



Comparative Quantitative Evaluation of Distributed Methods for Explanation Generation and Validation of Floor Plan Recommendations

Christian Espinoza-Stapelfeld¹, Viktor Eisenstadt^{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
{christian.espinoza-stapelfeld,viktor.eisenstadt}@uni-hildesheim.de,
klaus-dieter.althoff@dfki.de

² German Research Center for Artificial Intelligence, Trippstadter Strasse 122,
67663 Kaiserslautern, Germany

Abstract. In this work, we compare different explanation generation and validation methods for semantic search pattern-based retrieval results returned by a case-based framework for support of early conceptual design phases in architecture. Compared methods include two case- and rule-based explanation engines, the third one is the discriminant analysis-based method for explanation and validation prediction and estimation. All of the explanation methods use the same data set for retrieval and subsequent explainability operations for results. We describe the main structure of each method and evaluate their quantitative validation performance against each other. The goal of this work is to examine which method performs better under which circumstances, at which point in time, and how good the potential explanation and its validation can be predicted in general. To evaluate these issues, we compare not only the general performance, i.e., the average rate of valid explanations but also how the validation rate changes over time using a number of time steps for this comparison. We also show for which search pattern type which methods perform better.

Keywords: Case-based design
Distributed artificial intelligence · Explainable agents
Comparative evaluation · Discriminant analysis · Validation
Semantic search

1 Introduction

Explainability of artificial intelligence systems (also known as Explainable AI or XAI) is currently a much discussed topic in the area of AI research. Many approaches were started and new trends of this research topic are discussed,

for example, at the XAI Workshop. For distributed AI systems, that rely on case-based reasoning (CBR) as its main underlying reasoning means, a number of approaches, but even more general theoretical foundations, were presented. However, in the combined research area of computer-aided architectural design (CAAD), multi-agent systems (MAS), and CBR no such approach has been presented to date (except our approach [1]) and no comparative evaluation between the approaches was conducted. In this paper, we present such a comparative evaluation between three explanation approaches implemented in MetisCBR, a distributed case-based framework for support of early phases in architectural conceptual design. The framework prototype was developed during the activities of Metis¹, a joint basic research project for the research domains of CBR, MAS, and CAAD.

The initial core functionality of MetisCBR was the retrieval of possibly helpful building design recommendations that could provide inspiration for the user (architect) during his or her conceptualization process. However, the growing interest of the AI community in XAI, user experience requirements for modern recommendation engines, and the absence of a versatile working explanation generation approach for retrieval results among the CAAD support software, lead to the idea of conceptualization and implementation of an explanation module for MetisCBR, whose first version [1] was based on explanation patterns and a ruleset for their detection. The explanation engine of MetisCBR is aimed at answering the questions of how the framework was able to find the results presented, what is the purpose of presenting exactly this set of results (i.e., why they are recommended), and which semantic and relational differences and similarities between the query and each of the single results are crucial and lead to inclusion of this result in the final result set.

After the first version of the explanation module, also called *the Explainer*, two other versions were conceptualized and implemented: an advanced version of the Explainer (*Explainer-2*) and an explanation estimation approach based on discriminant analysis (*DA-Explainer*, currently still in the early stage of development). In this paper, we compare all three approaches with each other in terms of their functionality for estimation of an explanation to be correctly produced, i.e., to contain a valid explanation expression.

This paper is structured as follows: related work for XAI in CBR and MAS will be presented in Sect. 2. In the next section, we give a short description of the MetisCBR framework: its main functionalities, including semantic search patterns, will be briefly described. In Sect. 4, we in detail present the explanation approaches implemented in MetisCBR. The comparative evaluation of these approaches will be described in detail in Sect. 5. The last section concludes this work and provides an outlook of our future research.

¹ http://ksd.ai.ar.tum.de/?page_id=240&lang=en.

2 Related Work

In this section we present work related to the purposes of this paper, i.e., work released in the research domains of (explainable) CBR and MAS, and the CBR-based approaches for support of architectural design phases.

2.1 Explainable CBR and MAS

CBR has a long and rich history in conceptualization and implementation of explainability features in the corresponding case-based systems. Early work on explanations for CBR approaches [2] was one of the precursors for the development of theoretical foundations for this area. Later, Roth-Berghofer [3] presented general questions of CBR-based explainability and examined a number of future research directions. The theoretical foundations of explanation problem frames for intelligent systems (see Sect. 4.2) were discussed by Cassens and Kofod-Petersen [4]. On the practical side, an explanation-aware system module for the CBR software myCBR was presented [3].

For the MAS research area, the most notable explainability approach is an explainable BDI (belief, desire, intention) agent [5, 6]. These research contributions describe an explanation module inside a BDI architecture-based agent that contains a so-called *behavior log* that is parsed by an explanation algorithm for finding beliefs and goals for the current explanation of actions.

2.2 CBR-Based Architectural Design Support

CBR was one of the first AI areas to support the conceptual design phases by means of applying case-based decision support approaches, such as FABEL [7], PRECEDENTS [8], SEED [9], DIM [10], VAT (Visual Architectural Topology, a semantic representation method) [11], or CaseBook [12]. The latter approach CaseBook is the only one known to contain an explicit explainability feature, the *similarity explanation report*, but information available in [12] does not provide a sufficient amount of insight into this feature.

A comprehensive review of these and other CBR-based architectural design support approaches is available in [13]. Another seminal work [14] contains a detailed review of CBR's current state, influence, and history in CAAD. Current issues of CBR in CAAD are published in a short review [15].

3 MetisCBR

The MetisCBR framework prototype for support of early design phases of the architectural conceptualization process is based on a distributed structure where the agents of the system perform a case-based search for similar architectural designs in a database (case base) of previous designs. After the search the system automatically applies an explanation process for each single result in the result

set and enriches the result, if possible, with explanations. The available explanation methods, that are the evaluated in this paper, are described in Sect. 4.

The actual search for similar architectural designs is performed by means of applying a number of semantic search patterns (*semantic fingerprints*) – (graph-based) abstractions of established architectural room configuration concepts, such as adjacency of rooms, or availability of natural light for the rooms. The list of fingerprints (FPs) currently implemented in MetisCBR is shown in Fig. 1.

Fingerprints can be divided into graph-based (FP3, FP5, FP6, FP7) and metadata-based, i.e., use an abstract summarizing attribute, such as count of available rooms, for comparison (FP1, FP2, FP4). Multiple fingerprints can be applied for each query/request to the system, result sets of each particular fingerprint search are then combined/amalgamated and presented to the user. For graph-based FPs, a pre-selection step is applied during retrieval, that governs the exclusion of the non-similar atomic parts of a floor plan (such as rooms and room connections) from the search process.

MetisCBR has been object of different comparative evaluations of retrieval methods for search of architectural designs. Examples of such evaluations are the performance comparison and qualitative evaluation with graph-based methods of the Metis project [16], and the comparison with the rule-based retrieval coordination software KSD Coordinator [17].

4 Explanation Generation Methods

In this section, we present the explanation generation and validation methods of MetisCBR that were used in the comparative evaluation presented in Sect. 5. Each of the methods will be presented including the description of its general structure and functionality, how the explanation patterns are applied, and how the validation of generated explanations is performed. Before the actual description of the methods, we give a short review of general requirements for explanation methods to be used for MetisCBR.

4.1 General Explainability Requirements for MetisCBR

Generally, MetisCBR can use every compatible explanation method that can interpret the agent messages constructed with the FIPA-SL language. The main requirement for explanation methods to be used in MetisCBR, however, is that it should be able not only to produce/generate explanations but also validate them. The validation step is an essential one as it ensures the general quality of explanations and can exclude explanations that make no sense to the user. Which validation method is used is a decision of the method’s developers, however, it is advisable to make the validation process transparent to be able to compare it to other methods.

For explanations themselves, *explanation patterns* (see Sect. 4.2) should be used. Alternatively the explanations should be able to answer the why-, how-, purpose-questions as described by Roth-Berghofer [3]. This ensures the common

explanation structure for all explainable recommendations and provides the user with a familiar structure of expressions.









Fingerprint	Name / Specifics	Fingerprint	Name / 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. Semantic search patterns (FPs) currently implemented in MetisCBR. Figure from [1].

4.2 Explanation Problem Frames and Patterns

A framework for *explanation patterns* for intelligent information systems and applications was conceptualized by Cassens and Kofod-Petersen [4] to provide such systems with a possibility to make the behaviour of such systems more transparent and traceable for their users. Initially conceptualized for case-based reasoning applications, the patterns can be used for almost every type of an intelligent AI system that follows their structure, i.e., uses the patterns with their initially conceptualized structure and purpose. Explanation patterns themselves are based on *Problem Frames* conceptualized by Jackson [18]. Thus, the patterns enhance the problem frames for use for explainability problems. A number of different patterns was conceptualized that provide different explanation functions. The most important of them, and implemented in all our examined explanation methods are *Justification*, *Transparency*, and *Relevance*. The adaptation of explanation patterns for MetisCBR is shown in Fig. 3.

Relevance Pattern. The Relevance pattern is aimed at explaining why the question that the systems asks the user is relevant in the current context. For the purposes of our design support framework, this means that the system may ask the user (an architect) for more relevant data for proper comparison of case and query if the structural and relational connections provided in the query

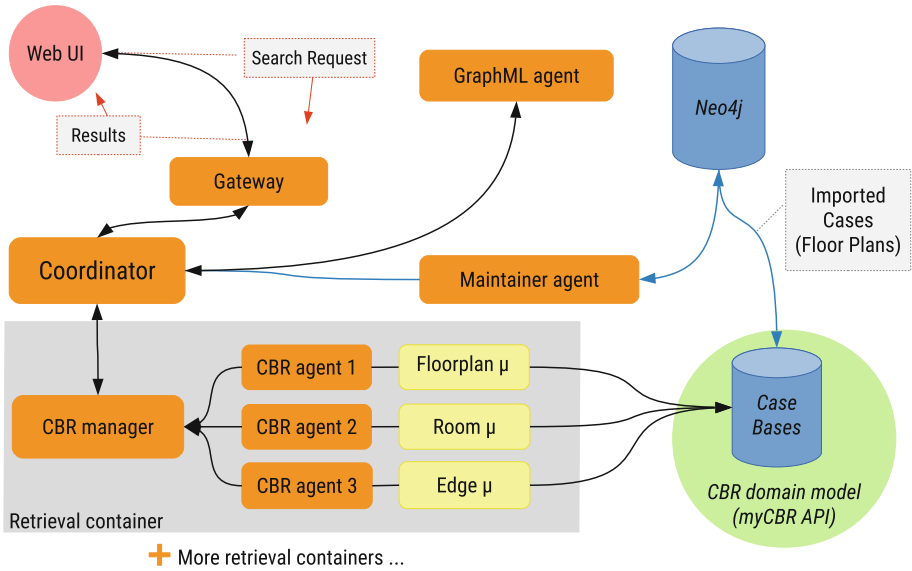


Fig. 2. Overview over the retrieval component of MetisCBR. Figure from [16].

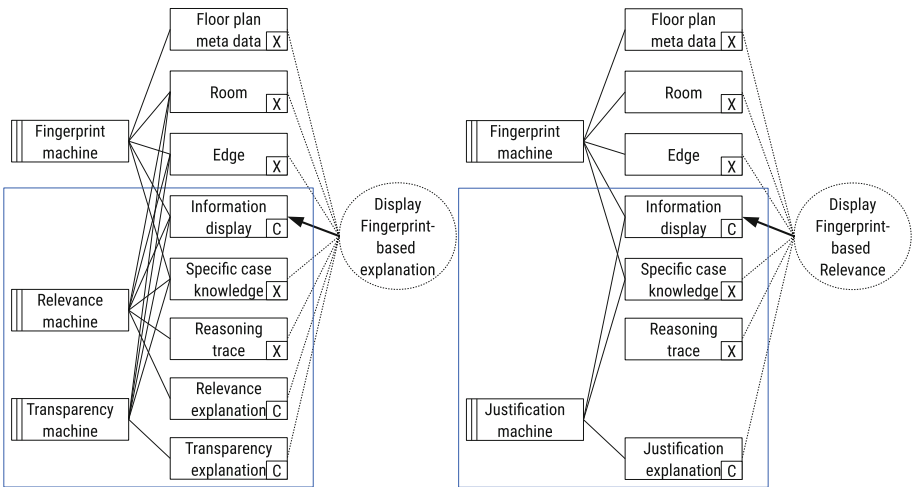


Fig. 3. Explanation patterns in MetisCBR. Highlighted in blue rectangles are the original patterns [4]. C denotes the goal of explanation, X is the system knowledge. Figure adapted from [1]. (Colour figure online)

did not contain sufficient amount of information. Usually, this is the case when some of the requirements, that should be met for proper similarity assessment, were not fulfilled. For example, a room in the room configuration of the design should be connected to at least one other room, otherwise the structure of the floor plan is not considered *well-formed*. The requirement for room connections is similar, they should not have an undefined start point or destination, i.e., each edge should be connected to a room on both ends (Fig. 2).

Justification Pattern. The Justification pattern was conceptualized to justify the current results, i.e., to answer the user’s question ‘Why do I see this result?’. The main aim of this pattern is to build trust between the user and the system, i.e., to provide the user with more confidence in the system behavior. Generally, this means that the system should ‘speak’ the user’s language, i.e., technical terms and expressions that the user is familiar with. This is also important from the human-computer interaction (HCI) point of view as explainability of intelligent (AI) systems is a question of HCI too. The proper justification improves the usability of the system, it also helps both sides to learn from each other in a better and more trustful way. Thus, the justifications should follow the system’s language conventions with a proper wording for technical terms of the explanation expression.

Transparency Pattern. The Transparency pattern’s goal is to provide the user with information of *how* the result was reached by the system. That is, it should be made transparent, for example, how exactly the final similarity value of the presented result has been calculated, i.e., which attributes were considered for comparison, how the preselection of cases was executed, or how the ranking works in case of identical similarity values. For our system, mostly the first case is important, as the semantic search patterns we use (semantic fingerprints) rely on attribute-value-structure. To achieve a good grade of transparency, two general possibilities exist:

- *Sequential* transparency – each similarity assessment step, i.e., outcome of each attribute comparison can be included in the explanation expression. This way is more suitable for users who had much experience with the system and quickly can differentiate between the concepts and attributes of the result.
- *Cumulative* transparency – a summarized statistical expression about the assessment data. For example, the average similarity value for an attribute or concept, or a trend overview, e.g., how the similarity changes over time with addition and/or deletion of attributes considered.

Beside this, two general possibilities of assigning the transparency pattern expression to the retrieval results are available:

- *Global* transparency that is assigned to the complete result set and can be placed over all of the single results to provide a general transparency expression about the results. The above mentioned cumulative transparency is usually used for this global expression.

- *Local* transparency that is provided for a single result to enrich it with insights for its own similarity assessment only. Cumulative as well as sequential transparency can be used for this type of transparency assignment.

4.3 CBR-Explainer-1

The first version of the explanation module for MetisCBR, the Explainer, was created to initially implement the explanation patterns for retrieval of architectural designs. This version implemented pattern detection based on a common ruleset for all patterns, however, it did not use the particular attributes of rooms and edges and relied on floor plan metadata only. Following exemplary rules can be applied (rules from our paper on the first version of the Explainer [1]):

- FP 1, 2, 4, 8: The pattern ‘Transparency’ is detected if 2/3 properties from {Room Count, Edge Count, Room Types} could be detected in the meta data of the query floor plan.
- FP 1, 2, 4, 8: The pattern ‘Justification’ is detected if the similarity grade of the result floor plan is better than *unsimilar*.
- FP 6: The pattern ‘Transparency’ is detected if *all* properties from {Room Count, Edge Count, Edge Types} could be detected in the meta data of the query floor plan.
- FP 7: The pattern ‘Justification’ is detected if the similarity grade of the result floor plan is better than *unsimilar* and all properties from {Room Count, Edge Count, Room Types, Edge Types} could be detected in the meta data of the query floor plan.

The first version of the Explainer (CBR-Explainer-1) employed two agents responsible for creation and validation of explanation expressions: the *Explanation Deliverer* agent, who is responsible for receiving of the query and result to be explained and sending the results enriched with explanations back for displaying in the user interface; and the *Explanation Creator* agent responsible for generation and validation of actual explanation expressions. In Fig. 4, the general structure of the CBR-Explainer-1 is shown.

The validation process in the CBR-Explainer-1 is implemented as a case-based validation process. That is, each produced explanation is handled as a case and gets validated against the case base of ground-truth explanations provided by an expert in the architectural domain/CAAD. The maximum similarity value from the comparison with each of the ground-truth explanations becomes then the *validation similarity* v_{max} . If v_{max} exceeds a specified threshold, then the produced explanation is considered valid. The similarity measure for validation determination is a dynamically adapted weighted sum, i.e., the weights get adapted with increasing/decreasing of the number of detected patterns. Attributes used for the dynamically weighted sum are shown in Table 1.

After the validation process, the *Explanation Creator* adds the explanation, if valid, to the result object and sends it to the *Explanation Deliverer*, which in turn sends it to the *Result Collector* agent that is responsible for collection of results for all FPs of the current query.

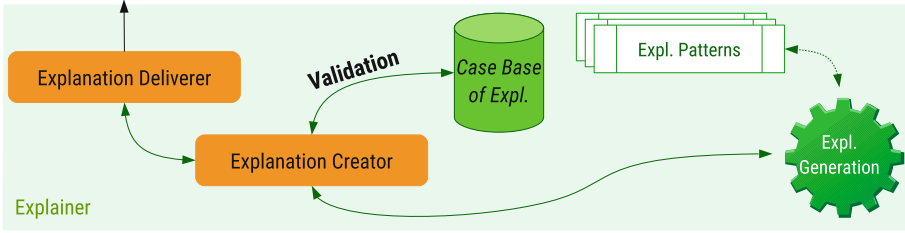


Fig. 4. General structure of the first version of the MetisCBR explanation module (Figure adapted from [1]).

Table 1. Attributes for validation in the CBR-Explainer-1 (Table from [1]).

Attribute	Type	Description	Similarity Func.
id	string	Internal Explanation ID	<i>not in use</i>
Case	string	Reference to the case (result)	<i>not in use</i>
Query	string	Reference to the query	<i>not in use</i>
Text	string	Text of the explanation	Levenshtein dist. sim
PatternJustification	boolean	Justification pattern available?	boolean comparison
PatternRelevance	boolean	Relevance pattern available?	boolean comparison
PatternTransparency	boolean	Transparency pattern available?	boolean comparison

4.4 CBR-Explainer-2

The second version of the Explainer, the CBR-Explainer-2, is an advanced version of the CBR-Explainer-1 and is intended to provide a more detailed, and thus restricted, explanation approach which also takes the particular attribute values of the room and room connection concepts into account. Structurally, the tasks of the agents of the Explainer remained the same, however, for each explanation pattern a special *pattern agent* was created that works with its assigned pattern only and communicates with the Creator. In Fig. 5, the general structure of CBR-Explainer-2 is shown.

The detection of patterns is different for almost all patterns. For the Relevance pattern, the CBR-Explainer-2 analyzes all rooms and room connections of the query and the currently compared case for availability of specific requirements. If the required features are not available, e.g., a room does not have connections or its label is unknown to the system, or an edge does not have a source or target, then it is not considered for comparison. If a certain percent (determined by a special *Relevance score*) of rooms and edges does not provide a proper feature set, then the complete query is not considered for comparison and gets the *Relevance label* of **true**, otherwise this label is **false**.

If the Relevance label is **true**, the Justification and Transparency detection does not take place. Otherwise the pattern recognition continues with Justification, where the justification expression, like in the CBR-Explainer-1 depends on the *similarity grade* of the result. After Justification, Transparency is detected

by means of applying a two-step reasoning process, where in the first step all available similarity data is collected for each room and room connection of the result. This data contains all historical comparison information in the context of the current query, including how often the room or edge has been used for each FP. The collected data is then cumulated (see also Sect. 4.4) and added as local transparency to each of the single results, for the complete result set this data is also cumulated and added as global transparency expression.

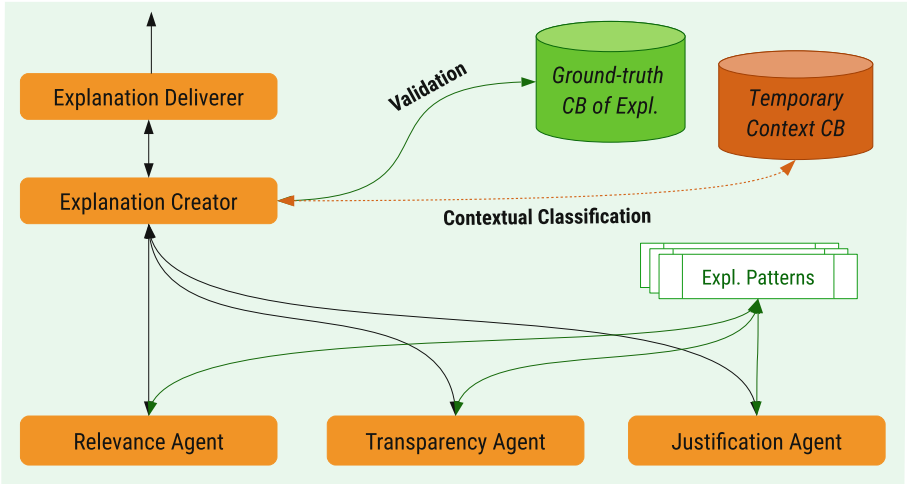


Fig. 5. General structure of the second version of the MetisCBR explanation module, the CBR-Explainer-2.

The validation process in the CBR-Explainer-2 is similar to that of the first version, however, the expression text similarity with Levenshtein distance has been replaced in the second version with the name of the semantic fingerprint and the overall similarity value of the result to provide a more exact comparison. The dynamic adaptation has also been replaced by a static weight distribution, as in the CBR-Explainer-2 the unrecognized explanation patterns are part of the similarity assessment as well. The mode of operation for determination of the validation similarity v_{max} remained identical in the CBR-Explainer-2.

The special feature of the CBR-Explainer-2, which is however not part of the comparative evaluation, is the contextual classification of the results with valid explanations. That is, each result of each FP is parsed by a *feature extraction* engine and is decomposed in its main features such as room count, edge count, or room types set. All of these features are then analyzed for their potential to include the result in a specific context class. A context class is a category of results with special properties, such as *RoomTypeDominance* for *floor plans with a room type that dominates over all other room types*, i.e., takes more than 50% of the entire room type set. The main purpose of the contextual classification is automatic tagging of the result floor plans.

4.5 DA-Explainer

The third possibility of creation of explanations for floor plan retrieval results is based on *discriminant analysis* (DA), a well-known versatile classification method of machine learning. The adaptation of discriminant analysis for the purposes of the Explainer module of MetisCBR has been explored by Espinoza-Stapelfeld [19] where a detailed description of DA for each semantic fingerprint and explanation patterns is available. In this paper, like for the previous two explainers, we give only a description of the relevant features.

Generally, the explanation generation and validation are based on prediction of their corresponding explanation and validation classes. We also differentiate between graph-based and non-graph-based (metadata-based) FPs. For the graph-based FPs, following discriminant function is used for the prediction of the explanation class:

$$c_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} \langle x - \mu_k, \Sigma_k^{-1} (x - \mu_k) \rangle + \log \pi_k \quad (1)$$

where k represents the similarity grade (whose calculation is identical to the categorization process for non-graph-based FPs, see below). Other values that are being used for preparation of the discriminant parameters are: n that represents the total number of the attributes for the given FP, and m_k that represents the total number of elements with the given similarity grade. Based on these values (k , n , and m_k), x represents single attribute values for each result with the given k ; μ_k represents *the sums* of values of the particular attributes of results with the given k divided by m_k ; Σ_k represents the products of $(x - \mu_k)$ and $(x - \mu_k)^T$ with factor m_k , and $\pi_k = m_k/n$.

For non-graph-based fingerprints, a decision tree is used that categorizes the result into one of the explanation classes: A if result's overall similarity $Sim \geq 0.75$, B if $0.75 > Sim \geq 0.5$, C if $0.5 > Sim \geq 0.25$, and D if $Sim < 0.25$. After the predicted class is determined, the assignment of the explanation patterns, that are in turn assigned to the classes, takes place. Following explanation classes are available for the patterns:

1. A – High Justification, High Transparency, i.e., a sufficient amount of information is available in query and result to perform a similarity-based comparison between them.
2. B – Middle Justification, Middle Transparency.
3. C – Low Justification, Low Transparency, i.e., information in query and result is sufficient, but local similarity values are very low and do not allow for recommendation of this floor plan.
4. D – Relevance only, i.e., more information is required for a proper comparison between query and result.

After the assignment of explanation patterns, a validation estimation process takes place. The DA-Explainer does not use a case base for validation, instead, its validation process is intended to sort candidates for validation in the both above described explainers. Like in the assignment of explanation classes, we

differentiate between graph-based and non-graph-based FPs. However, for the non-graph-based FPs, no explicit validation process is conducted, instead, the class determined in the explanation classification process is mapped to its corresponding validation estimation class v (A to V1, B to V2, C to V3, D to V4). For graph-based FPs, following formula is used first to estimate the validation candidacy for the complete result set:

$$v_k(x) = -\frac{1}{2} \log |\Sigma_k| - \log |(x - \mu_k)| + \log \pi_k \quad (2)$$

The validation estimation class can then be one of the following:

1. V1 – Highest category, the results with this class are most likely will produce a valid explanation.
2. V2 – Middle category, the results with this class have a good probability to produce a valid explanation.
3. V3 – Weak candidate, validation of its explanation can produce an insufficient value (below threshold).
4. V4 – Not recommended for validation of explanation.

After that $v_k(x)$ is multiplied with the overall similarity value of each of the floor plans, so that an individual validation estimation value for each single result can be calculated.

5 Comparative Evaluation

To quantitatively compare the performance of all three above described explanation generation and validation methods we decided to conduct an experiment with all methods on the same data set and to answer three general questions (Q[n]):

1. Which method has a better validation performance for a common query set?
2. Which method performs validation better over time?
3. Which method has the best performance for which type of FPs?

For all questions, specific aspects of each method should be considered:

- CBR-Explainer-2 is a more advanced, however, also more restricted version of CBR-Explainer-1 in terms of the validation process, i.e., it uses more attributes for validation and does not adapt weights in favor of the number of detected explanation patterns. From a reversed point of view, the CBR-Explainer-1 is a light and more permissible method.
- DA-Explainer is a non-CBR method, that is, it is not fully adapted to the main distributed case-based paradigm of MetisCBR. However, it provides a different point of view at the validation problem as it is able to apply fuzzy validation estimation with different classes (see Sect. 4.5).

5.1 Setting

The evaluation was performed on a common set of 120 retrievable and explainable architectural designs constructed with a special web-based user interface [20]. The designs of the data set were available in different abstraction and complexity levels, see Fig. 6. For validation, the CBR-Explainer-1 and the CBR-Explainer-2 used their own validation base, however, cases in all of these bases referred to the same ground-truth cases provided in the very first validation base (of the CBR-Explainer-1).

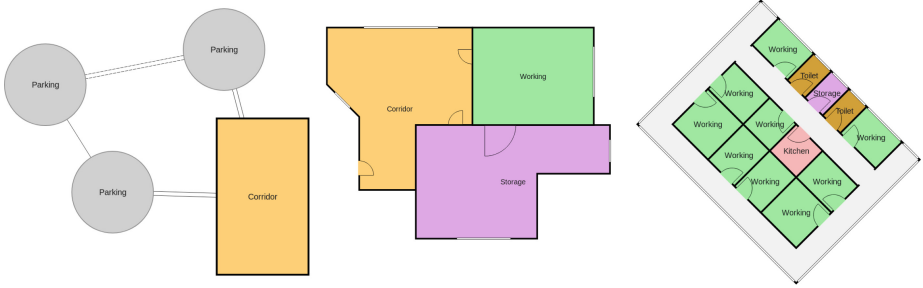


Fig. 6. Examples for different abstraction levels of the floor plans contained in the tested case base. From left to right: *abstract*, *semi-abstract*, *non-abstract* (complex).

5.2 Results for Q1

In Q1, we were interested in a general performance of each explanation method for a given, common for all three, amount of queries. To accomplish this, we sent 24 different queries for retrieval and subsequent explanation process for each of the methods. Each query included 2 FPs, i.e., consisted of 2 sub-queries, all FPs were the same for each method.

In Fig. 7, the total number of produced explanations is shown. In Fig. 14, the percent of valid explanations for each of the methods is shown. Following validity criteria were applied for the methods:

- CBR-Explainer-1, CBR-Explainer-2: the explanation is valid if its validation similarity value v_{max} exceeds the threshold value of 0.5.
- DA-Explainer: the explanation is estimated valid if the result set’s validation class *is not* V4 and the validation estimation value of the single result is above the threshold of 1.5.

The results of Q1 showed that the DA-Explainer was generally able to predict the amount of possibly valid explanations for both CBR-Explainers, despite its very low total number of explanations produced (which however was developed/implemented on purpose using the discriminant analysis). The both CBR-Explainers also showed a good general rate of valid explanations (Fig. 8).

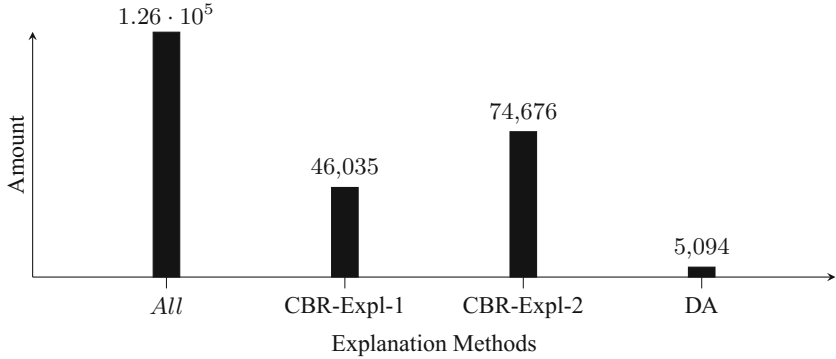


Fig. 7. The total numbers of explanations produced.

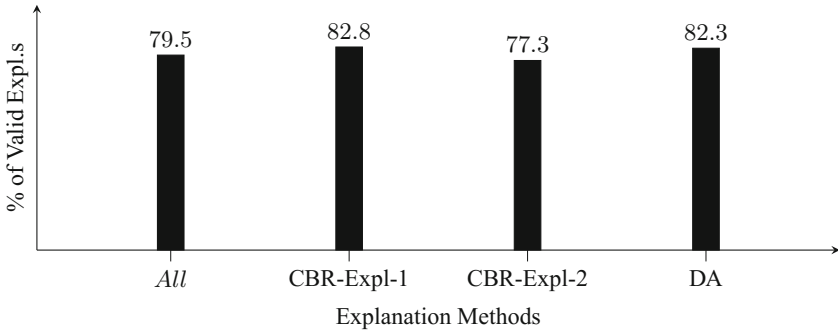


Fig. 8. The percent number of valid explanations.

5.3 Results for Q2

In Q2, we measured how each of the methods was performing over time, i.e., we also recorded the validation stepwise, after every 4th query (8th sub-query). The reason to perform this measurement was the question of how good or bad the validation percentage will be if the system will run for a longer time. A total number of 6 steps or sub-measurements of validation rate was produced, the results are shown in Figs. 9, 10, and 11.

As the results of Q2 show, the CBR-Explainer-1 shows the most constant performance in this measurement, only once its rate falls below the 82% rate, remaining between 82 – 83% for the other sub-measurements. CBR-Explainer-2 is constant as well, however, its start rate is lower. The most inconstant is the DA-Explainer, its final rate (and even the second) is much lower as its start rate.

5.4 Results for Q3

In Q3 we aimed at exploring the performance of the methods for different types of semantic fingerprints. The reason for this measurement is our intention to

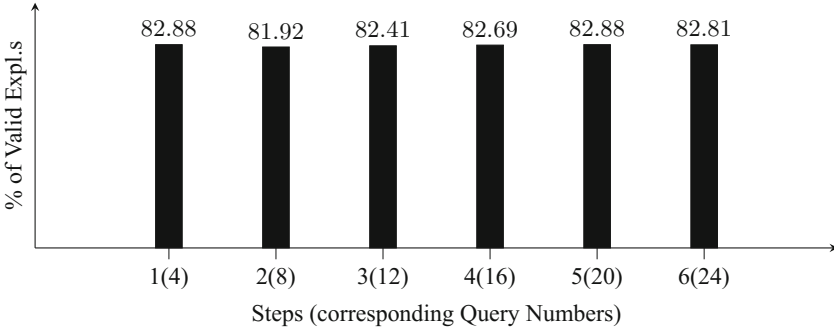


Fig. 9. The stepwise measurement of valid explanations for CBR-Explainer-1.

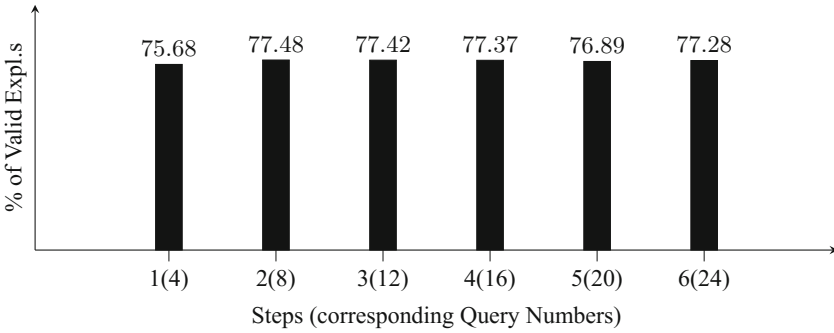


Fig. 10. The stepwise measurement of valid explanations for CBR-Explainer-2.

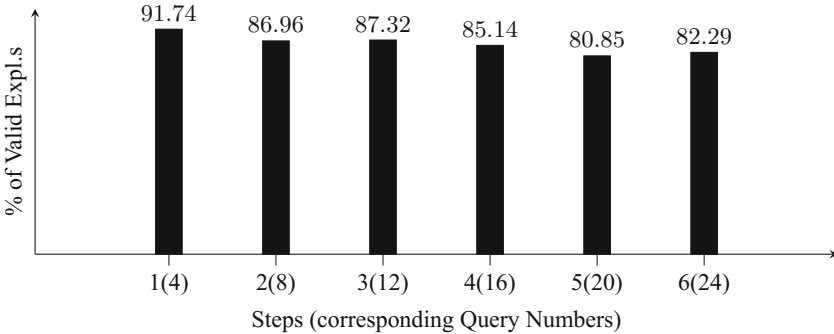


Fig. 11. The stepwise measurement of valid explanations for the DA-Explainer.

implement an automatic selection method for choosing the proper explanation method for the corresponding FP type. The results of Q3 for each method are shown below (Fig. 12).

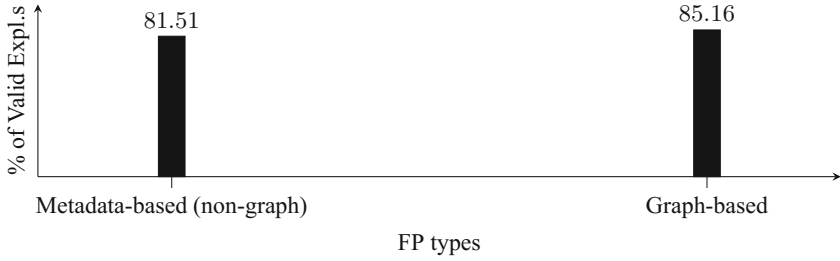


Fig. 12. The FP-type-wise measurement of valid explanations for the CBR-Explainer-1.

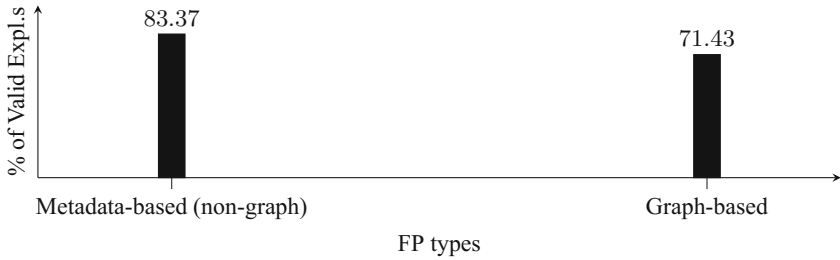


Fig. 13. The FP-type-wise measurement of valid explanations for the CBR-Explainer-2.



Fig. 14. The FP-type-wise measurement of valid explanations for the DA-Explainer.

The results of Q3 showed that for this measurement DA-Explainer performed better on graph-based FPs, that is, its prediction feature can be more trusted for this type of FPs and should be improved for non-graph-FPs. In contrast to the DA-Explainer, the CBR-Explainer-2 performed better on non-graph-based FPs, however the more strict behavior of this explainer is the most probable cause for its results. The CBR-Explainer-1 showed constant values in this measurement, however, for particular types, other explainers performed (slightly) better (Fig. 13).

6 Conclusion and Future Work

In this paper, we presented three methods for explanation generation and validation for results of retrieval within MetisCBR, a case-based and multi-agent based framework for support of the early conceptual phases in architecture. All three methods were described and evaluated in a comparative evaluation that aimed at examination of the current state of their development.

Our future work in this area will be concentrated on improvement of the methods presented in this work. For example, the next study we are planning is aimed at detailed examination of the prediction feature of the DA-Explainer to determine if the results and their explanations predicted valid are considered valid by case-based explainers. Also, an explainable BDI-Agent (see Sect. 2) will be conceptualized and implemented as another alternative.

References

1. 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. International Conference on Agents and Artificial Intelligence (ICAART-18), Scitepress (2018)
2. 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>
3. 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. 195–209. Springer, Heidelberg (2004). <https://doi.org/10.1007/978-3-540-28631-8.29>
4. Cassens, J., Kofod-Petersen, A.: Designing explanation aware systems: the quest for explanation patterns. In: ExaCt, pp. 20–27 (2007)
5. Broekens, J., Harbers, M., Hindriks, K., van den Bosch, K., Jonker, C., Meyer, J.-J.: Do you get it? user-evaluated explainable BDI agents. In: Dix, J., Witteveen, C. (eds.) MATES 2010. LNCS (LNAI), vol. 6251, pp. 28–39. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16178-0.5>
6. Harbers, M., van den Bosch, K., Meyer, J.J.: Design and evaluation of explainable BDI agents. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 2, pp. 125–132. IEEE (2010)
7. Voss, A.: Case design specialists in FABEL. In: Issues and Applications of Case-Based Reasoning in Design, pp. 301–335 (1997)
8. Oxman, R., Oxman, R.: Precedents: memory structure in design case libraries. In: CAAD Futures. vol. 93, pp. 273–287 (1993)
9. Flemming, U.: Case-based design in the SEED system. *Autom. Constr.* **3**, 123–133 (1994)
10. Lai, I.C.: Dynamic idea maps: a framework for linking ideas with cases during brainstorming. *Int. J. Architectural Comput.* **3**, 429–447 (2005)
11. 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)

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. Richter, K., Heylighen, A., Donath, D.: Looking back to the future—an updated case base of case-based design tools for architecture. In: Knowledge Modelling-eCAADe (2007)
14. Richter, K.: Augmenting Designers’ Memory: Case-based Reasoning in Architecture. Logos-Verlag, Berlin (2011)
15. 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)
16. 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, pp. 50–60 (2017)
17. Ayzenshtadt, V., et al.: Comparative evaluation of rule-based and case-based retrieval coordination for search of architectural building designs. In: Goel, A., Díaz-Agudo, M.B., Roth-Berghofer, T. (eds.) ICCBR 2016. LNCS (LNAI), vol. 9969, pp. 16–31. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47096-2_2
18. Jackson, M.: Problem analysis using small problem frames. S. Afr. Comput. J. 47–60 (1999)
19. Espinoza, C.: Case-based classification of explanation expressions in search results of a retrieval system for Support of the early conceptual design phase in architecture. Bachelor Thesis. University of Hildesheim (2018)
20. 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