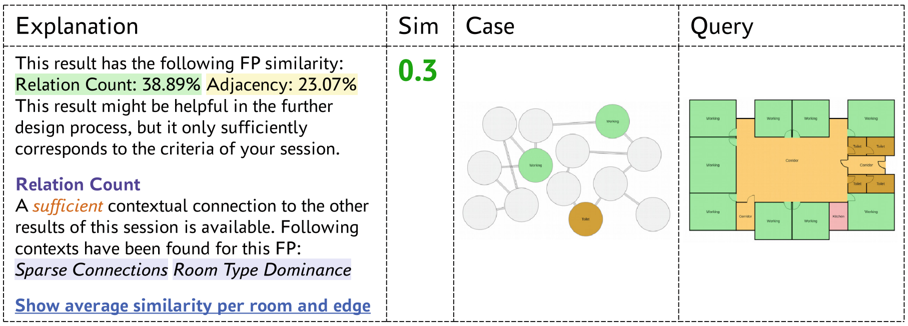
Fig. 6. Context classes recognized in the results. *SC*, *RTD*, *RC*, and *EC* stand for Sparse Connections, Room Type Dominance, Room Count, and Edge Count respectively.

rating process, each of the participants was asked to run a self-sketched or one of the already existing queries against the data set and get the corresponding results and explanations (see Fig. 7). 2 FPs should be used for either type of the query. On average, we spent ≈1.5 h per experiment session for each participant. Following questions (Q[n]) were included in the rating questionnaire:

1. Are the explanation expressions and their purpose understandable?
2. *(After the concept of explanation patterns was explained)*:
   Is it easy to recognize which partial expression belongs to which pattern?
3. Specific pattern questions:
   (a) `Justification:` are more reasons for recommendation required/advisable?
   (b) `Transparency:` Is more statistical data required, why and what exactly?
   (c) `Relevance:` is it understandable why the system needs more data?
4. Is the context-awareness and classification of results easy to recognize?
5. *(After the contextualization process was explained)*:
   Is it understandable how the classification/contextualization feature works and how would you estimate the helpfulness of the current contexts?

Generally, for Q1, experienced users who already knew how the system works and participated in one or more of our previous studies, found that the textual explanations are only partly helfpul and wished an additional visualization between query and result (which is already available for one of the MetisCBR-compatible user interfaces [7]). They also wished more detailed transparency explanation, e.g., for each attribute of each room and room connection. The inexperienced participants, i.e., architects who did not work with MetisCBR before (but at least knew the general concepts of CBR), found the textual explanations adequate and could recognize their purpose directly. The expressions themselves were also considered adequate and understandable.

For pattern questions (Q2-Q3), all of the participants were able to recognize which textual part is responsible for which explanation pattern. The positioning of the justification explanation before all others was considered a good decision, the length of the corresponding justification text was also considered good. Besides the already mentioned more detailed transparency data for experienced



Explanation	Sim	Case	Query
This result has the following FP similarity: Relation Count: 38.89% Adjacency: 23.07% This result might be helpful in the further design process, but it only sufficiently corresponds to the criteria of your session. **Relation Count** A *sufficient* contextual connection to the other results of this session is available. Following contexts have been found for this FP: *Sparse Connections* *Room Type Dominance* Show average similarity per room and edge	**0.3**		

Fig. 7. Query, case, and explanation of a result with a 'sufficiently similar' similarity grade as they appeared to the participants. Blank lines separate texts for Justification and Contextualization, the last line contains a link for opening the statistical expression (Transparency).

users, a filter function for an 'entity-for-entity' comparison was requested. The purpose of systems' questions in case of lack of sufficient data was also found adequate.

The last examined feature, the contextualization of results (Q4-Q5), was considered interesting and most enriching. All of the participants could recognize the context classes and their purpose for the explainability. The fact that the participants were already familiar with similar concepts from internet services and portals played a big role. However, some of the contexts, i.e., which features exactly they represent, were not always clear (e.g., *What does 'sparse connections' mean?*). Suggestions for new contexts also were made, e.g., `RoomDominance`, for floor plans where a certain room has a dominant area value.

6 Conclusion and Future Work

In this work we presented a new, extended, version of our results explanation approach for architectural design recommendations, which now works by means of applying the PRVC (pattern recognition, validation, contextualization) methodology. The complete methodology was presented in detail, including algorithms for partial steps and the overall explanation algorithm. Detection of explanation patterns, automatic validation of generated explanations, and the contextualization of results are the main aspects of the approach. We evaluated the new Explainer module with a quantitative expriment and a user study with participation of the architectural domain representatives.

Our future work will be concentrated on a better analysis of explanations (e.g., how the validation and the context classes change over time). Additionally, we will improve each of the steps, especially by applying an improved, more domain-oriented wording (e.g., Space Syntax) for explanations and context classes.

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. Aha, D., Darrell, T., Pazzani, M., Reid, D., Sammut, C., Stone, P.: IJCAI-17 Workshop on Explainable AI (XAI) (2017)
3. 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: Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018). Scitepress (2018)
4. 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: Aha, D.W., Lieber, J. (eds.) ICCBR 2017. LNCS (LNAI), vol. 10339, pp. 46–60. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61030-6_4
5. 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.) Proceedings of the 20th UK Workshop on Case-Based Reasoning (UKCBR-2015), 15–17 December, Cambridge, United Kingdom (2015)
6. 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), 24–26 February, Rome, Italy. SCITEPRESS (2016)
7. Bayer, J., et al.: Migrating the classical pen-and-paper based conceptual sketching of architecture plans towards computer tools - prototype design and evaluation. In: Lamiroy, B., Dueire Lins, R. (eds.) GREC 2015. LNCS, vol. 9657, pp. 47–59. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52159-6_4
8. Cassens, J., Kofod-Petersen, A.: Designing explanation aware systems: the quest for explanation patterns. In: ExaCt, pp. 20–27 (2007)
9. De Mantaras, R.L., et al.: Retrieval, reuse, revision and retention in case-based reasoning. Knowl. Eng. Rev. **20**(3), 215–240 (2005)
10. Espinoza, C.: Analysis of Identification of Explanation Patterns for an Explanation Module for Support of Design Phase in Architectural Domain. Project report. University of Hildesheim (2017)
11. Flemming, U.: Case-based design in the SEED system. Autom. Constr. **3**(2), 123–133 (1994)
12. Inanc, B.S.: Casebook. an information retrieval system for housing floor plans. In: The Proceedings of 5th Conference on Computer Aided Architectural Design Research (CAADRIA), pp. 389–398 (2000)
13. Jackson, M.: Problem analysis using small problem frames. S. Afr. Comput. J. **22**, 47–60 (1999)
14. Lai, I.C.: Dynamic idea maps: a framework for linking ideas with cases during brainstorming. Int. J. Arch. Comput. **3**(4), 429–447 (2005)
15. Langenhan, C., Petzold, F.: The fingerprint of architecture-sketch-based design methods for researching building layouts through the semantic fingerprinting of floor plans. Int. Electron. Sci. Educ. J. Arch. Mod. Inf. Technol. **4**, 13 (2010)

16. Lim, B., Smith, A., Stumpf, S.: ExSS 2018: workshop on explainable smart systems (2018)
17. Lin, C.J.: Visual architectural topology. In: Open Systems: Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia, pp. 3–12 (2013)
18. Maher, M., Balachandran, M., Zhang, D.: Case-Based Reasoning in Design. Lawrence Erlbaum Associates, Mahwah (1995)
19. Muhammad, K., Lawlor, A., Rafter, R., Smyth, B.: Great explanations: opinionated explanations for recommendations. In: Hüllermeier, E., Minor, M. (eds.) ICCBR 2015. LNCS (LNAI), vol. 9343, pp. 244–258. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24586-7_17
20. Oxman, R., Oxman, R.: Precedents: memory structure in design case libraries. CAAD Futures **93**, 273–287 (1993)
21. Pereira-Fariña, M., Reed, C.: Proceedings of the 1st Workshop on Explainable Computational Intelligence (XCI 2017) (2017)
22. Roth-Berghofer, T.: Explanation-aware computing exact 2012. In: Proceedings of the Seventh International ExaCt workshop (2012)
23. Roth-Berghofer, T., Tintarev, N., Leake, D.B.: Explanation-aware computing exact 2011. In: Proceedings of the 6th International ExaCt workshop (2011)
24. 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
25. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 451–464. Springer, Heidelberg (2005). https://doi.org/10.1007/11536406_35
26. 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), 24–26 February, Porto, Portugal (2017)
27. Sørmo, F., Cassens, J.: Explanation goals in case-based reasoning. In: Proceedings of the ECCBR 2004 Workshops, pp. 165–174. No. 142–04 (2004)
28. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in case-based reasoning-perspectives and goals. Artif. Intell. Rev. **24**(2), 109–143 (2005)
29. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: 2007 IEEE 23rd International Conference on Data Engineering Workshop, pp. 801–810. IEEE (2007)
30. Tintarev, N., Masthoff, J.: Designing and evaluating explanations for recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 479–510. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_15
31. Voss, A.: Case design specialists in FABEL. In: Issues and Applications of Case-Based Reasoning in Design, pp. 301–335 (1997)
32. 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)