



Semi-automated Testing of an Architectural Floor Plan Retrieval Framework: Quantitative and Qualitative Comparison of Semantic Pattern-Based Matching Approaches

Qamer Uddin Sabri^{1,2}, Johannes Bayer^{1,2(✉)}, Viktor Ayzenshtadt^{1,3}, Syed Saqib Bukhari^{1,2}, Klaus-Dieter Althoff^{1,3}, and Andreas Dengel^{1,2}

¹ German Research Center for Artificial Intelligence,
Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
{Qamer.Uddin.Sabri, Johannes.Bayer, Viktor.Ayzenshtadt, Saqib.Bukhari,
Klaus-Dieter.Aldhoff, Andreas.Dengel}@dfki.de

² Technical University Kaiserslautern, P.O. Box 3049,
67663 Kaiserslautern, Germany

³ Institute of Computer Science, University of Hildesheim,
Samelsonplatz 1, 31141 Hildesheim, Germany
<http://www.dfki.de>

Abstract. Early design phases in architecture deal with the conceptualization of a building. During these phases, a high-level description of a building (usually coming from a contractor or customer) is iteratively turned into a first floor plan layout. One established method for architects to get inspiration is the search of references from former building projects. However, this search is usually conducted manually (and therefore labor-intensive) nowadays. Hence, an automated search for similar architectural concepts is desired. In the course of this paper, case-based reasoning and (in)exact graph matching are utilized to construct an end-to-end system for floor plan retrieval, accessible by a refined version of our design-supporting web interface. In our approach, a floor plan is modeled as a graph, where each room is represented as a node and the relations between rooms are modeled as edges. We use a set of high-level abstractions, so-called semantic fingerprints, to generate simplified graphs that are simple to match. The retrieval process itself is performed by three systems (case-based reasoning, exact graph matching and inexact graph matching), whose results are unified internally. We conducted several tests to show the deployment ability of our system: firstly, we run a stress-test for determining the computational limits our system can handle. Secondly, we tested our system qualitatively and showed that each retrieval system is superior in at least one search scenario.

This paper is an extended version of [1]. In the paper at hand, we introduce a new feature that maps components of search queries to results

Q. U. Sabri, J. Bayer and V. Ayzenshtadt have equally contributed to the paper.

and demonstrate this function by the means of a case study. Finally, we conducted an extended literature comparison of the case-based system in this area.

Keywords: Graph matching · Subgraph matching
Graph isomorphism · Architectural floor plan · Case-based reasoning
Pattern recognition · Query-result mapping

1 Introduction

During the early phases of architectural design, the architect's task is to develop a first, rough floor plan layout given a high-level description of the building. In order to accomplish this task, different working methods have been established. In general, working with references of previously completed building projects is common. However, searching such references is usually conducted manually nowadays, involving the labor-intensive and manual consultation of dedicated magazines and libraries. Speeding up this process by computerized means is therefore desired. To address this issue, we have already introduced MetisCBR [4], an approach for distributed case-based retrieval of similarly structured floor plans.

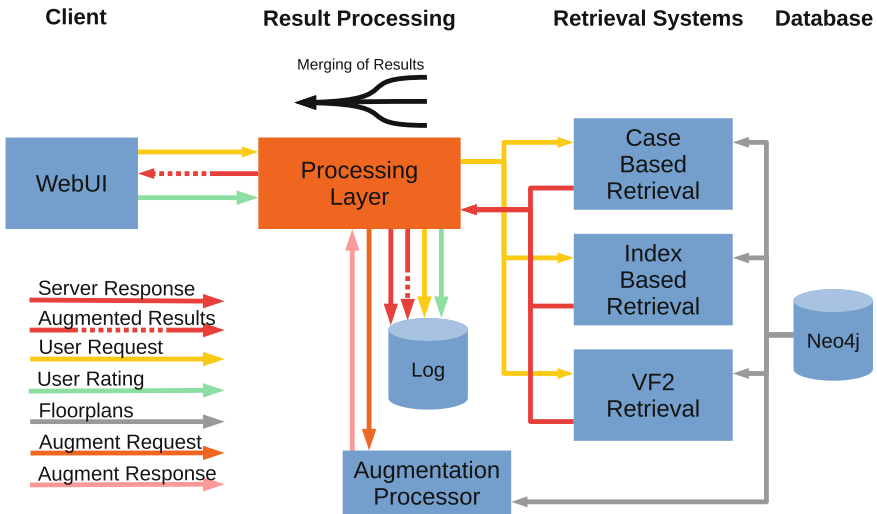


Fig. 1. Overview over the system architecture of Archistant (simplified, adapted from [1]).

In this paper, we present *Archistant*¹, an end-to-end solution for supporting the architect in conceptualizing a building (see Fig. 1 for a system overview).

¹ <http://www.dfki.uni-kl.de/archistant>.

The Archistant user interface helps the user to develop an early architectural concept. For that purpose, it is designed to follow one of the established working methods in architecture, the so-called room schedule (see Sect. 3). After such a sketch has been entered, the user can invoke the retrieval function. Archistant takes care of distributing the search query to MetisCBR and graph-matching-based retrieval systems and to collect and unify their results. The results are sent back to the user interface, where they can be contemplated by the user. Furthermore, the user is helped to reflect the results by a mapping feature, that indicates, which room in the query relates to which room in a search result.

Until now, case-based reasoning (CBR) and graph matching have been used to retrieve the similar floor plans in separately implemented systems. The novelty of Archistant is that it takes the advantages of both methods, and combines them in one common system.

This paper is structured as follows: after the problem has been motivated and the solution roughly sketched in this section, a literature review incorporating a description of the utilized user interface is given in Sect. 2. The floor plan retrieval techniques themselves as well as the query-result mapping are stated in detail in Sect. 3. Afterwards, the system is evaluated by a stress test and qualitative evaluations of the results as well as the mapping function in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Related Work

In this section, we describe work related to our research presented in this paper. We divide this related research into three main contexts: case-based reasoning, (sub)graph matching, and sketch-based interfaces.

2.1 Case-Based Reasoning

Case-based retrieval, a sub-domain of case-based reasoning, is a technique used by previously mentioned MetisCBR to find similar floor plans. Comprehensive overviews of tools and approaches related to MetisCBR are contained in studies of Heylighen and Neuckermans [14] and Richter et al. [28]. In these two overviews, the CBR-based and related approaches were compared with different features to provide the best comparison possible for both designers (in this case architects) and academic and professional staff of the knowledge-based design domain. In [2], a table-based summary of these two studies is presented which is shown in Fig. 2. Besides this overview, we also provide descriptions of the most influential approaches that inspired the creation and development of MetisCBR.

FABEL [24] is an approach that comes very close to the current purpose of MetisCBR and has served as one of the most inspirational approaches. In *FABEL*, the special modules (called *specialists* in *FABEL*) work with cases that have a multidimensional aspect-based representation in order to find the most similar ones to a given problem (a user query which is converted to such an aspect-specific structure). The database of cases (case base) inside the *FABEL*

	Storage			Input			Output						
	Floor plans + text	Abstraction	Topology	Graphic	Verbal	Adaption	Reference projects	Applying solutions	Graphical Information	Learning	Subproblems	Semantic net	Analogy
<i>Archite-II</i>	X	X			X		X		X		X		X
Domeshek et al. (1994)					X								
<i>CADRE</i>	X	X	X	X	X	X		X	X	X	X		
Hua et al. (1996)					X								
<i>FABEL</i>	X	X	X	X	X	X	X	X	X		X		X
Voss (1997)					X								
<i>IDIOM</i>			X	X	X	X		X	X				
Lottaz et al. (1998)					X						X		X
<i>PREC.</i>	X	X					X		X		X		X
Oxman and Oxman (1993)													
<i>SEED L</i>		X			X	X	X	X	X		X		
Flemming (1994)													
<i>SL_CB</i>	X	X			X	X	X	X	X				
Lee et al. (2002)													
<i>TRACE</i>		X	X	X	X		X	X	X				
Mubarak (2004)													
<i>CaseBook</i>	X		X	X	X		X						
Inanc (2000)													
<i>MONEO</i>	X	X			X		X		X				X
Taha (2006)													
<i>CBA</i>	X	X			X		X						X
Lin and Chiu (2003)													
<i>DYNAMO</i>	X	X		X	X		X		X	X		X	X
Heylighen and Neuckermans (2001)													

Fig. 2. A tabular summary of CBR tools and approaches for architectural design support, provided in [2], of the studies by Heylighen and Neuckermans [14] and Richter et al. [28]. The comparison has three main categories: *storage*, *input*, and *output*. Storage is divided in *floor plans + text*, *abstraction*, and *topology*. Input is divided in *graphic*, *verbal*, and *adaption*. Output is divided in *reference projects*, *applying solutions*, *graphical information*, *learning*, *subproblems*, *semantic net*, and *analogy*. Figure from [2].

contains the retrievable cases where identical aspects of two cases are connected by relational arcs. The retrieval algorithm of FABEL uses a so-called *fish-and-sink* approach.

The CBR-based framework *CBArch* [8] supports the construction of buildings that have a commercial background. *CBArch* aims at helping the architects and other professionals involved in a construction of such a building to improve the currently developed building design by providing alternative suggestions for its configuration. *CBArch* considers the main architectural aspects of a building (such as size) from the energy efficiency point of view. the main functionality of *CBArch* supports the CBR cycle (Retrieve, Reuse, Revise, Retain). In the retrieval phase, the feature vectors are used to compare the information from query and case to assess similarity between them. The cases are also saved in a parametric ontology-based representation for graphical representation of cases.

DYNAMO (Dynamic Architectural Memory Online) is a web-based project (described in [27]) started in 1996 to provide a case base for architecture professionals and students. The service aims at providing an easy access to architectural designs by providing searching and filtering functions for the designs in the database. *DYNAMO* is related to other CBR approaches in the use of the dynamic memory theory of Schank [30] (in *DYNAMO*'s case the architectural memory). Cases available in the case base of *DYNAMO* can be architectural designs of already existing as well as unbuilt projects, a single case consists of architectural aspects of the building as well as its graphical representation. The attribute-value-based structure underlies the representation of the cases in



Fig. 3. Screenshot of the Archistant WebUI.

the database. The retrieval process consists of two steps: in first step, an exact matching tries to determine the structurally identical cases, after that cases from the case base are selected that have at least one criterion in common. DYNAMO can also apply Data Mining techniques, such as Collaborative Filtering.

One of the first CBR-related architectural design support applications is *CADRE* (description of which is available in [29]), developed between 1990–1994. *CADRE* was constructed to work with 3D models of buildings and extends the model with some features that can emphasize its context (e.g., the environmental criteria such as street context or direction, or topological features such as room transformation). However, *CADRE* does not implement a retrieval component (a user her/himself should select a proper case from the case base) and concentrates on adaptation of solutions, i.e., the Reuse phase of the CBR cycle. In this phase, *CADRE* tries to adapt the existing solution into a new environment with given constraints. A successor of *CADRE* is the *IDIOM* system that instead of using the 3D models concentrates on the 2D-based representations of floor plans (or parts of them).

2.2 Graph and Subgraph Matching

Graph matching is widely applicable nowadays for its usability in matching and retrieval problems. In real life scenarios, there are situations when there is no exact match with the whole graph but there is a part (subgraph) that matches. If a subgraph is available, then we can use subgraph matching that tells us about the parts of two graphs that are isomorphic. Technique presented in [2] used

graph matching to retrieve the similar floor plans. This work slightly modifies the method in [19] and uses it for retrieval of similar floor plans by arranging the row-column vectors of the adjacency matrix in the decision tree. Work in [25] uses graph and subgraph isomorphism to check the similarity between a query graph and models of different buildings stored in the database. In this work, a check has also been implemented that ensures that if the query graph corresponds to some rules only then the system proceeds for checking the similarity of query graph.

2.3 Sketch-Based Interfaces

In order to make the retrieval system accessible by the user, a sketch editor is needed to enter an architectural concept. In the course of the research project Metis (see Sect. 5), two different approaches have been compared [6]. The first (Touchtect) was based on free-hand sketching, while the other (Metis WebUI) was based on polygonal rooms (Fig. 3).

The retrieval system presented here is accessible by a dedicated user interface, the Archistant WebUI. This browser-based application is an improved version of the Metis WebUI (first described in [6]). The main purpose of the Archistant WebUI is to help the user develop an architectural concept and thus generate retrieval queries. The general usability of the Metis WebUI has been shown by the means of a user study. For query construction, the Archistant WebUI uses the AGraphML [16] specification (also see Sect. 3.1).

The Archistant WebUI provides a room-oriented floor plan editor. Is designed to follow the room schedule working method as established in architecture. A room-schedule in architecture is a set of high-level requirements (usually coming from an end-customer or contractor), that has to be turned into a floor plan layout by the architect. Its formal structure is assumed to be a graph in the course of this paper. Hence, attributes of rooms are modeled as node attributes and attributes of room connections are modeled as edge attributes. Rooms are created in an abstract, shapeless mode indicated by a circle. Rooms may always be dragged independently from each other and their attributes and connections can iteratively refined by the user, where each aspect can be specified as abstract or specific as desired. As a convention, a single line wall between two rooms indicate a wall connection, double lines represent doors.

In order to be usable as a search interface, the Archistant WebUI possesses a search sidebar, in which the fingerprint weights can be adjusted and the retrieval process can be triggered. Furthermore, the result thumbnails are also shown here as well as the full screen query-result mapping view (showing which room of the query relates to which room of the result) can be invoked. Finally, results can be rated by the user, allowing for machine learning-based optimization in future.

3 Floor Plans Retrieval Techniques

In this Section, we present the main components and underlying concepts for our floor plans retrieval framework that combines three different search methods for

this purpose. The framework is an integral part of the Archistant infrastructure and allows for comprehensive search process with CBR and two (sub)graph matching methods: VF2 (exact matching) and IB (index-based retrieval based on the Neo4j graph database index). The underlying structure for search of similar floor plans is the paradigm of semantic fingerprint that allows for decomposition of the search request into different semantically enhanced sub-patterns, thus giving us opportunity to look for the best fit for the floor plan query based on the features that are important for this particular query only. In Sect. 3.1 we describe the *semantic fingerprint*, i.e., our underlying sub-patterns paradigm, followed by Sect. 3.2 that briefly describes our query structure. In Sect. 3.3 we present our three retrieval methods, including CBR-based MetisCBR and (sub)graph-based VF2 and IB.

3.1 Semantic Fingerprints Concept

Langenhan and Petzold [17] describe semantic fingerprint as a hierarchically constructed index for definition of floor plans that enhances the well-known concept of Building Information Modeling (BIM). To represent the fingerprints, a graph-based structure is developed that can represent the topology of the floor plan and the connections between particular node units (rooms) including only the graph attributes defined for this fingerprint. To transform this graph-based structure into a machine-readable format (XML), the AGraphML specification [16] is used. Furthermore, semantic fingerprint is a representative of room-based configuration, thus rooms and their relations play the most important role in resolving queries that are constructed in the same way. Our searching techniques can detect a number of fingerprints in the query provided by the user: VF2 and IB apply the decomposition of the floor plan query, whereas MetisCBR implements the recognition of patterns based on the fingerprints data contained in the query. In Fig. 4 a list of 7 fingerprint patterns that are common for each of our searching techniques are shown.

3.2 Query Structure

Retrieval queries in Archistant are constructed by utilizing the AGraphML structure: for each room in the floor plan concept, a node is created and the room's properties are used as node properties. Likewise, connections between rooms are represented as edges and the connection's attributes are used as edge attributes. Finally, the resulting AGraphML is wrapped into a search query XML structure along with the user-defined fingerprint weights. In Listing 1.1, a general structure for our queries is shown.








Fingerprint	Name	Description	Specifics
FP1 	Room Count	Number of rooms	No connections between rooms and no labels specified
FP2 	Relation Count	Number of edges	No room information specified
FP3 	Room Graph	Anonymous representation of rooms and edges	No labels specified for rooms and edges
FP4 	Room Types	Labels of rooms	No connections between rooms only room labels are specified
FP5 	Adjacency	Emphasis on room semantics	Rooms information is complete no edge labels specified
FP6 	Accessibility	Emphasis on edge semantics	Edge information is complete no room labels specified
FP7 	Full Graph	Complete graph	All information about rooms and edges available

Fig. 4. Fingerprint patterns currently available in all three (MetisCBR, VF2, IB) retrieval techniques of Archistant (Figure from [1]).

Listing 1.1. General structure of a query for the retrieval methods in our framework (adapted version from [5]).

```
<?xml version="1.0" encoding="UTF-8"?>
<searchrequest>
  <fingerprint name="Room_Types" weight="0.7" />
  <fingerprint name="Adjacency" weight="0.3" />
  <agraphml>
    <graphml>...</graphml>
  </agraphml>
</searchrequest>
```

3.3 Matching Techniques

Case-Based Retrieval (MetisCBR). MetisCBR was developed to apply a multi-agent system with case-based agents to problems of retrieval of similar floor plans during the early phases in architectural conceptual design. Its main features are the retrieval containers that can concurrently resolve different queries

that may belong to the same retrieval process (or be completely independent, i.e., triggered by another retrieval process). Before the actual retrieval takes places, the search request is analyzed, divided in the sub-queries (if multiple semantic fingerprints were detected in the request), and then assigned to the corresponding retrieval container that consists of the agents most suitable for this type of query/fingerprint. This assignment process is governed by a special coordinator agent described in [5]. Figure 5 shows a general overview of the MetisCBR system.

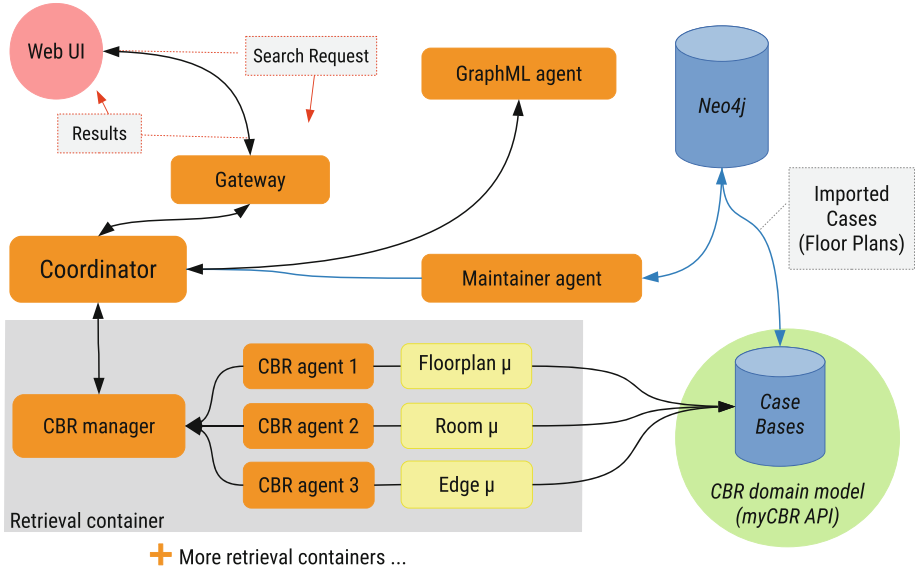


Fig. 5. General architecture of MetisCBR (a detailed description is available in [4,5]).

As a CBR-based system, MetisCBR defines an underlying structure for each case saved in its case base. This structure is mostly based on a domain model. For MetisCBR, a distributed domain model (described in [3]) was created to govern the system’s cases. Each case represents a single floor plan and is divided into three main concepts: FLOORPLAN (meta data about the floor plan), ROOM (information about rooms), and EDGE (information about edges, i.e. room connections). Attributes, such as `roomType` or `windowExist` for rooms, and `edgeType` and `linearDistance` for edges define the detailed structure of a case.

The attributes are combined in different amalgamation functions that either correspond to semantic fingerprints or can be of generic type. It depends on actions of the user (who may or may not include the fingerprint patterns in the query) which amalgamation will be used for the current search. For the amalgamations that are connected to the fingerprints, a combination of attributes is selected for the search that is predefined and unique for this fingerprint only (it

is of course possible that an attribute is available in multiple fingerprints, i.e., an attribute can be used multiple times during the same search process). In [22] a footprint sets based retrieval system is presented that became an inspiration for our fingerprint-amalgamation-based retrieval. The fingerprint amalgamation and the generic amalgamation (that uses all attributes for comparison) can be used in two different types of retrieval strategies:

- A strategy for fingerprints that have a complicated structure (such as FP5 or FP6) and a comprehensive search without fingerprints defined. This strategy is presented in [3].
- Faster strategy that uses more simple fingerprints (such as FP1 or FP4) and applied for simplified search for requests without fingerprints defined.

After the actual search, the results can be elevated by means of applying the user-defined fingerprint weights and sorted in descending order by the computed similarity value.

The current work on MetisCBR is concentrated on further development of retrieval strategies. A study of Ayzenshtadt et al. [26], conducted among architectural domain representatives to investigate their cognitive reasoning processes during the search for similar architectural designs, revealed that a number of commonalities exist among the similarity assessment processes of all of the representatives. The findings of this study helped to infer the definitions for retrieval strategy and superstructural (conceptualization) process. These definitions will be considered foundations for every future strategy of the system (e.g., each strategy should satisfy the requirements from the strategy definition to be accepted for implementation in MetisCBR).

VF2-Based Retrieval (Exact Graph Matching). In graph matching domain, the phenomena of one-to-one mapping is referred as isomorphism. The graphs are isomorphic when they follow the exactly same topology, that is, they both have the same number of nodes and each of the corresponding node is connected in same way. Exact graph matching is a way to detect the isomorphism [7]. Some of the one-to-one exact graph-based matching approaches include: [18, 20, 23]. For Archistant, we decided to use the VF2 algorithm, proposed in [11], its implementation is provided in the `NetworkX` library. As compared to other available implementations, VF2 has the capacity to achieve the best performance for small and sparse graphs [12]. In addition to this, it requires less memory.

Our exact graph matching system (VF2) relies on a preprocessing step. During preprocessing, one AGraphML file is generated for each of the floor plans in the data base. Later on, these AGraphML files are used by the VF2 system. A tool named “Neo4j Shell Tools” is available that is used to generate these AGraphML files.

VF2 system performs different steps in order to compare the search request with the floor plans in the data base (see Fig. 6). Firstly, once the request is received by the system, its validity is ensured. Only the valid requests are forwarded to the next step. In this second step, AGraphML is extracted from

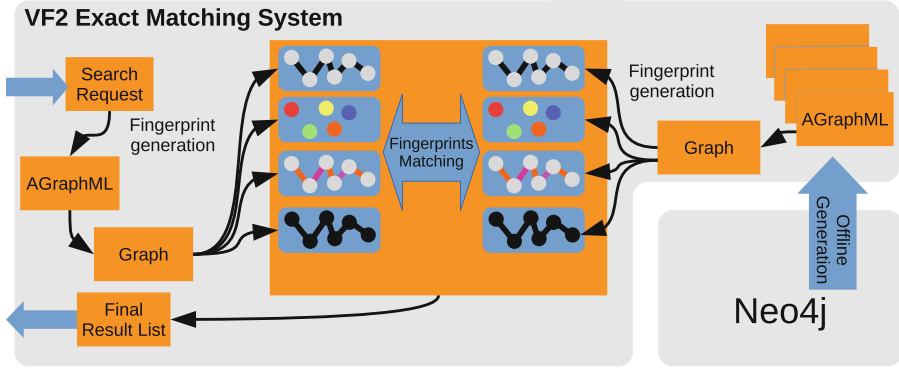


Fig. 6. The above diagram shows the workflow of the VF2 exact matching system. It shows the step by step details of how a search request and the floor plans in data base are decomposed into fingerprints and then their corresponding fingerprints are matched. Finally, the results are transferred to the requester (Figure from [1]).

the search request to generate a graph, that is referred as *query-graph*. The *query-graph* represents nodes and connections between the nodes. Finally, the *query-graph* is decomposed into fingerprints. All the aforementioned steps take place each time, when the user creates a query, before the actual matching part. Once the fingerprints are generated for the *query-graph*, the fingerprints for floor plans in the data base need to be generated. For this purpose VF2 system, one by one takes each of the AGraphML files, referred as *db-graph*, and generates its fingerprints. These fingerprints are then matched with the fingerprints of the *query-graph*. Each of the corresponding fingerprints, that is, FP1 of *query-graph* is matched against FP1 of *db-graph*, FP2 of *query-graph* is matched against FP2 of *db-graph* and so on. Based upon the matching fingerprints, a similarity score is computed, that shows how closely a *db-graph* is similar to the *query-graph* (see Fig. 4). Finally, VF2 system sends back the results with top similarity scores in descending order.

Index-Based Retrieval (Inexact Graph Matching). Several different approaches of index-based graph matching methods have been described in literature, including GraphGrep [13], Lucene index [21], FG-Index [10] and cIndex [9]. Archistant’s index-based retrieval uses Lucene index since this indexing method is used by the Neo4j database by default.

The index-based retrieval can be described as follows (see Fig. 7): A search request AGraphML file is decomposed into the different fingerprints and a set of fingerprint weights. Graph-based fingerprints are represented by internal graph structures. These internal representations are then translated into cypher queries and successively passed to the Neo4j server.

The Neo4j server replies each request with a set of floor plans (more precisely URIs referencing the floor plans are used). These sets are unified discarding all

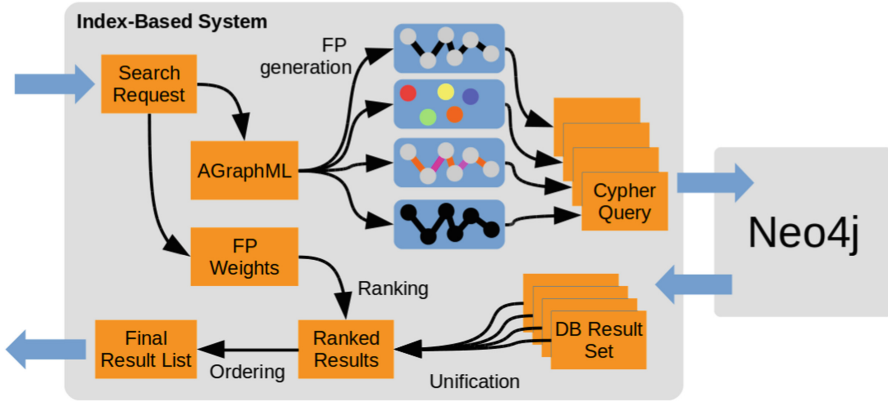


Fig. 7. The above figure shows index-based retrieval flow chart. After the search request is decomposed into a set of internal graph structures representing the different fingerprints, cypher queries for each fingerprint are created. These cypher queries are successively passed to the Neo4j server, and the Neo4j’s replies to each query with a set of floor plan references. These references are unified, taking the user-defined weighting into account (Figure from [1]).

redundant entries. Simultaneously, the index-based similarity score is calculated for every item. This similarity score is the sum of the user-defined weights of the fingerprints for which the query matches the database entry. Finally, the result list is brought into descending order according to the index-based similarity score.

A graph-based fingerprint is considered to match a database entry if the fingerprint’s graph is a subgraph of the database entry. The fingerprints are processed independently from each other for simplicity reasons, hence one room in the query may be mapped to different rooms within the same floor plan in the database. Figure 8 illustrates an example of the fingerprints processing within the index-based method. The query consists of three rooms labeled as Living, Kitchen and Sleeping. The Living room is connected with Kitchen via an edge connection labeled as Passage, the Kitchen is connected with Sleeping via an edge connection labeled as Wall, and Sleeping room is connected with Living via an edge connection labeled as Door. The right side of the diagram shows exemplary matching and non matching fingerprints between search query and floor plan in the database.

3.4 Augmentation of Retrieved Floor Plans

The retrieval systems deliver results in the form of URLs which point to plain image files. These image files serve as thumbnails for the individual results. In order to allow for better user experience, additional information is needed: firstly, detailed information about the results’ graph structures allows for rendering of the results’ floor plans in higher quality. By using the same layout as employed

Fingerprint Matching

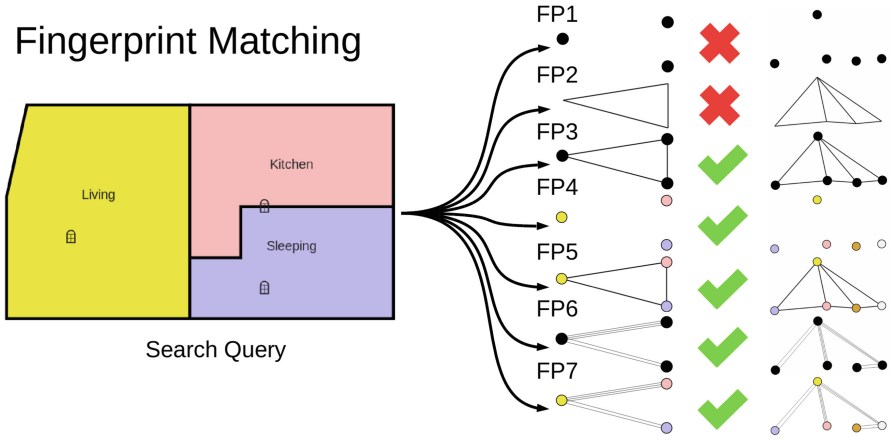


Fig. 8. Decomposition of a floor plan query into fingerprints and subsequent matching with a database entry (Figure from [1]). Depicted is the subgraph matching behavior as implemented in the index-based retrieval.

in the WebUI editor, the user may orient himself more easily in the results. Secondly, a map from individual rooms in the user's query to individual rooms in the server's results helps the user to understand how the results have been derived. These informations are gathered and centralized for all results of all retrieval systems at the augmentation processor (AP, see Fig. 9).

Generation of Result-Related AGraphML Files. Both the generation of AGraphML files related to the result image file URLs and the generation of room maps from query rooms to result rooms are implemented by querying the same Neo4j database on which the retrieval systems are based. As a basic principle, the image URLs used by the retrieval systems are also attached to the graphs in the database. The generation of result AGraphML files is implemented as follows:

1. Result image URL is used to retrieve the id of a so-called storey vertex. These storey vertices are used to organize floor plans. All room-representing nodes of a floor plan are connected to a single storey vertex.
2. The node IDs (along with the relevant node properties like room purpose and room layout polygon) of all room nodes connected to the storey vertex of interest are obtained.
3. All connections between the nodes retrieved earlier are obtained.

Generation of Room Maps. The AP uses Neo4j's matching mechanisms to obtain maps from the user's query to the retrieval system's results. Therefore, the fingerprint abstractions are employed here just like in the retrieval systems.

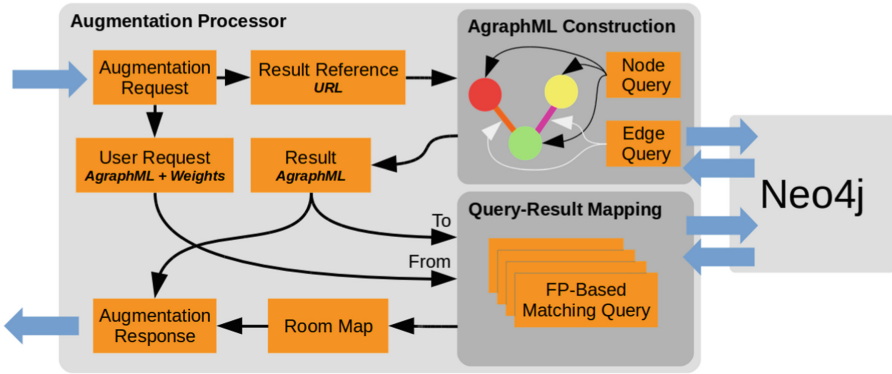


Fig. 9. Structure of the augmentation processor (Figure Adapted from [31]).

Based on the order of the user’s fingerprint weights, different abstractions of result and query are tried to be matched. Since not all retrieval systems use exact matching techniques (and not all fingerprints are selected as mandatory by the user), there might be results to which no fingerprints of the query match the result’s fingerprints at all. The first matching fingerprint (where the order is determined by the user) is used for the generation of the final room map. There are situations, in which the abstractions of result and query match in more than one way (e.g. in FP1, any room may match). In such cases, one of the matches with the highest number of matching room purposes is selected (if there are multiple of them, randomly). A visualization of the room map can be displayed to the user (see Fig. 10).

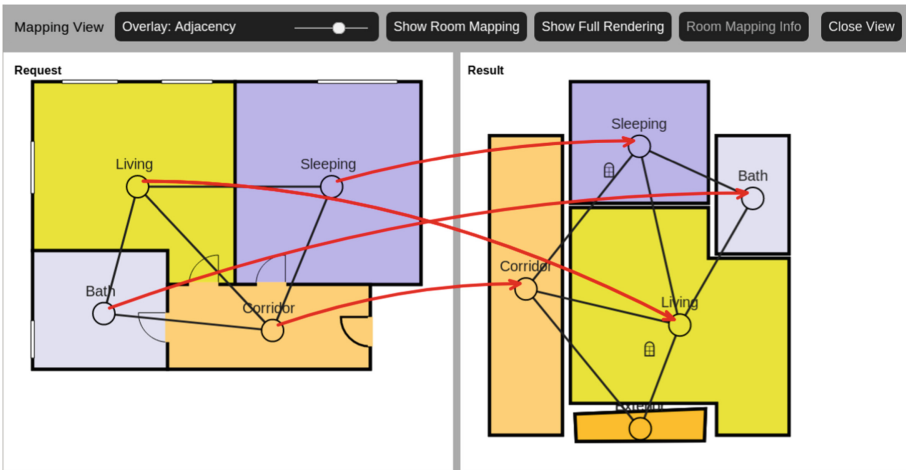


Fig. 10. Screenshot of the room mapping view in the Archistant WebUI.

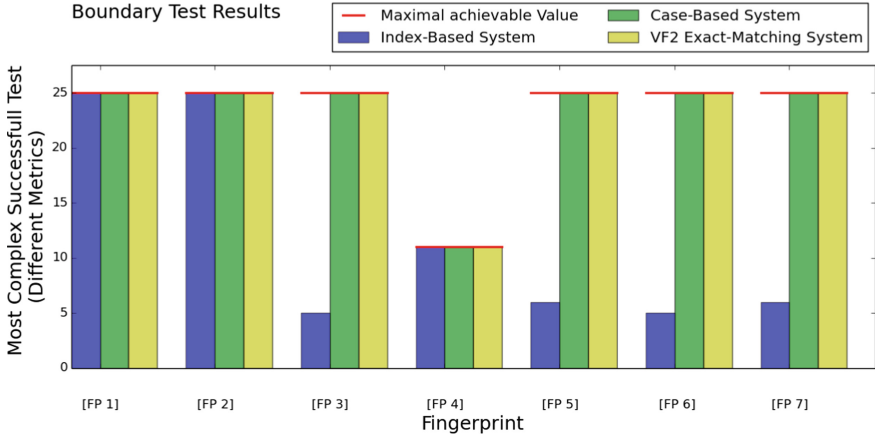


Fig. 11. Overview of the boundary test results. For each fingerprint and retrieval system the boundary is depicted (the metrics differ between different fingerprints). For each fingerprint, the maximum achievable value is given (Figure from [1]).

4 Evaluation of Our System

4.1 Computational Limitations (Boundary Test)

All the presented algorithms are expected to terminate properly for any given search query in theory. However, since both graph matching and CBR are computational demanding, there are practical limits (boundaries) to the complexity of a search query our system can handle. In order to determine these boundaries, we conducted an automated stress-test in which for every fingerprint we run a series of test cases and record the behavior of the retrieval systems. In each series, test cases of increasing complexity are used. In most cases, a test scenario is considered of complexity n , if it consists of n rooms. For FP2 however, n connections are used instead. For graph-based fingerprints, we use linear graphs. Based on the type of fingerprint, we used different node and edge attributes, that are randomly selected for each test case. A boundary of a retrieval system for a certain fingerprint is considered to be the complexity rating of the lowest test case if the system was unable to process without crash minus 1. The results of the boundary test we conducted are depicted in Fig. 11. Both the VF2-based retrieval and the case-based retrieval managed to process all test cases without crash. Only the index-based retrieval system exhibited limitations over the maximum size of fingerprints FP3, FP5, FP6 and FP7. It is assumed that these limitations arose from internal timeout errors of the underlying Neo4j database. Generally, given more memory and computational power, these boundaries could be raised. Using the determined boundaries of the different retrieval systems, the productive use of Archistant can be secured by restricting the system to queries that are known to be manageable.

4.2 Qualitative Analysis

In order to assess the usefulness of the generated results for the architects, the quality of results is subjectively estimated for a set of dedicatedly generated sample queries. With the help of architects, we created 10 different search queries. One by one, each of the queries was entered to the Archistant’s retrieval framework. Archistant performed the same fingerprint matching for each of the queries and the results were observed and stored. In order to make the fair comparison, the same architects, who designed search queries with us, also took part in the qualitative analysis. All the participants rated the three retrieval methods on scale of 1st, 2nd, and 3rd or equal. Table 1 shows the results of this qualitative rating study. The table contains the summary of the results and the ratings for each method. To make it more elaborative, we show the results in two categories. A method is regarded as best or clear winner when all the participants ranked it best. The first category shows the queries which were ranked as best for the corresponding retrieval method. The second category shows the results of methods that were considered as best by majority of the participants for a particular query. The third column shows the queries that got equal number of votes. The last column shows the percentage of the dominating queries.

It is clear from Table 1, that none of the retrieval methods failed completely, rather they were able to produce quality results of some of the queries. Randomly, we selected three queries, their two best results for each of the retrieval methods are shown in Fig. 15. For ease of understanding we show the results in graph-based representation and graphical representation shown by top and bottom representation in Table 1 respectively. It is worth noting that the CBR method has a higher score in relation to other two methods.

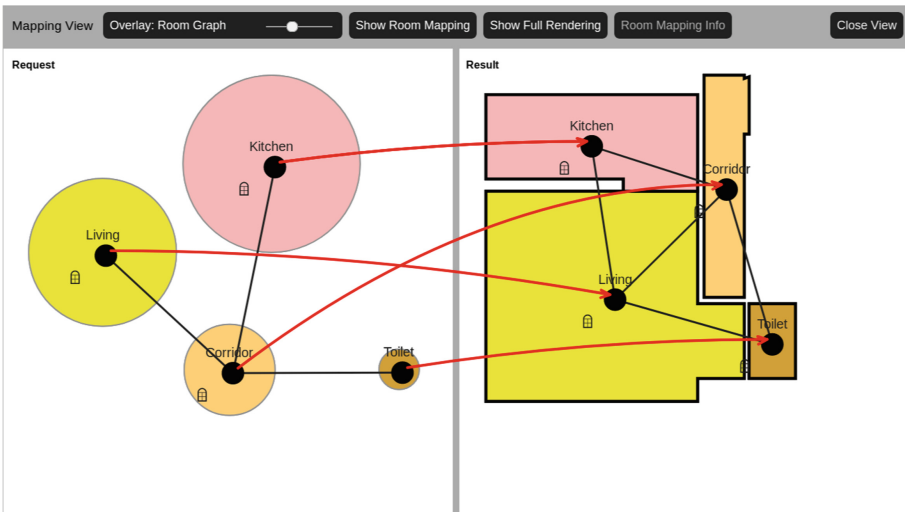


Fig. 12. Mapping between query and result 1 (room graph fingerprint overlay).

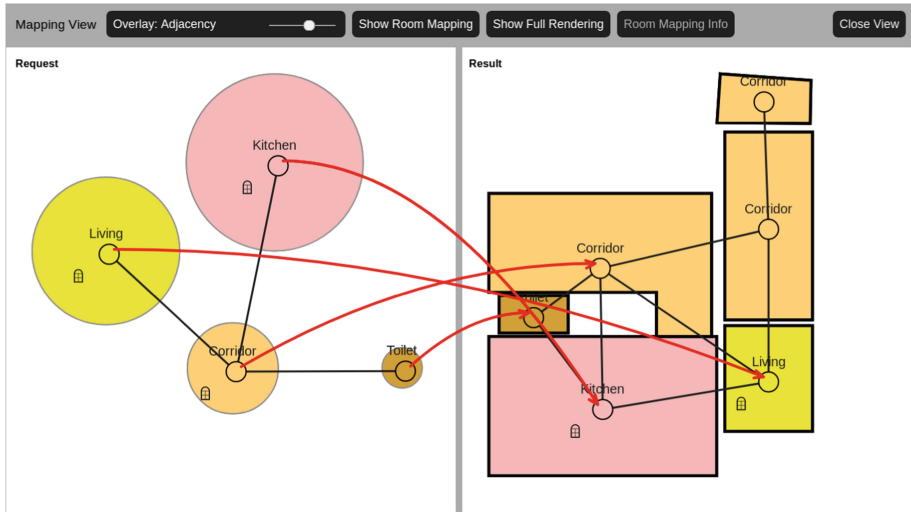


Fig. 13. Mapping between query and result 1 (adjacency fingerprint overlay).

In general, we can notice that the retrieval methods were able to find and present sets of reasonable results. Subjectively, the VF2 method outperformed other techniques overall, confirming the assumption that the exact isomorphism can be seen as the most suitable method for matching in databases with certain structural and technical constraints.

4.3 Query-Result Mapping Case Study

In order to demonstrate the usefulness of the query-result mapping functionality, we applied this algorithm to a query selected from the qualitative study described above.

Given was a sketch with a living room, a kitchen, a toilet, and a corridor, where all rooms are connected to the corridor by passage connections. The retrieval system returned several results, from which 3 were investigated. In Fig. 12, the query is matched to a floor plan that has exactly the same amount of rooms, all room functions in the query are matched to rooms with same functions. However, the architect might at least get inspiration for a room layout. In the second retrieval result (see Fig. 13), the queried structure is mapped to a larger one, that could inspire the architect to make some additions to his concept. Finally, in the third result (Fig. 14), a graph structure is found, that is also extended compared to the query. When switching to the full room graph overlay, it became obvious, that all room functions could be matched, not all connections could. However, the founded mapping is suggesting new connection types. This could hint the user that a doorless connection between a corridor and a toilet may be improved.

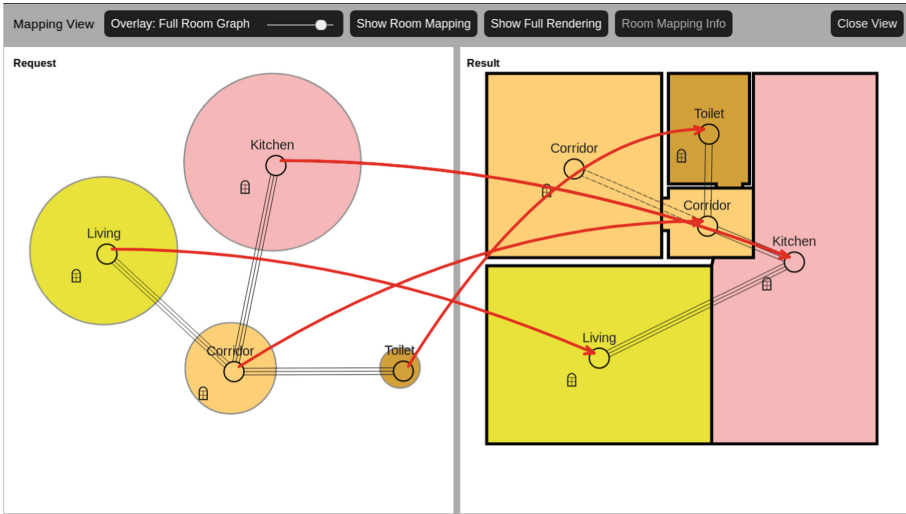


Fig. 14. Mapping between query and result 1 (full room graph fingerprint overlay).

		CBR		VF2		IB	
		1st Best	2nd Best	1st Best	2nd Best	1st Best	2nd Best
		Q4 Building Vertex Storey vertex Living Corridor Kitchen Bath Room Sleeping	0.837	0.837	0.42	0.42	0.28
Q5 Building Vertex Storey vertex Kitchen Bath Room	0.837	0.837	0.57	0.57	0.571	0.571	
Q9 Building Vertex Storey vertex Living Bath Room Sleeping Corridor	0.88	0.84	0.42	0.42	0.57	0.43	

Fig. 15. The above figure shows the similarity scores of selected queries. Color codes represent the room purposes, the first column contains the queries with rooms and assigned purposes. The two best results of a query against each retrieval method are shown. Each box shows the similarity score, the corresponding graph, and a graphical floor plan representation. The colored boxes show the best results (Figure from [1]). (Color figure online)

Table 1. The above table shows the results of selected queries for each of the retrieval method. The second column shows the queries whose results were ranked as best by all the participants against the corresponding retrieval method in the first column. The third column shows the queries which won the support of majority of participants. Queries that get equal number of votes are placed in fourth column. The last column contains the percentage of queries dominated by the particular method (Table from [1].)

Retrieval method	Queries won	Queries won by majority	Co-winner in	Summarized results
VF2	Q3, Q4	Q6, Q7	Q8, Q10	50%
Index-Based	–	Q5	Q10	15%
MetisCBR	Q2, Q9	Q1	Q8	35%

5 Conclusion and Future Work

In this work, we presented a novel possibility for architects to enhance the early conceptual design phase by using an end-to-end system Archistant that is able to search for similar floor plans during this phase of the design process. Archistant uses a sketch-based interface for construction of floor plan queries and distributes this query, with the help of a processing component, among three different retrieval methods that are based on different research paradigms of artificial intelligence, namely, case-based reasoning and graph matching. The retrieval results are enhanced by an augmentation processor that is able to visualize room mapping between the query and the corresponding result, thus providing a justification of how both room configurations match. We evaluated the complete system with a boundary test to determine the retrieval-related limitations of our three searching techniques, where most of the methods were able to deal with the highest complexity of a query. We also conducted a qualitative analysis where each of the retrieval methods was able to satisfy the expectations on the delivered results in at least some of the cases. Our future work will concentrate on building of a bigger collection of retrievable floor plans, and an inclusion of machine learning for automatic improvement of results.

Acknowledgement. *MetisCBR* [4] and the *Metis WebUI* have been implemented in the course of the research project *Metis – Knowledge-based search and query methods for the development of semantic information models (BIM) for use in early design phases*. *Metis* is an interdisciplinary project, funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

References

1. 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: De Marsico, M., di Baja, G.S., Fred, A.L.N. (eds.) *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, (ICPRAM 2017)*, 24–26 February, Porto, Portugal, pp. 50–60. SCITEPRESS (2017). ISBN 978-989-758-222-6
2. Ahmed, S., Weber, M., Liwicki, M., Langenhan, C., Dengel, A., Petzold, F.: Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recogn. Lett.* **35**, 91–100 (2014)
3. 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)*, located at SGAI International Conference on Artificial Intelligence, 15–17 December, Cambridge, United Kingdom. School of Computing, Engineering and Mathematics, University of Brighton, UK (2015)
4. 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.) *Proceedings of the 8th International Conference on Agents and Artificial Intelligence, (ICAART 2016)*, 24–26 February Rome, Italy. SCITEPRESS (2016)
5. Ayzenshtadt, V., Langenhan, C., Roith, J., Bukhari, S., Althoff, K.-D., Petzold, F., Dengel, A.: 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
6. Bayer, J., Bukhari, S.S., Langenhan, C., Liwicki, M., Althoff, K.-D., Petzold, F., Dengel, A.: 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
7. Bengoetxea, E.: Inexact graph matching using estimation of distribution algorithms. *Ecole Nationale Supérieure des Télécommunications, Paris* **2**(4) (2002)
8. Cavieres, A., Bhatia, U., Joshi, P., Zhao, F., Ram, A.: CBArch: a case-based reasoning framework for conceptual design of commercial buildings. In: *Artificial Intelligence and Sustainable Design - Papers from the AAAI 2011 Spring Symposium (SS-11-02)*, pp. 19–25 (2011)
9. Chen, C., Yan, X., Yu, P.S., Han, J., Zhang, D.-Q., Gu, X.: Towards graph containment search and indexing. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 926–937. VLDB Endowment (2007)
10. Cheng, J., Ke, Y., Ng, W., Lu, A.: FG-Index: towards verification-free query processing on graph databases. In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pp. 857–872. ACM (2007)
11. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(10), 1367–1372 (2004)

12. Foggia, P., Sansone, C., Vento, M.: A performance comparison of five algorithms for graph isomorphism. In: Proceedings of the 3rd IAPR TC 2015 Workshop on Graph-based Representations in Pattern Recognition, pp. 188–199 (2001)
13. Giugno, R., Shasha, D.: GraphGrep: a fast and universal method for querying graphs. In: Proceedings of the 16th International Conference on Pattern Recognition, vol. 2, pp. 112–115. IEEE (2002)
14. Heylighen, A., Neuckermans, H.: A case base of case-based design tools for architecture. *Comput.-Aided Des.* **33**(14), 1111–1122 (2001)
15. 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)
16. Langenhan, C.: A federated information system for the support of topological bim-based approaches. In: Forum Bauinformatik Aachen (2015)
17. 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.: Archit. Mod. Inf. Tech.* **4**, 13 (2010)
18. McKay, B.D., et al.: Practical graph isomorphism. Department of Computer Science, Vanderbilt University Tennessee, US (1981)
19. Messmer, B.T., Bunke, H.: A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recogn.* **32**(12), 1979–1998 (1999)
20. Schmidt, D.C., Druffel, L.E.: A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM (JACM)* **23**(3), 433–445 (1976)
21. Sharanya Jayaraman, S.V.: Comparative survey of query processing on graph databases. Project report, Florida State University (2013)
22. Smyt, B., McKenna, E.: Footprint-based retrieval. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS, vol. 1650, pp. 343–357. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48508-2_25
23. Ullmann, J.R.: An algorithm for subgraph isomorphism. *J. ACM (JACM)* **23**(1), 31–42 (1976)
24. Voss, A.: Case design specialists in FABEL. In: Issues and Applications of Case-Based Reasoning in Design, pp. 301–335 (1997)
25. Wessel, R., Blümel, I., Klein, R.: The room connectivity graph: shape retrieval in the architectural domain (2008)
26. 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
27. Heylighen, A., Schreurs, J., Neuckermans, H.: ENTER instead of SUBMIT. In: DesignNet-Knowledge e Information Management per il design, pp. 171–182 (2002)
28. Richter, K., Heylighen, A., Donath, D.: Looking back to the future—an updated case base of case-based design tools for architecture. *Knowl. Model.-eCAADe* **25**, 285–292 (2007)
29. Richter, K.: Augmenting designers’ memory: case based reasoning in der Architektur. Logos-Verlag (2011). ISBN 9783832527334
30. Schank, R.C.: Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press, Cambridge (1983)
31. Bayer, J.: Development of a modular software framework for supporting architects during early design phases. Master’s thesis, University of Kaiserslautern (2017)