Multi-Agent-Based Generation of Explanations for Retrieval Results Within a Case-Based Support Framework for Architectural Design

Viktor Ayzenshtadt[1], Christian Espinoza-Stapelfeld[1], Christoph Langenhan[3] and Klaus-Dieter Althoff[1,2]

[1]*University of Hildesheim, Institute of Computer Science, Samelsonplatz 1, 31141 Hildesheim, Germany*

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

[3]*Faculty of Architecture, Technical University of Munich, Arcisstrasse 21, 80333 Munich, Germany*

Keywords: Multi-Agent Systems, Case-Based Agents, Explanations, Architectural Design Support.

Abstract: In this paper, we describe the general structure and evaluation of a multi-agent based system module that was conceptualized to explain, and therefore, enrich the search results of the retrieval process within a distributed case-based framework for support of early conceptual design phase in architecture. This explanation module is implemented as an essential part of the framework and uses case-based agents, explanation ontology, and explanation patterns as its underlying foundational components. The module's main goal is to provide the user with additional information about the search results to make the framework's behavior during the retrieval stage more transparent and traceable. System's justification for displaying of results plays an important role as well, and is also included in the explanations. We evaluated the explanation generation process with a ground-truth set of explanations and a case-based validation process to ensure the suitability of the generated explanation expressions for displaying in user interfaces connected to the framework. The results of the evaluation confirmed our expectation and showed the general validity of the explanations.

1 INTRODUCTION

Computer-aided architectural design (CAAD) is a research domain where one of its main tasks is to utilize knowledge-intensive data about building designs to support the early phases of architectural design conceptualization. Case-based reasoning (CBR) is a widely-used approach that can make use of such data to provide designs similar to the currently created one. Those similar designs are aimed to inform an architect about possible similar semantic and relational structures of the prior designs found in a database (case base), or show how the problem at hand was solved in a different context. Other important objective of similar designs (floor plans) is to provide the architect with inspirational space to explore the designs that can be helpful during the current project.

Many *case-based design* (CBD) approaches were developed to utilize the methods of CAAD and CBR to provide a tool for the support of the architectural conceptual design phase. MetisCBR (Ayzenshtadt et al., 2016a) is one of such approaches that was created as part of the basic research project *Metis* that was aimed to combine the research areas of CAAD, CBR, and multi-agent systems (MAS). MetisCBR uses case-based agents and CBR-based retrieval methods to find possibly helpful similar architectural designs that correspond to the criteria entered by the user. These criteria are based on a paradigm of *semantic fingerprint*, that hierarchically represents meta data and structural properties of a floor plan.

However, not only the information about the retrieved similar designs themselves is important for speeding up and qualitative improvement of the architectural design process. In many cases, users, in our case: architects, want to have additional information about the results: e.g., *how exactly* the system achieved the given results, *why exactly* the results can be helpful, or what are the structural differences of query and result. That is, the users want the system to be explanation-aware in order to provide *explanations* for the returned search results.

In this paper, we present the recently developed *explanation module* of MetisCBR, where case-based agents work with *explanation patterns* to generate explanations for the results returned during the retrieval process. After the generation, the explanations are validated by a case-based comparison with an expert-

Ayzenshtadt, V., Espinoza-Stapelfeld, C., Langenhan, C. and Althoff, K-D.
Multi-Agent-Based Generation of Explanations for Retrieval Results Within a Case-Based Support Framework for Architectural Design.
DOI: 10.5220/0006650201030114
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 1*, pages 103-114
ISBN: 978-989-758-275-2

proven set of ground-truth explanations saved in a special case base of explanations.

This paper is structured as follows: first, the internal MetisCBR-related work as well as external work related to the purposes of the framework, and explanations in CBR will be briefly presented. In the next section, we describe MetisCBR more in detail, including its underlying foundations for CBR-based retrieval and internal communication of agents. Next, we in detail present the explanation module (*Explainer*), including the general functionality, the characterization and functionality of the agents, description of explanation patterns and their use and modification for the *Explainer*, and the underlying communication pattern and its ontology. After that, the validation-based evaluation of the generated explanations is presented. The conclusion section summarizes the presented work and gives a brief overview of the future development of MetisCBR and the explanation approach.

2 RELATED WORK

In this section, we present work related to the purposes of MetisCBR, dividing it in external and internal related work, as well as a short overview of work that has been made so far in the domain of CBR-based explanations.

2.1 Internal Related Work

During the basic research project *Metis* (KSD Research Group, 2015) that aimed at combination of MAS, CBR, and CAAD for architectural design support purposes, we developed a number of approaches to reach this research goal. MetisCBR is one of these developed approaches; a web-based floor plan editor Metis WebUI (Bayer et al., 2015) that can communicate with retrieval approaches, a retrieval coordination middleware KSD Coordinator (Roith et al., 2016) that can query a number of (subgraph matching-based) design retrieval engines, an android app, and a touch-table-based application are other examples. A comprehensive framework that unites some of these techniques has been constructed as well and presented in detail in (Sabri et al., 2017). Theoretical foundations developed during the project are the semantic fingerprint (see also Section 3.2) and the specification of the architectural graph description language AGraphML (Langenhan, 2015).

2.2 External Related Work

A number of CBR-based approaches was introduced in the past for the conceptual phase of the architectural design process. In (Heylighen and Neuckermans, 2001) and (Richter et al., 2007), the corresponding authors provide surveys of these approaches. In (Richter, 2011), a comprehensive research into CBR in architecture/CAAD was conducted to provide a view on present, past, and future of combination of these research areas. In (Richter, 2013) a summary of this work is available that includes an overview of the current problems of CBR in architecture.

Notable approaches are FABEL (Voss, 1997), PRECEDENTS (Oxman and Oxman, 1993), SEED (Flemming, 1994), DIM (Lai, 2005), or VAT (Visual Architectural Topology, a semantic representation approach) (Lin, 2013). An approach that is most likely closely related to the purpose of this paper is CaseBook (Inanc, 2000), where a *similarity explanation report* is mentioned as one of the system's functionalities. However, not much informaion is provided about its mode of operation or evaluation.

The most notable approach from the research area of multi-agent systems is (Anumba et al., 2007). In this comprehensive work, examples and technical foundations of use of MAS in construction and architecture domains are examined.

2.3 Explanations in Case-based Reasoning

Explainability of case-based systems is one of the questions of the CBR research field that has gained much attention during the last decades, producing a number of theoretical foundations and technical terms. Aamodt, one of the founders of the CBR's 4R cycle (Retrieve, Reuse, Revise, Retain), was one of the first authors to examine explanations for CBR systems in (Aamodt, 1993). Roth-Berghofer described foundational issues of explanations in CBR in (Roth-Berghofer, 2004). Later, an explanation-aware plugin myEACBR for the CBR prototypes framework *myCBR* (Bach and Althoff, 2012) was developed in (Lillehaug, 2011). Finally, in (Cassens and Kofod-Petersen, 2007), application of explanation patterns to CBR-based systems or intelligent systems in general was examined.

We consider this previous work a valuable contribution to the CBR research area and explanation-aware systems. Our goal, therefore, was to take the above described research as a basis and adapt it for our specific purpose, the distributed case-based support for the architectural conceptualization phase.

More specifically, the pattern approach described in (Cassens and Kofod-Petersen, 2007) was considered very similar to semantic fingerprint of architecture (see Section 3.2) used by MetisCBR for retrieval, as semantic fingerprints can be considered retrieval patterns as well, which is documented in (Sabri et al., 2017). This led us to adaptation of the explanation patterns from (Cassens and Kofod-Petersen, 2007) for our explanation module. That is, our aim in this case was to achieve a sufficient degree of consistency between the system modules, using an established method and workflow for new agents. However, some modifications during the application process of explanation patterns were required, especially for detection of patterns, as tasks of retrieval and explanation differ in their nature (see Sections 4.2.1 and 4.2.2).

3 MetisCBR

MetisCBR is a multi-agent system that makes use of *case-based agents* and *CBR-based retrieval* to find possibly helpful previous solutions during the early phases of architectural conceptual design. The framework's main aim is to efficiently improve the design process in its early stages by providing the architects with a tool that can find designs suitable for improvement of the project at hand. The detailed descriptions of the framework's system architecture and the domain model were already provided in (Ayzenshtadt et al., 2015) and (Ayzenshtadt et al., 2016a). The framework has been subject of different comparative performance and qualitative evaluations with participation of other search methods, examples are (Ayzenshtadt et al., 2016b) and (Sabri et al., 2017).

The decision to design MetisCBR as a multi-agent based software has a number of important reasons. First, a system with agents that can communicate with each other by means of applying a common language and ontology allows us to unify the overall communication of the system modules. This is relevant, as the queries and results of architectural designs used by the system use a strict specification for their construction and thus can be converted to an ontological object for application in multi-agent frameworks. Second, some system components require to be able to conduct a number of parallel operations, that should not always be executed permanently, but, for example, periodically. Agents design in many multi-agent frameworks allows for execution of behaviors of different types concurrently – this fact was also important for our choice of the system design paradigm. Third, a requirement for each retrieval system within the Metis project was to implement a coordination component that is able to select a proper retrieval method or strategy for the current query. Coordination of system processes has a long tradition in the research field of multi-agent systems, therefore, this fact played an essential role in our decision as well. As described in (Ayzenshtadt et al., 2015), we apply adapted methods of coordination provided in (Nwana et al., 1996).

Further in this section, we provide a brief description of the general functionality of MetisCBR, and the underlying concepts of semantic search patterns (*semantic fingerprints*) and communications patterns. Both of the pattern types are relevant for the purpose of this paper.

3.1 General Functionality

The general functionality of MetisCBR is based around the so-called *retrieval containers* that work concurrently on resolving of search requests. In every container, a number of CBR retrieval agents, that retrieve the case base of floor plans in order to find similarly structured designs according to the selected retrieval strateg, and a *Manager*, that controls the retrieval agents and manages the resolving of the current query, are available. The search request itself can consist of different sub-requests, where each of these sub-requests represents a semantic fingerprint (see Section 3.2) selected by the user (in this case, a retrieval container is created for each sub-request, the particular results of the containers are then amalgamated into a common result). On the higher level of the system's hierarchy, the *Coordinator* agent governs the system's processes, e.g., it selects the proper retrieval strategy for the current search request (or a sub-request) and assigns it to the corresponding retrieval container. The *Coordination Team* is a group of Coordinator's helper agents that consists of agents for specific tasks, such as result collection from the active retrieval containers. The agents communicate by means of applying the specific communication patterns that allow for standardization and unification of processes within the system (see Section 3.3). The general system structure *with inclusion of the explanation component* is depicted in Figure 1.

3.2 Semantic Fingerprint

Semantic fingerprint of architecture (Langenhan and Petzold, 2010) provides a description paradigm for representation of architectural designs in CAAD systems. The concept of semantic fingerprint is aimed at enrichment of the BIM (Building Information Modeling) objects with semantic and topological information contained in graphs derived from a floor plan.

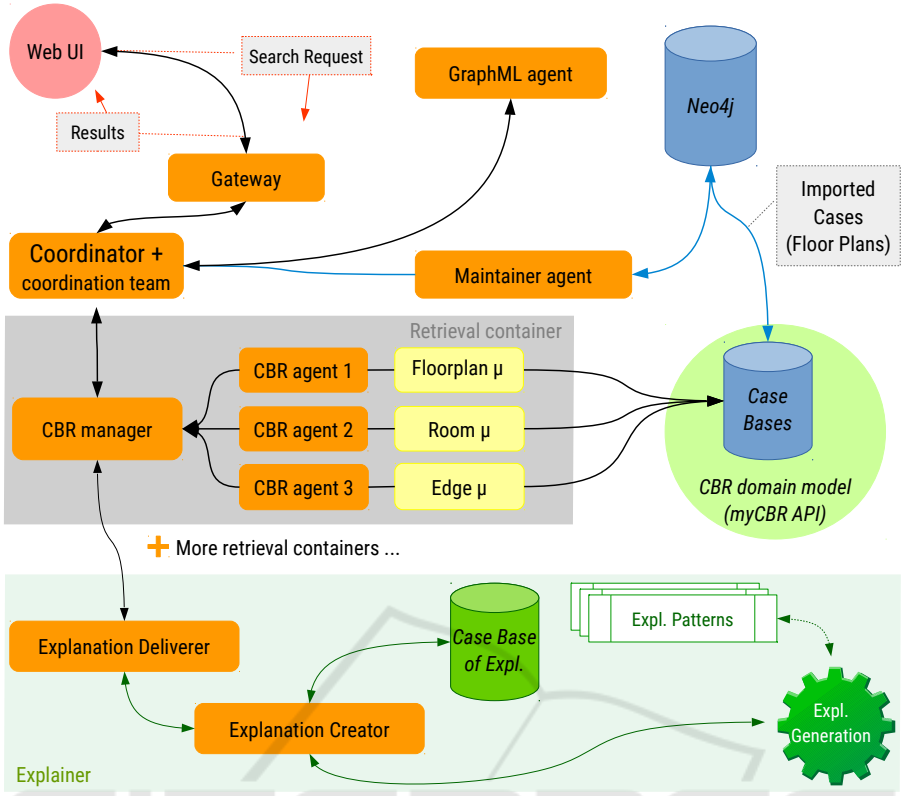

Figure 1: General structure of MetisCBR (with explanation module, the *Explainer*, included).

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

Figure 2: Semantic fingerprints currently implemented in MetisCBR.

Therefore, each fingerprint is a graph or a subgraph that can contain certain semantic and/or topological information about the structure, e.g., room geometry, of an architectural design. Semantic fingerprint follows an index-based hierarchical structure with *Levels* that are divided in *Units* which contain certain *Zones* with a number of *Rooms*. For retrieval purposes, semantic fingerprints can be used as search patterns. In this way, they are implemented in MetisCBR, and can be specified by the user during the retrieval session. A list of fingerprints (FPs) currently in use in MetisCBR is shown in Figure 2.

3.3 Communication Patterns

The *communication patterns* of MetisCBR are the internal communication workflows that the agents make use of during execution of their tasks in order to collaborate with each other. The communication patterns were previously discussed in (Ayzenshtadt et al., 2016c), in this section, similarly to the general structure of MetisCBR and semantic fingerprint, we give only a brief description of the concept.

A communication pattern is a directed graph for one of the system workflows, e.g., retrieval, where nodes represent the agents involved in this workflow, and edges represent steps or actions that are predefined and either assigned by an agent of managing type, such as Coordinator or Manager, or requested by agents of another type. For each pattern, a subset of the system's *communication ontology* is used to ensure the standardized communication among the agents. The ontology, and thus a subset as well, consists of three layers: *object:* abstract concepts of query and result, *data:* concrete architectural data from query or result, *action:* an action that should be executed with this data. Therefore, the pattern steps are named after the ontology concept (*action class*) that holds information about this type of action. In Figure 3, the communication pattern 'retrieval' is depicted for demonstration of a pattern structure.

4 EXPLANATION MODULE (EXPLAINER)

Users of (decision) support systems are likely to increase their satisfaction with the system if the system is able to produce and present reasonable traces of its behavior. Therefore, our motivation behind the development of the explanation component is to give our user group, the architects, a sufficient amount of insights into how the system achieved the results contained in the final result list. In this section, we in detail present the foundations of the explanation module *Explainer*: agents, explanation patterns, communication pattern, and construction and validation of explanation expressions.

4.1 Explainer Agents

As depicted in Figure 1, the general structure of the Explainer employs two agents that are responsible for construction, validation and transfer of explanations: the *Explanation Deliverer* and the *Explanation Creator*.

4.1.1 Explanation Deliverer

The Deliverer agent of the explanation module is responsible for receiving of explanation requests from the managers of the active retrieval containers and sending back the corresponding explained results to these managers. After receiving an explanation request, that at first consists of the query that has to be explained, the Deliverer saves this query in a specific *explanation schedule* and waits for retrieval results to be completed. When the resolving of the query is finished by the corresponding retrieval container, the manager of this container sends the results to the Deliverer. The Deliverer then retrieves the query from the schedule, constructs an object that consists of the query and the corresponding result set, a so-called *query-result object*, and sends it to the other agent of the explanation module, the *Explanation Creator*. After receiving the explained results from the Creator, the Deliverer forwards them to the corresponding manager who in turn sends them to the *Result Collector* agent of the Coordination Team (previously mentioned in Section 3, see more about Result Collector's role in 4.1.3).

4.1.2 Explanation Creator

The Explanation Creator agent is responsible for the creation/construction of explanations for the result set of a query. After receiving the query-result object from the Deliverer, the Creator tries to generate an explanation for each result in the result set of this object, *based on the semantic fingerprint of the query*. The explanation is then added to the corresponding result in the result set. At this point, the query-result object contains the explained results and is sent back to the Deliverer. The Creator's mode of operation during the actual explanation generation process consists of two subsequent phases:

1. Detection of explanation patterns (see Section 4.2.1) and creation of explanation texts with predefined expressions (see Section 4.2.1)

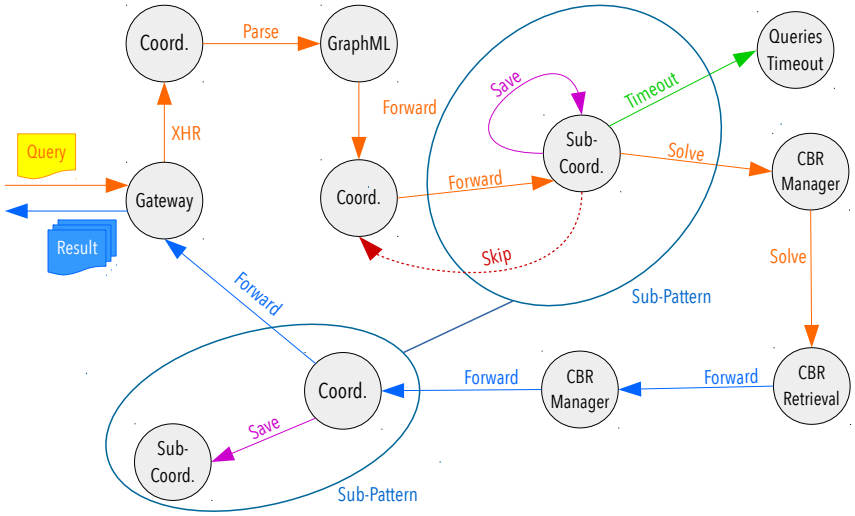

Figure 3: The communication pattern 'retrieval' (including the sub-pattern for retention of the currently entered query).

2. Case-based validation of explanations (see Section 4.3)

4.1.3 Other Agents

A special role in the explanation process is assigned to the Result Collector agent, despite the fact that this agent does not belong to the explanation module agents. The Result Collector receives the set of explained results for each fingerprint (i.e., sub-request) of the query and, while amalgamating the fingerprint results (i.e., computing the common similarity value for the complete result), it selects the most valid explanation (see Section 4.3) among all available explanations and adds it to the result information in the final result set that is then presented to the user.

Another important role is played by the Manager agent of a retrieval container, who is, among its other functions, the communication interface between the Explainer and the Result Collector. However, the Manager does not apply any modifications to the result data (where the explanations are included).

4.2 Explanation Patterns

Explanation patterns for case-based reasoning applications and approaches is a concept developed in (Cassens and Kofod-Petersen, 2007). In the narrower sense, the explanation patterns are also known as *Explanation Problem Frames* and extend the original concept of problem frames presented by Jackson in (Jackson, 1999). Taking the Jackson's problem frames as a basis for description of problems in software engineering, the explanation problem frames for intelligent systems were contructed in (Cassens and Kofod-Petersen, 2007) and specified a number of special patterns that serve as foundation for creation or generation of explanations in such systems.

Following the structure proposed by Jackson, the explanation problem frames consist of a computation unit *machine*, representation of a real-world domain, and the reference to the requirements for a problem that has to be solved. All three components of a problem frame should be connected with each other in order to provide a common foundation for creation of an explanation. In (Cassens and Kofod-Petersen, 2007), a number of explanation problem frames were identified for CBR-based systems, that are aimed at achieving, inter alia, the following, expert systems-inspired, *explanation goals*:

- *Justification* – Explanation of *why* the given answer or solution is good at this point of working/interacting with the system. The goals is to make the user confident with the system's decision-making by *reasonably justifying* the answer as suitable and helpful for the current situation, question, or query.
- *Transparency* – Explanation of *how* the system found the given answer or solution. The goal is to give the user some insight into how the system's reasoning works, that is, make the decision-making processes of the system more *transparent*.
- *Relevance* – Explanation of why the question that the system is asking is relevant in the current situation. For example, if an insufficient amount of properties or attributes of the failure was provided by the user to find a proper solution, the system may ask for further refinement of the problem description, thus ask the user for more relevant data.

However, the system should be able to provide a reasonable explanation of why this refinement is relevant, i.e., in this example, why the data provided by the user is insufficient.

4.2.1 Modification for MetisCBR

In a recent work in (Espinoza, 2017), the general concept of explanation patterns was extended for use in MetisCBR's results explanation module. For this purpose, the explanation patterns described in Section 4.2.1 were modified to work with the underlying concepts of MetisCBR, such as semantic fingerprint and the domain model of the case base of architectural designs. Based on the machine/domain/requirement structure of the problem frames, the explanation patterns were adapted for the domain of semantic fingerprint-based CBR-retrieval of architectural designs, taking the fingerprint context (more precisely, constraint) as its basis.

The explanation patterns for MetisCBR were conceptualized w.r.t explanation goals, previously described in 4.2.1. That is, three explanation patterns were created to extend MetisCBR with the explainability function. For each pattern, the fingerprint constraint serves as a basis for the problem frame structure, it is then integrated with justification, transparency, and relevance frames. In Figure 4 the integration of patterns for justification and transparency is shown: a new machine *Fingerprint* has been added to the previously existing Justification and Transparency to allow for connection of these original patterns to the main properties (meta data, rooms, edges) of the fingerprint- and CBR-based retrieval of floor plans.

4.2.2 Detection of Patterns

The explanation patterns were implemented in MetisCBR in the way, that the system (more precisely, the Explanation Creator agent) tries to detect if a pattern is available for the current query and the current result. Only if the pattern is available, the Creator will add the corresponding explanation expression to the explanation text string of the result.

The detection conditions are saved in a special heuristic ruleset and are different for each pattern and fingerprint. That is, it depends on complexity and type of the fingerprint (FP) if the pattern can be detected. Below, some examples are provided to demonstrate heuristics of rules of pattern detection:

- `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.

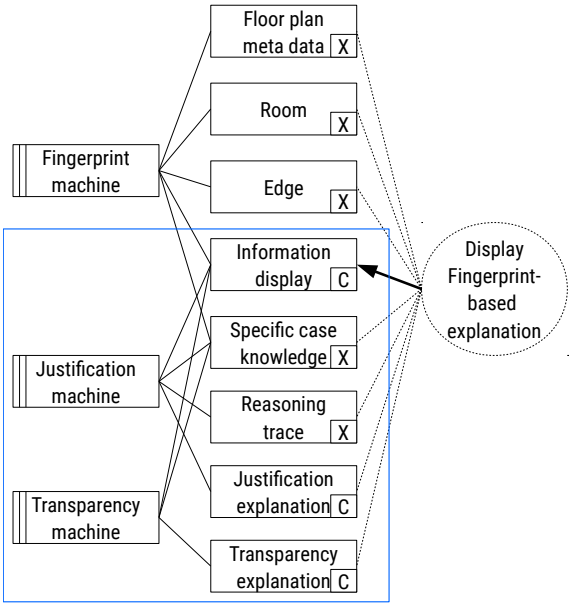

Figure 4: Modification of explanation patterns for MetisCBR. Enclosed in blue are the original problem frames from (Cassens and Kofod-Petersen, 2007). C stands for the goal of explanation, X denotes the system knowledge. The *Information Display* goal is mandatory for connection to the user interface.

- `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.

As seen in the examples provided above, the *similarity grade* plays an important role in the detection of the explanation patterns. This similarity grade depends on the *similarity value* of the result floor plan and is classified as follows: as very similar we classify the result if its 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 $Sim < 0.25$.

4.2.3 Construction of Explanations

The actual explanation is a combination of explanation expressions as text strings, where for each of the patterns for each of the fingerprints and for each of

the similarity grades a short and long (full) version of an expression exist. An explanation text is then a concatenation of particular expressions, it is possible to combine both short and full versions of an expression, however, either short or full version of the same category (similarity grade, fingerprint, or pattern) should be used. In Figure 5, examples of the explanation texts are shown for an exemplary query and the corresponding result.

All text strings for full explanations were provided by an expert from the architectural design domain to ensure the suitability and understandability of explanations for the representatives of this domain.

4.3 Validation of Explanations

The aim of validation of explanations is to prove if the generated explanation corresponds to the architectural understanding and technical term requirements. We consider the validation an important step of the process, as it helps to increase the possibility of returning of more usergroup-friendly explanation expressions.

To validate an explanation we decided to apply a case-based validation approach. That is, an explanation is validated against a ground-truth set of explanations saved in a case base of explanations. Each explanation in this set consists of a number of attributes (see Table 1), most of those attributes are used for comparison with their counterparts from the generated explanations. The unused attributes are used for information purposes.

After the attribute-wise comparison, where for each relevant attribute a similarity value is computed, these attribute similarity values are amalgamated with a weighted sum (see (1)) to produce a *validation similarity* value v, that shows how similar the query explanation is to the currently compared ground-truth explanation. The weight values for the attribute similarities used in this weighted sum depend on the number of detected explanation patterns. For example, if two patterns were detected, the weight value ω_t for text similarity t is 0.2, the weight ω_p for a pattern similarity p is 0.4 (if only one pattern was detected: 0.3/0.7 accordingly). This dynamic distribution of weights was applied in order to avoid penalization of results where not all of the patterns could be detected, but a highly valid explanation can be expected anyway. That is, we are interested in results with most reasonable explanations possible.

The highest validation similarity that indicates the most similar explanation from the set of all validated similarity values V then becomes an *explanation similarity* of the query similarity, i.e., the currently generated explanation. The generated explanation is valid if its explanation similarity exceeds a threshold value, e.g., 0.5.

$$v = \omega_t t + \omega_p \sum_{i=1}^{n} p_n, \quad v \in V \tag{1}$$

4.4 Explainer Communication and Ontology

To exchange information among each other, the agents of the Explainer make use of the communication patterns described in 3.3. That is, they follow the workflow communication steps defined in the pattern. For the Explainer, a specific communication pattern 'explainer' was created that underlies all communication processes within the explanation module. Similarly to the 'retrieval' pattern shown in Figure 3, where *Query* and *Result* objects hold the concrete information about the floor plans from the corresponding query and result, the 'explainer' pattern contains the *Explain* object that contains information about data that should be enriched with explanations. In this case, this is the previously mentioned query-result object that includes query and result data before, and is enhanced with explanation expressions for each result after the explanation generation process. The communication pattern 'explainer' is shown in Figure 6.

The 'explainer' communication pattern relies on the explainer ontology that contains all possible concepts that can occur during collaboration between both agents of the Explainer (Deliverer and Creator) and agents of the retrieval module (Manager, Result Collector). A special concept is *Explanation*, that consists of data for explanation validation as described in Section 4.3. All ontological concepts are represented as objects in FIPA-SL content language.

5 EVALUATION

To evaluate the current functionality of the Explainer, we decided to conduct a comprehensive case-based validation of generated explanations by means of querying MetisCBR with a number of different queries. The process of case-based validation of an explanation is described in Section 4.3. In this section, we first give a brief description of the setting of the experiment, followed by the results achieved during the evaluation. To our knowledge, no comparable explanation tool in current architectural design support software exists at the moment, i.e., this is the first evaluation of a system module of such kind. Our general expectation was, that the Explainer can produce a sufficient number of queries that are valid for inclusion

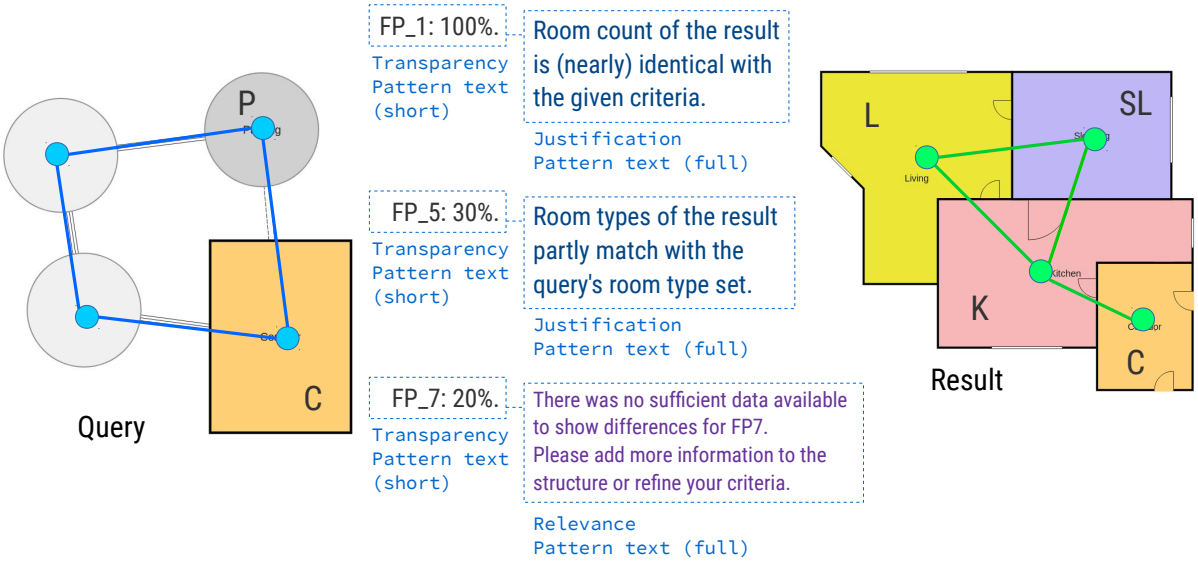

Figure 5: An examle of constructed explanations between an exemplary qurey and the corresponsing result. Bubbles denote rooms without specific functionality and/or geometry. C denotes the **C**ORRIDOR room type, others are **L**IVING, **SL**EEPING, **P**ARKING, and **K**ITCHEN. Fingerprints (FPs) are described in Figure 2.

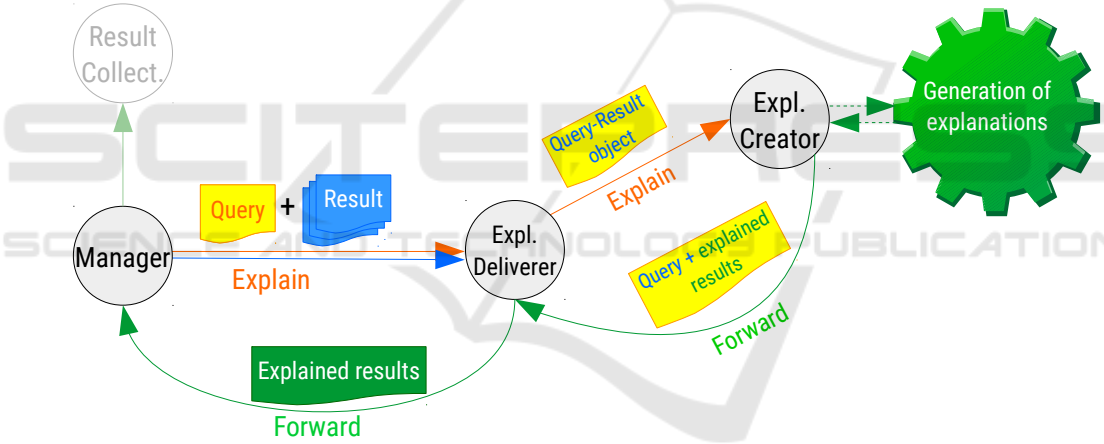

Figure 6: The communication pattern 'explainer'(the order of actions is represented clockwise, starting at Manager).

Table 1: Attributes of the *Explanation* concept in the case base of explanations.

Attribute	**Type**	**Description**	**Similarity Function**
`id`	`string`	Internal Explanation ID	*not in use*
`Case`	`string`	Reference to the case (result)	*not in use*
`Query`	`string`	Reference to the query	*not in use*
`Text`	`string`	Text of the explanation	Levenshtein distance-based sim.
`PatternJustification`	`boolean`	Justification pattern available?	boolean comparison
`PatternRelevance`	`boolean`	Relevance pattern available?	boolean comparison
`PatternTransparency`	`boolean`	Transparency pattern available?	boolean comparison

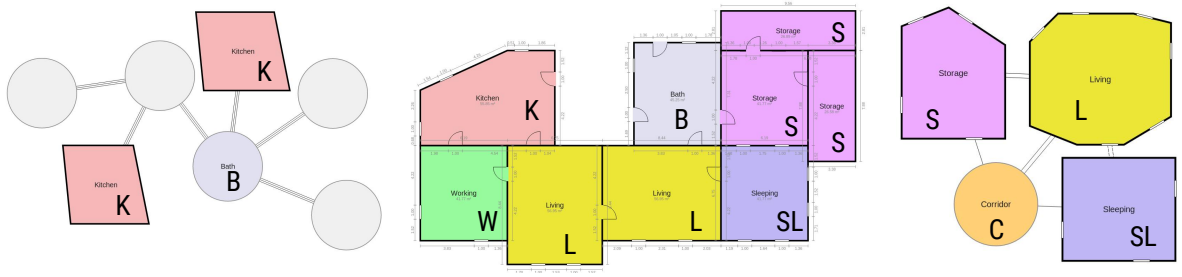

Figure 7: Examples of evaluation queries. Bubbles denote rooms without specific functionality and/or geometry. C denotes the **C**ORRIDOR room type, others are **W**ORKING, **L**IVING, **S**TORAGE, **SL**EEPING, **B**ATH, **P**ARKING, and **K**ITCHEN.

into the final result set, thus for showing them to the user group of the framework.

5.1 Setting

We performed the evaluation on a set of 225 retrievable architectural designs (floor plans) saved as cases in the case base of MetisCBR's retrieval component. The maximum complexity of a floor plan was 368, i.e., the maximum value of count $c \in C_R \cup C_E$, where C_R is the set of all room count values and C_E is the set of all edge counts. To provide a ground-truth set for the evaluation, a case base of 15 explanations (with case structure as described in Section 5) was created. This ground-truth set then became a validation base for the explanations generated during the experiment. Each explanation case in this base was manually examined to be a valid explanation itself – in this case, this means it has been checked for the technical validity of the combination query/result/similarity value/explanation. It has also been checked that the differences between the fingerprints and corresponding explanations are sufficient to provide the system with space for validation comparison, i.e., the FPs were selected in the way that they do not appear too often and the corresponding explanation expressions were all from different similarity grades.

As queries, 30 floor plans of different complexity and abstraction level were used. These floor plans were created with one of MetisCBR's floor plan editors, the Metis WebUI. In Figure 7, some examples of the query floor plans are shown. On average, 2 FPs were used per query, making 59 sub-requests in total. The FPs were selected in the similar way as in the explanations ground-truth set, i.e., they were prevented from appearing too often.

5.2 Results

The results revealed that our expectation about the general suitability of the explanation module has been met. With a total number of 67875 constructed explanations, we built a sufficient basis for evaluation of the Explainer's current capabilities. From this total explanation count, 57575 (84.825%) were valid and 10300 (15.175%) were invalid, this confirmed our assumption that the case-based validation is a reasonable method to validate such an explanation. Regardless of relatively high threshold value of 0.5 for validity rating, the system could produce a high amount of valid explanations that can be shown to the user. In the following sections, some detailed characteristics and findings from the experiment are presented.

5.2.1 Detected Patterns

The number of the explanation patterns detected for the retrieval results varied highly between the particular patterns. *Relevance* was detected in 1710 cases, whereas *Transparency* in all 67875 cases, and *Justification* in 50850 cases. However, these are expectable values, considering the fact that the system does not always require additional information from its user, i.e., does not provide questions to the user.

In Figure 9, the distribution of *combinations of patterns* is shown. Here, the expected high value for Justification/Transparency and low numbers of combinations with Relevance can also be seen.

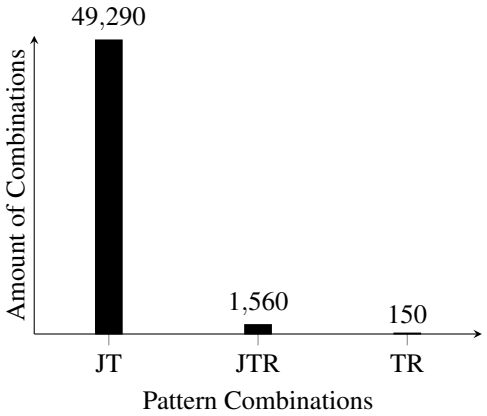

Figure 8: The distribution of explanation pattern combinations detected in the results. J, T, and R stand for Justification, Transparency, and Relevance respectively.

5.2.2 Similarity Distribution

As established for the experiments in our work, we show how the similarity grades (see Section 4.3) were distributed among the retrieval results (Figure 10), based on total count of 4525 results. Firstly, however, we show the difference between average *validation similarity* and average *results similarity* (Figure 9).

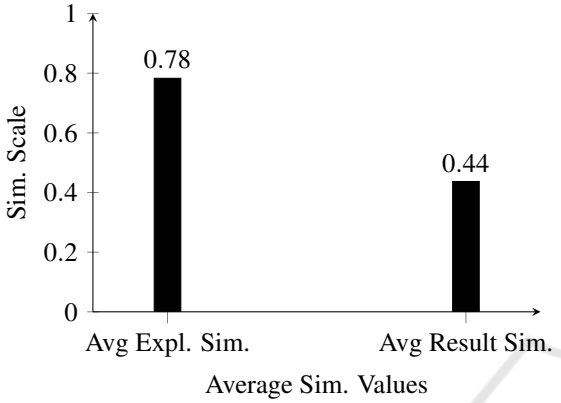

Figure 9: The average similarity values for results and explanations.

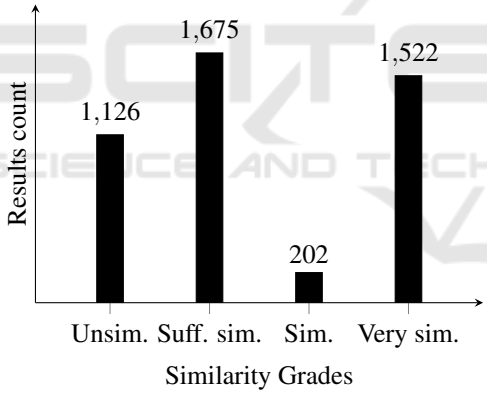

Figure 10: The distribution of results counts among the similarity grades.

6 CONCLUSION

In this work, we presented the structure and main components of the explanation module (the Explainer) for retrieval results of MetisCBR for case-based search of similar architectural designs. The module employs two agents, communication pattern and ontology, as well as explanation patterns that build the theoretical base for creation of explanation expressions.

The generated explanations are validated with a case-based validation by comparison with ground-truth explanations saved in a case base of explanations. The validation-based evaluation of generated explanations showed their general suitability for presentation to the user group as part of information in the retrieval results.

Our future work on the Explainer will be concentrated on its further development with a goal to built more fine-grained explanation patterns and detection rulesets, that will be based not only on semantic fingerprints but on the attributes of rooms, relations, and the meta data of the design graph. To evaluate these patterns, a comprehensive study with participation of architectural domain representative will be conducted.

REFERENCES

Aamodt, A. (1993). Explanation-driven case-based reasoning. In *European Workshop on Case-Based Reasoning*, pages 274–288. Springer.

Anumba, C., Ren, Z., and Ugwu, O. (2007). *Agents and multi-agent systems in construction*. Routledge.

Ayzenshtadt, V., Langenhan, C., Bukhari, S. S., Althoff, K.-D., Petzold, F., and Dengel, A. (2015). Distributed domain model for the case-based retrieval of architectural building designs. In Petridis, M., Roth-Berghofer, T., and Wiratunga, N., editors, *Proceedings of the 20th UK Workshop on Case-Based Reasoning. UK Workshop on Case-Based Reasoning (UKCBR-2015), located at SGAI International Conference on Artificial Intelligence, December 15-17, Cambridge, United Kingdom*. School of Computing, Engineering and Mathematics, University of Brighton, UK.

Ayzenshtadt, V., Langenhan, C., Bukhari, S. S., Althoff, K.-D., Petzold, F., and Dengel, A. (2016a). Thinking with containers: A multi-agent retrieval approach for the case-based semantic search of architectural designs. In Filipe, J. and van den Herik, J., editors, *8th International Conference on Agents and Artificial Intelligence (ICAART-2016), February 24-26, Rome, Italy*. SCITEPRESS.

Ayzenshtadt, V., Langenhan, C., Roith, J., Bukhari, S. S., Althoff, K.-D., Petzold, F., and Dengel, A. (2016b). Comparative evaluation of rule-based and case-based retrieval coordination for search of architectural building designs. In Goel, A., Roth-Berghofer, T., and Diaz-Agudo, B., editors, *Case-based Reasoning in Research and Development. 24th International Conference on Case Based Reasoning (ICCBR-2016), October 31 - November 2, Atlanta, Georgia, USA*. Springer, Berlin, Heidelberg.

Ayzenshtadt, V., Mikyas, A., Althoff, K.-D., Bukhari, S. S., and Dengel, A. (2016c). Ontology-based communication architecture within a distributed case-based retrieval system for architectural designs. In Mueller, E., Krestel, R., and Radtke, G., editors, *LWDA 2016*

- Lernen, Wissen, Daten, Analysen - Workshop Proceedings. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2016), September 12-14, Potsdam, Germany. Hasso-Plattner-Institut.

Bach, K. and Althoff, K.-D. (2012). Developing case-based reasoning applications using myCBR 3. In *Case-Based Reasoning Research and Development*, pages 17–31. Springer.

Bayer, J., Bukhari, S., Dengel, A., Langenhan, C., Althoff, K.-D., Petzold, F., and Eichenberger-Liwicki, M. (2015). Migrating the classical pen-and-paper based conceptual sketching of architecture plans towards computer tools - prototype design and evaluation. *11th IAPR International Workshop on Graphics Recognition - GREC'15, Nancy, France*.

Cassens, J. and Kofod-Petersen, A. (2007). Designing explanation aware systems: The quest for explanation patterns. In *ExaCt*, pages 20–27.

Espinoza, C. (2017). *Analysis of Identification of Explanation Patterns for an Explanation Module for Support of Design Phase in Architectural Domain*. Project report. University of Hildesheim.

Flemming, U. (1994). Case-based design in the SEED system. *Automation in Construction*, 3(2):123–133.

Heylighen, A. and Neuckermans, H. (2001). A case base of case-based design tools for architecture. *Computer-Aided Design*, 33(14):1111–1122.

Inanc, B. S. (2000). Casebook. An information retrieval system for housing floor plans. In *the Proceedings of 5th Conference on Computer Aided Architectural Design Research (CAADRIA)*, pages 389–398.

Jackson, M. (1999). Problem analysis using small problem frames. *South African Computer Journal*, pages 47–60.

KSD Research Group (2015). KSD - Forschungsprojekte - metis. `http://ksd.ai.ar.tum.de/?page_id=140` Last visited: 19.11.2017.

Lai, I.-C. (2005). Dynamic idea maps: a framework for linking ideas with cases during brainstorming. *International journal of architectural computing*, 3(4):429–447.

Langenhan, C. (2015). A federated information system for the support of topological bim-based approaches. *Forum Bauinformatik Aachen*.

Langenhan, C. and Petzold, F. (2010). 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.

Lillehaug, M. B. (2011). Explanation-aware case-based reasoning. Master's thesis, Institutt for datateknikk og informasjonsvitenskap.

Lin, C.-J. (2013). Visual architectural topology. In *Open Systems: Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia*, pages 3–12.

Nwana, H. S., Lee, L. C., and Jennings, N. R. (1996). Coordination in software agent systems. *British Telecom Technical Journal*, 14(4):79–88.

Oxman, R. and Oxman, R. (1993). Precedents: Memory structure in design case libraries. In *CAAD Futures*, volume 93, pages 273–287.

Richter, K. (2011). *Augmenting Designers' Memory: Case-based Reasoning in Architecture*. Logos-Verlag.

Richter, K. (2013). 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*.

Richter, K., Heylighen, A., and Donath, D. (2007). Looking back to the future-an updated case base of case-based design tools for architecture. *Knowledge Modelling-eCAADe*.

Roith, J., Langenhan, C., and Petzold, F. (2016). Supporting the building design process with graph-based methods using centrally coordinated federated databases. In *16th International Conference on Computing in Civil and Building Engineering*.

Roth-Berghofer, T. R. (2004). Explanations and case-based reasoning: Foundational issues. *Advances in case-based reasoning*, pages 195–209.

Sabri, Q. U., Bayer, J., Ayzenshtadt, V., Bukhari, S. S., Althoff, K.-D., and Dengel, A. (2017). Semantic pattern-based retrieval of architectural floor plans with case-based and graph-based searching techniques and their evaluation and visualization. In *ICPRAM*, pages 50–60.

Voss, A. (1997). Case design specialists in FABEL. *Issues and Applications of Case-based Reasoning in Design*, pages 301–335.