

# Semantic Pattern-based Retrieval of Architectural Floor Plans with Case-based and Graph-based Searching Techniques and their Evaluation and Visualization

Qamer Uddin Sabri<sup>1,2,\*</sup>, Johannes Bayer<sup>1,2,\*</sup>, Viktor Ayzenshtadt<sup>1,3,\*</sup>,  
Syed Saqib Bukhari<sup>1</sup>, Klaus-Dieter Althoff<sup>1,3</sup> and Andreas Dengel<sup>1,2</sup>

<sup>1</sup>German Research Center for Artificial Intelligence, Trippstadter Strasse 122, 67663 Kaiserslautern, Germany

<sup>2</sup>Technical University Kaiserslautern, P.O. Box 3049, 67663 Kaiserslautern, Germany

<sup>3</sup>University of Hildesheim, Institute of Computer Science, Samelsonplatz 1, 31141 Hildesheim, Germany  
{qamer.uddin.sabri, johannes.bayer, viktor.ayzenshtadt, saqib.bukhari, klaus-dieter.althoff, andreas.dengel}@dfki.de

**Keywords:** Graph Matching, Subgraph Matching, Graph Isomorphism, Architectural Floor Plan, Case-based Reasoning, Pattern Recognition.

**Abstract:** Until today, for the conceptual design of architectural floor plans, architects widely follow the traditional pen and paper based method to draw the conceptual floor plans, and retrieve the similar floor plans in the printed reference collections. In this paper we present a complete end-to-end system that helps architects to retrieve similar floor plans in early design phases. This work makes a three-fold contribution. Firstly, we have adapted three state of the art techniques to retrieve the similar floor plans: case-based reasoning (CBR), exact graph matching, and inexact graph matching. Secondly, we conducted a test to detect the computational limits of the searching techniques. And finally, we performed a qualitative analysis by running more realistic test cases created by architects while keeping in mind the computational limits. For visualization of results, we have integrated advanced version of our previously implemented web-based user interface. The qualitative analysis showed that the exact graph matching gives in general better results for a majority of test cases, as compared to other two methods. The novelty of our approach is that it combines CBR, exact, and inexact graph matching in one system in the domain of retrieval of architectural floor plans.

## 1 INTRODUCTION

When starting the design of a building, architects have to develop floor plan concepts facing usually only vague description of the building's requirements. In order to get inspiration to solve the creative problems, working with references is an established method in architecture. Traditionally, this leads to a labor-intensive search and review of magazines and books to find ideas similar to an initial concept. In order to speed up this process, a dedicated search automation would be needed.

For this purpose, we have already presented a distributed case-based retrieval approach *MetisCBR* (Ayzenshtadt et al., 2016a) for search of similar architectural building designs which is prototypically implemented as part of the infrastructure of a basic research project *Metis – Knowledge-based search and query methods for the development of semantic infor-*

*mation models (BIM) for use in early design phases.* *Metis* is an interdisciplinary project, funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

*MetisCBR* can be accessed by a dedicated user interface called *WebUI* (see Figure 1). This web-based GUI allows for creating architectural concepts (floor plans) and sending them to the retrieval engines. The general usability of the *WebUI* has been shown by the means of a user study (Bayer et al., 2015).

However, so far other types of well-known search methods, like graph and subgraph matching-based retrieval, are not fully tested on this problem, as well as not compared directly with CBR-based approaches.

In this paper we present a complete end-to-end architectural design support system called *Archis-tant* (<http://www.dfki.uni-kl.de/archistant>), that implements the previously mentioned *WebUI* that lets the user draw the required floor plan with rooms and connection between these rooms, adjust the search

\*Equal contribution to the paper.

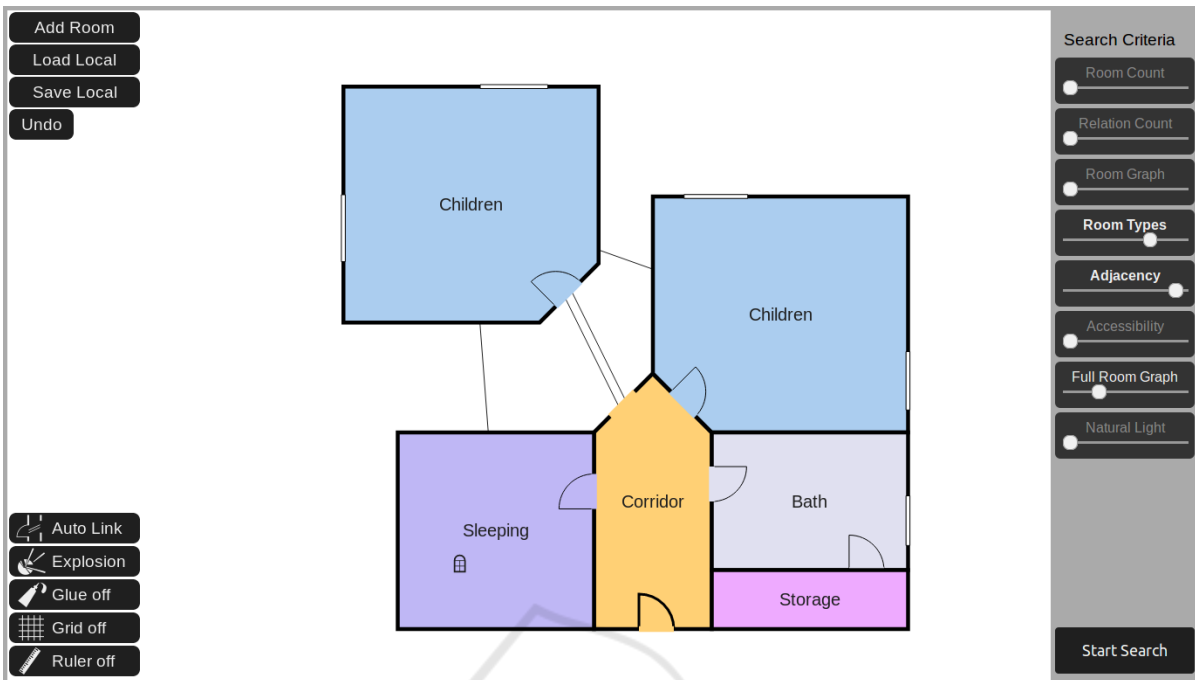


Figure 1: Screenshot of the Archistant WebUI. As a room oriented tool, the individual rooms may be dragged independently from each other, revealing their editable connections to each other. Single lines represent wall connections, double lines represent doors.

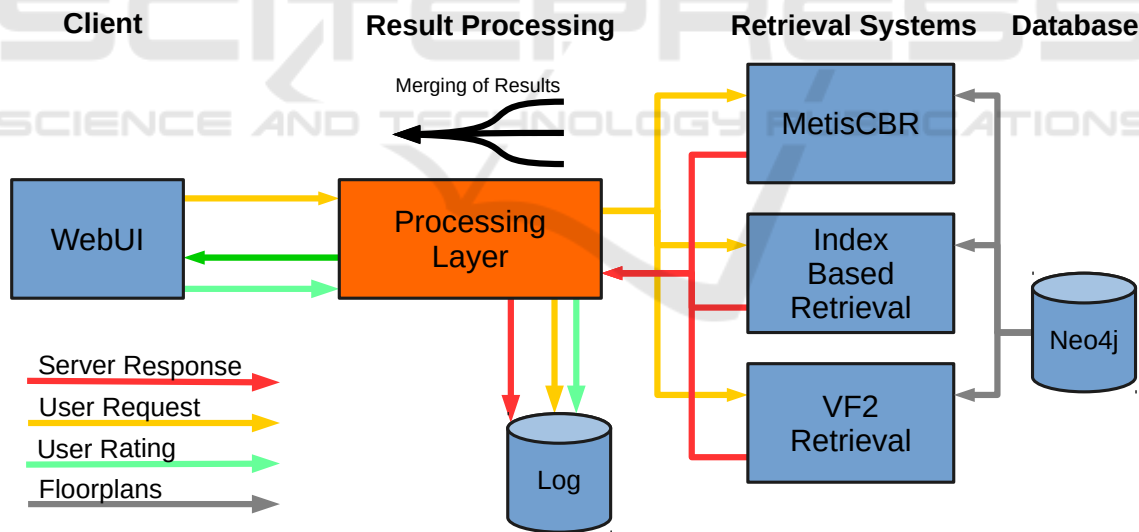


Figure 2: Overview over the system architecture of Archistant (simplified).

criteria, search the similar floor plans, and then visualize the mapping between user's search query (floor plan) and resulted floor plans. For retrieval of similar floor plans, we have implemented three searching techniques, namely case-based retrieval (previously mentioned MetisCBR), exact graph matching, and inexact graph matching. The processing pipeline of Archistant is divided in three steps. First, the user

draws the search query with the WebUI. Afterwards the query is forwarded to the search engines, each of them extracts the semantic search patterns (*fingerprints*) (see also Figures 2 and 3) and matches these patterns with floor plans in the database. The result sets of the systems are then unified and sent back to the WebUI for visualization and room mapping.

We also conducted a boundary test of each re-

trieval method with respect to each search pattern to analyze their technical limitations. Keeping in mind these technical limitations, we performed a qualitative analysis with some pre-defined search requests to determine which retrieval method is suitable for which user scenario.

## 2 RELATED WORK

This section provides an overview of work that is related to our paper. Related work can be divided into three categories: CBR, graph and subgraph matching, and sketch-based interfaces.

### 2.1 Case-based Reasoning

As mentioned in Section 1, MetisCBR uses methods of CBR-based retrieval to find similar floor plans. A comprehensive collections of work related to Metis-CBR is described in detail in (Heylighen and Neuckermans, 2001) and (Richter et al., 2007).

### 2.2 Graph and Subgraph Matching

Graph matching is being widely used these days for the retrieval problems. It might be the case that there is no exact match for the whole graph. To tackle this situation another promising feature is provided that is known as subgraph matching, that is, if two graphs are not completely isomorphic but their part(s) are isomorphic then they can be detected with subgraph matching. An implementation that uses graph matching to retrieve the similar floor plans is described in (Ahmed et al., 2014). This work uses a modified version of (Messmer and Bunke, 1999) for retrieval of similar floor plans by arranging the row-column vectors of the adjacency matrix in the decision tree. A related work in (Wessel et al., 2008) implements graph and subgraph isomorphism to check the similarity between a query graph and building models in the database, considering some constraints, that is, if a query graph corresponds to the constraints, only then it would be checked for similarity.

### 2.3 Sketch-based Interfaces

In the fields of engineering and architecture, different implementations of the sketch-based interfaces exist that let the user convey the idea by drawing it. A web-based user interface WebUI has been created as part of the previously mentioned research project Metis. It lets the user draw the architectural floor plans electronically, set the different search criteria and, once

the results are retrieved, the user can interpret and visualize them with mapping between the query and results (i.e. which part of the query relates to which part of the result). For query construction, WebUI uses the AGraphML (Langenhan, 2015) specification.

## 3 FLOOR PLANS RETRIEVAL TECHNIQUES

To accomplish the task of retrieval of building designs with similar context, we developed a unified retrieval framework (as part of Archistant) with currently three methods implemented. Besides MetisCBR, we have added the integration of two graph-based retrieval techniques: the exact matching method with the VF2 algorithm, and the inexact matching method with the underlying index-based data structure. The main concern we had during the development of the framework was the matter of fact that in real-world scenarios the data is not always consistent, i.e., in some cases the numbers of nodes and/or edges are not equal in the two graphs. This does not allow for determination of the exact isomorphism between these graphs. To deal with such cases, our algorithms should be able to tolerate these errors and instead of finding the only best solution, they should try to find a set of appropriate solutions. This knowledge led us to the implementation of the semantic fingerprint patterns concept in all of the three retrieval methods.

This section is further organized as follows: Section 3.1 defines the *semantic fingerprint*, i.e., the search patterns concept, Section 3.2 defines the graph structure, and then the descriptions of the three matching techniques using these fingerprints are presented in Section 3.3.

### 3.1 Semantic Fingerprints Concept

The concept of semantic fingerprint of architecture (Langenhan and Petzold, 2010) is based on an index-based hierarchical structure of building data. The hierarchical nature of semantic fingerprints allows for abstract representation of building topology and relational connections between its parts. A fingerprint pattern is a predefined prototype of such a structure and is represented as a graph (see Figure 3) with options to have node labels (names of rooms) and edge labels (names of edges/connections between rooms).

By using the idea of semantic fingerprint, our retrieval methods are able to either decompose (graph and subgraph matching based techniques) the user query into search patterns (fingerprints) or to derive

the fingerprints from the query (MetisCBR), and then match them with other floor plans in our database.

AGraphML is the XML-based representation for floor plans. The specification (Langenhan, 2015) of AGraphML defines attributes for this representation. It can represent a complete architecture for a floor plan, including the nodes and edges with their attributes.

Figure 3 shows a list of 7 fingerprints currently implemented in each of our retrieval engines.

## 3.2 Query Structure

Search queries and floor plans in our database are represented as graph  $G = (V, E)$ . Here,  $V$  represents the vertices (nodes) in the graphs, which are rooms in our case, e.g., Kitchen or Sleeping.  $E$  represents the edges (the connections between nodes), e.g., Kitchen and Sleeping are connected with an edge called Door.

## 3.3 Matching Techniques

This section presents the three matching and retrieval techniques implemented in Archistant: MetisCBR, exact graph matching (referred as *VF2*), and inexact graph matching (referred as *index-based*).

### 3.3.1 MetisCBR

MetisCBR is based on a decentralized structure where agents and agent groups (retrieval containers) work on different, predefined and assigned, tasks to support an architect during the early design phases. The actual retrieval process is coordinated by a special agent group (*MetisCBR Coordinator*) described in (Ayzenshtadt et al., 2016b). Being a case-based retrieval system, MetisCBR's theoretical emphasis is built on the basic CBR paradigm that similar problems have similar solutions. The system analyses the given problem (user's search request) and tries to find solutions (search results) that can be the most helpful to solve this problem, that is, adapt the currently created design according to findings in the search result. MetisCBR is related to other systems and approaches that were developed for the purpose of assisting an architect, some examples are FABEL (Voss, 1997), CBArch (Cavieres et al., 2011), or CaseBook (Inanc, 2000). To define a case (architectural design graph) within the system, a specific distributed model (see Figure 4) is applied to each graph-based floor plan. The floor plans are then imported into the case base of the system, where they are separated into three main concepts: floor plan meta data, rooms and edges. Each of the concepts consists of a number of specific

attributes (see also (Ayzenshtadt et al., 2015)). During the search, the weighting of single attributes and an amalgamation of them is applied to find the most similar concepts for the given search request (or its parts).

The amalgamations are divided into fingerprint-related and generic. The fingerprint-related amalgamations are applied only if one or more semantic fingerprint patterns were added to the query. Each of the fingerprint amalgamations considers only attributes that were defined for this particular pattern. This process is related to the footprint sets-based retrieval described in (Smyth and McKenna, 1999). In contrast to the fingerprint-related, the generic amalgamation functions use all of the attributes, and are applied if no fingerprint definitions were found. Both of the amalgamation types can be executed within different retrieval strategies. Currently, two kinds of strategies are implemented in the system: a multistep results preselection-based strategy for complex fingerprint patterns as well as for the deep search without fingerprints (described in (Ayzenshtadt et al., 2015)), and a single-step strategy for general retrieval with or without fingerprints. A post-retrieval weighting process (available only for search requests with multiple fingerprints) can be applied to boost results for fingerprints that were considered more important by the user. The results of the retrieval are returned sorted in descending order by the computed confidence score.

### 3.3.2 Exact Graph Matching Method (VF2)

In exact graph matching, one-to-one mapping is known as isomorphism. When two graphs contain the same number of nodes, and they are connected in the same way, then we can call them isomorphic. Isomorphism can be found with exact graph matching (Bengoetxea, 2002). For example, (Ullmann, 1976), (Schmidt and Druffel, 1976), and (McKay et al., 1981) are one-to-one exact graph-based matching approaches. Our system (Archistant) uses implementation of the VF2 algorithm, proposed in (Cordella et al., 2004) (and implemented in the NetworkX library), because it significantly reduces the memory requirement and obtains the best performance for graphs of small size and for quite sparse graphs (Foggia et al., 2001).

In our system, VF2-based method uses AGraphML files to generate the graphs. Firstly, AGraphML files from Neo4j database are generated, as an offline step, with a tool named "Neo4j Shell Tools". One AGraphML file will be generated against each floor plan. Now our system can match the search request with other floor plans of Neo4j database.

The step by step details of how VF2 system








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

Figure 3: Fingerprints (i.e., search patterns) allow for abstract representation of building topology and relational connections between its parts. Fingerprints currently implemented in all examined retrieval methods are shown here. Nodes represent the rooms, edges represent the room connections.

matches (see Figure 5) search request with floor plans is described as follows: in the first step, the system receives the search request, then it checks for its validity, the system proceeds if the request is valid. Then, from the search request, AGraphML is extracted to generate a graph, referred as *query-graph*, and subsequently its fingerprints are generated. After having fingerprints for query-graph, VF2 system then one by one takes each of the offline generated AGraphML files, generates its graph, referred as *db-graph*, and its fingerprints, and then matches the corresponding fingerprints, i.e., FP1 of the query-graph will be matched against FP1 of db-graph of each AGraphML, FP2 of query-graph with FP2 of db-graph of each AGraphML and so on. Once the system is done with matching fingerprints, it delivers the resulting floor plans with confidence score in descending order, the confidence score shows how closely a floor plan matches with the user’s search request according

to each of the fingerprints in Figure 3.

### 3.3.3 Inexact Graph Matching Method (Index-based)

There are different types of inexact, index-based graph searching methods: GraphGrep (Giugno and Shasha, 2002), Lucene index (Sharanya Jayaraman, 2013), FG-Index (Cheng et al., 2007), cIndex (Chen et al., 2007). In this paper, we are using the Lucene index-based method as it comes with the Neo4j framework by default.

The index-based method operates as follows (see Figure 6): A search request is decomposed into the different fingerprints. The fingerprints are represented as *Cypher* (Neo4j query language) queries and the ones whose weight is different from 0 are transmitted to the Neo4j server. The Neo4j server answers the requests with a set of floor plans for each finger-



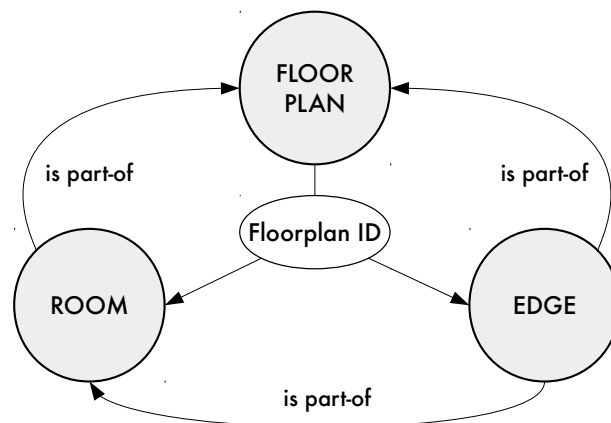


Figure 4: The general description of the underlying distributed model for case-based retrieval in MetisCBR.

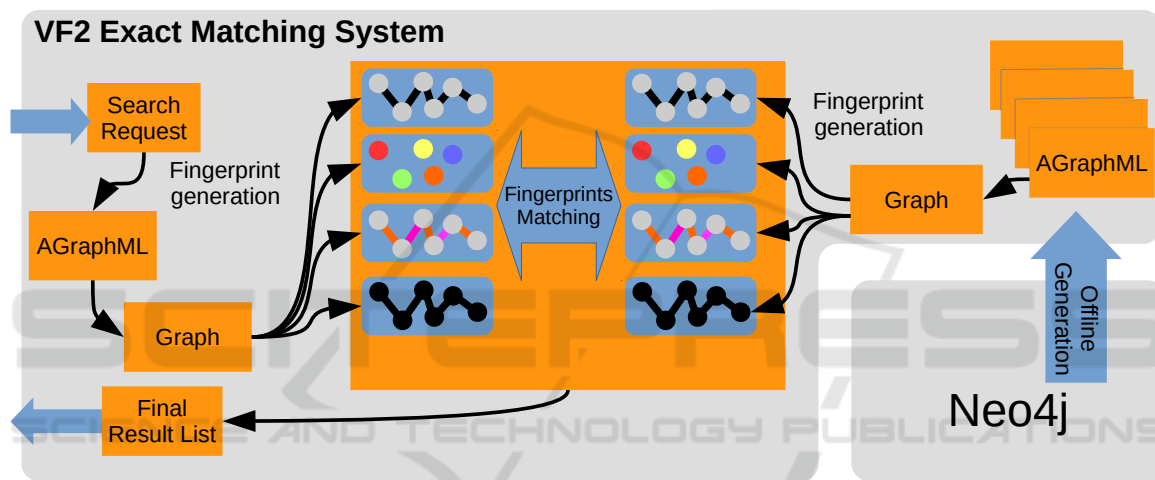


Figure 5: Above is the flow diagram of the VF2 exact matching system. It shows that from the search request first the AGraphML, then the graph and finally the fingerprints are generated. In the next step, the search request's fingerprints are matched with the corresponding fingerprints of the offline generated AGraphML files. Finally, the results are transferred to the requester.

print. These sets are unified and the floor plan results are ordered according to the similarity score which is calculated for every item in the final result set. This similarity score is the sum of the user-defined weights of the fingerprints for which the query matches the database entry.

A fingerprint is considered to match if the fingerprints graph is the subgraph of the database entry. The fingerprints are processed independently for simplicity reasons, hence one room in the query may be mapped to different rooms within the same floor plan in the database. Figure 7 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.

## 4 EVALUATIONS OF OUR SYSTEM

### 4.1 Technical Limitations (Boundary Test)

Because of the computational demands of matching and other problems, all retrieval methods have their

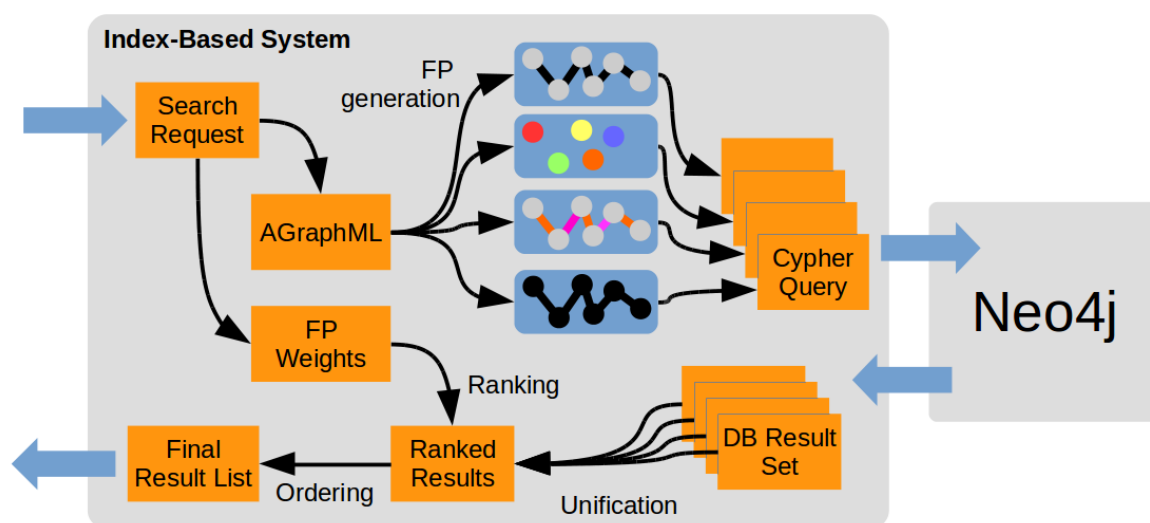


Figure 6: The index-based method flow diagram. A search request is decomposed into AGraphML and fingerprint weights. From the AGraphML the fingerprints are created and the fingerprints are translated into Cypher queries. The Cypher queries are sent to the database, each Cypher-translated fingerprint is answered with a set of floor plans. The result sets are unified. Each floor plan is scored by the sum of the weights of the fingerprints corresponding to the the result sets that contain the floor plan. Finally, the result set is sorted by the score and transferred to the requester.

own limitations regarding the complexity of search requests they can handle, especially with respect to each of the fingerprints shown in Figure 3. Given enough time and storage, all search queries should be answerable in theory, but since computation time must be limited due to usability reasons and storage is limited due to economical reasons, the question arises how complex the search requests may be so that they can be reasonably processed by a given system. In order to test our retrieval methods, we employed the following method: for each type of fingerprint we generated test scenarios of variable complexity which we instantiated over an interval between 1 and 25 units of complexity, where 25 is selected empirically as a maximum limit. The metrics for these units differ for the different fingerprints, but are either equal to the number of rooms in a test scenario or the number of edges in a test scenario. The test scenarios are designed so that the systems are advised to only take the fingerprint into account the scenario was designed for. The most complex test scenario that could be handled by the system is considered to be the system's boundary for that fingerprint. The results are shown in Figure 8 that contains the boundaries of the fingerprints we tested on our systems. Both the VF2 exact matching and MetisCBR executed all test scenarios flawlessly, only the index-based system has limitations over the maximum size of fingerprints FP3, FP5, FP6 and FP7. These limitations came from internal timeout errors of the underlying Neo4j database, most likely due to a congestion of the database system. Knowing the

boundaries of our different methods for a given hardware and time limitation, we defined test scenarios for a subsequent, qualitative analysis (see Section 4.2) in a more safe manner.

## 4.2 Qualitative Analysis

The evaluation of quality and helpfulness of the retrieval results of the above described retrieval methods was the main task of the subsequent special study, where the mixed team of architects and computer scientists created queries and rated collaboratively a number of graphical and graph-based result representations of the search result sets. The confidence scores (range  $R = 0.0..1.0$ ), computed by the retrieval methods during the retrieval, were taken into account as well.

As search queries, 10 AGraphML requests with floor plan constructions of different complexity, were used. Each of the presented retrieval methods was queried with each of those AGraphML-based floor plans and delivered accumulated result sets, sorted in descending order by confidence score of the single result. All of the fingerprint patterns were applied to the given queries. For this qualitative analysis we asked different participants to observe the results of each of the three retrieval methods, corresponding to each of the 10 queries. Each participant rated the three methods on the scale of first, second and third or equal. In Table 1 the results of the subjective qualitative analysis from the participants of the study are shown. This

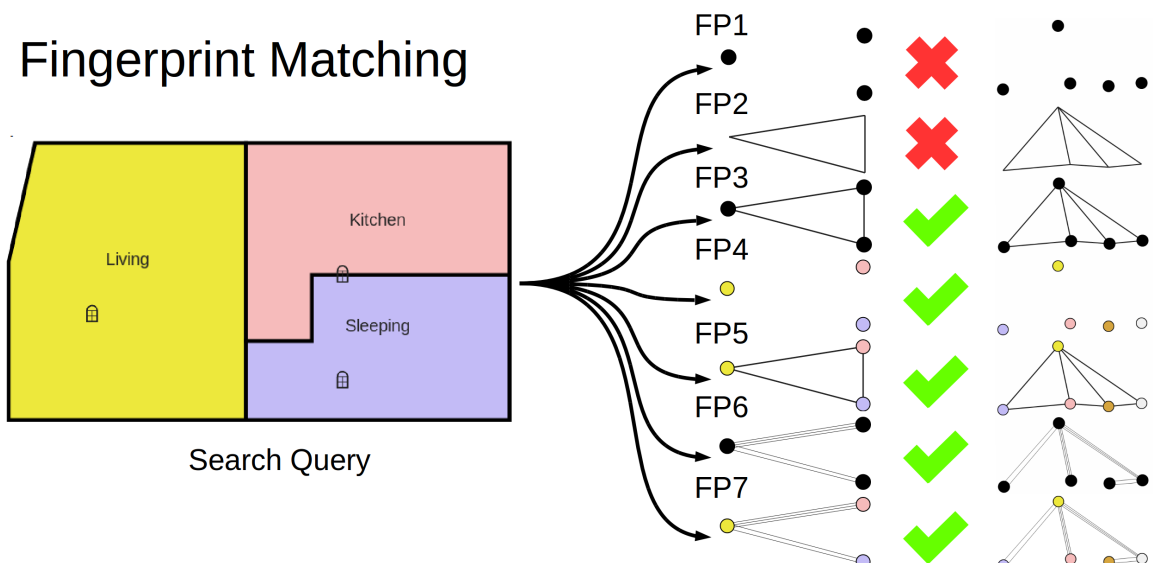


Figure 7: The above diagram shows how the fingerprints are integrated in the retrieval process. The right side of the diagram indicates which fingerprints of the search query are matched with which fingerprints of the floor plan in the data base. Cross indicates no match while green sign indicates a match. The diagram illustrates this for index based method, which also considers the inexact matching.

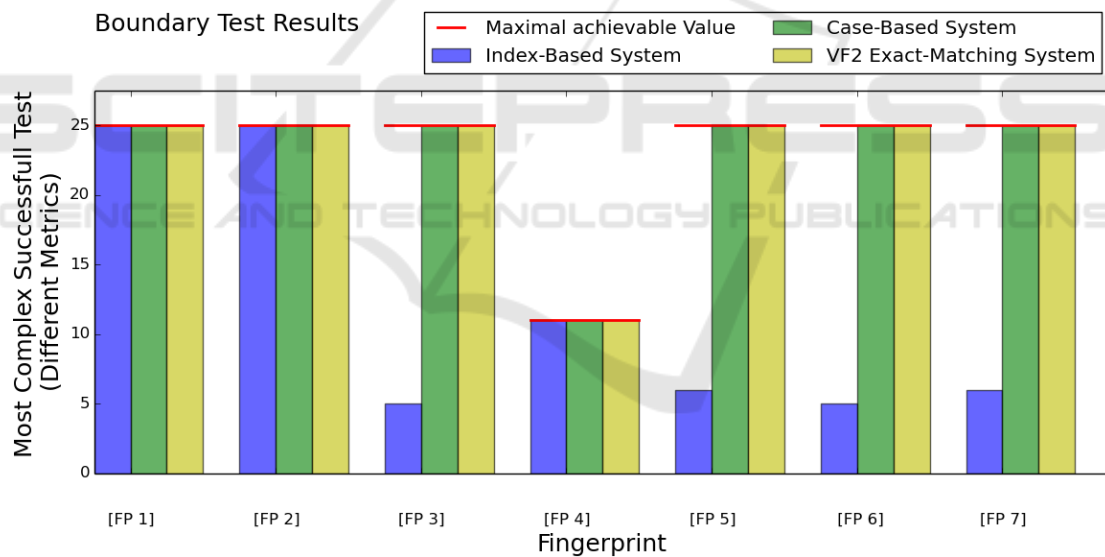


Figure 8: Boundary Test Results. Depicted are the results of the boundary test for each search technique and fingerprint. More precisely, for every combination of search technique and fingerprint the diagram shows the complexity rating of the most complex test scenario the system could handle successfully given a machine with certain computational and memory limits. The metric of complexity differs for the test scenarios of different fingerprints.

table accumulates the overall rankings and presents the summarized results for each query. The ranking results are divided into two categories, where the first category represents queries for which the corresponding retrieval method was determined as the clear winner, that is, was considered best by all participants. The second category is similar to the first one, but shows which method won the majority of rankings for

the particular query. In case of equality of votes, the corresponding query is placed in the third column. In the last column the percentage of queries dominated by the particular method is given.

Table 1 shows that each of the three retrieval methods is good for some query cases. In Figure 9, we show the two best results of three selected queries for each search method, where for each of the three se-



Table 1: Accumulated results of the result sets for the selected queries. The first category represents queries for which the corresponding retrieval method was considered best by all participants. The second category shows which method won the majority of rankings for the particular query. In case of equality of votes, the corresponding query is placed in the third column. The last column contains the percentage of queries dominated by the particular method.

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%

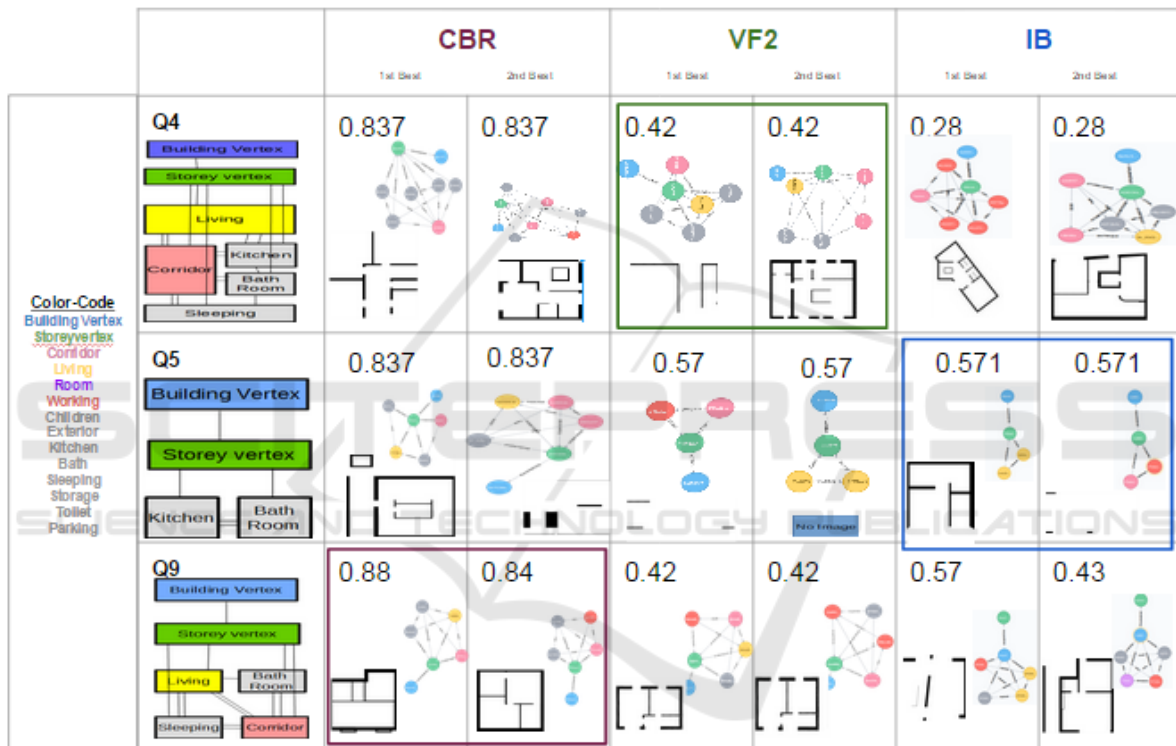


Figure 9: Computed similarity values of the result sets for the selected queries. Color codes represent the room purposes, the first column contains the queries with rooms and assigned purposes. For each retrieval method the first two best single results for the corresponding query are represented. Each single result box consists of a confidence score, the corresponding graph representation with color codes, and the graphical floor plan representation. The result set considered best by the participants is enclosed in a colored box.

lected queries the best case of each of three retrieval methods is represented, respectively, according to Table 1. For better understanding, how the query and the results looked like and which representations the architects would use while working with our system, we visualized them using both graph-based (the top representation in Table 1) and graphical (the bottom representation in Table 1) representations. This figure does not include the subjective rating of the results, only the confidence scores computed by the corresponding retrieval method are presented. It is notable

that the CBR method has a higher score in relation to other two methods. This can be explained with the fact that the application of some fingerprints (e.g., Room Purposes, Room Count or Edge Count) caused the finding of the (almost) exact matches of the room label sets or number of the contained rooms or room connections. This increased their confidence in the accumulated result set. This difference in scoring is observable in other result sets as well.

All things considered, the retrieval methods were able to present sets of reasonable results, taking the

current stage of the respective method development into account. According to the subjective rankings of results and in contrast to the computed confidence score, the VF2 exact matching method was the best one in the overall rating. This fact supports the assumption that exact matching is a better method for determination of isomorphism in graph-based datasets, where a set of tolerable technical limitations is given.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we introduced Archistant, a complete end-to-end system for search and retrieval of similar floor plans for architects. By using this system, we now have a complete automated pipeline for sketching architectural concepts and searching for similar ideas. We showed the capabilities of the implemented retrieval techniques by means of both a boundary test study (in order to show fundamental performance capabilities and limitations of the methods) and a qualitative study (in order to show that the methods produce reasonable results). In the boundary test study, the VF2-based and MetisCBR replied flawlessly and the index-based method at least answered most queries without errors. In the qualitative analysis, each method showed at least in some situation good performances. The currently implemented fingerprints mainly take graph-based floor-ground properties into account. Geometric properties are not considered. But these information have a big potential for making search results more accurate. A large-scale user study resulting in a big ground-truth database may be conducted in future and may hereby lead to the deployment of machine learning for automatically selecting the best approach for every search scenario in future.

## REFERENCES

- Ahmed, S., Weber, M., Liwicki, M., Langenhan, C., Dengel, A., and Petzold, F. (2014). Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recognition Letters*, 35:91–100.
- 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, *Proceedings of the 8th International Conference on Agents and Artificial Intelligence. International Conference on Agents and Artificial Intelligence (ICAART-2016), February 24-26, Rome, Italy*. SCITEPRESS.
- Ayzenshtadt, V., Langenhan, C., Roth, 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. International Conference on Case-Based Reasoning (ICCB-16), 24th International Conference on Case Based Reasoning, October 31 - November 2, Atlanta, Georgia, USA*. Springer, Berlin, Heidelberg.
- Bayer, J., Bukhari, S. S., Langenhan, C., Eichenberger-Liwicki, M., Althoff, K.-D., Petzold, F., and Dengel, A. (2015). Migrating the classical pen-and-paper based conceptual sketching of architecture plans towards computer tools - prototype design and evaluation.
- Bengoetxea, E. (2002). Inexact graph matching using estimation of distribution algorithms. *Ecole Nationale Supérieure des Télécommunications, Paris*, 2:4.
- Cavieres, A., Bhatia, U., Joshi, P., Zhao, F., and Ram, A. (2011). CBArch: A Case-Based Reasoning Framework for Conceptual Design of Commercial Buildings. *Artificial Intelligence and Sustainable Design - Papers from the AAI 2011 Spring Symposium (SS-11-02)*, pages 19–25.
- Chen, C., Yan, X., Yu, P. S., Han, J., Zhang, D.-Q., and Gu, X. (2007). Towards graph containment search and indexing. In *Proceedings of the 33rd international conference on Very large data bases*, pages 926–937. VLDB Endowment.
- Cheng, J., Ke, Y., Ng, W., and Lu, A. (2007). Fg-index: towards verification-free query processing on graph databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 857–872. ACM.
- Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2004). A (sub) graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–1372.
- Foggia, P., Sansone, C., and Vento, M. (2001). A performance comparison of five algorithms for graph isomorphism. In *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pages 188–199.
- Giugno, R. and Shasha, D. (2002). Graphrep: A fast and universal method for querying graphs. In *Pattern*

- Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 112–115. IEEE.
- 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 (CAADRRIA)*, pages 389–398.
- 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.
- McKay, B. D. et al. (1981). *Practical graph isomorphism*. Department of Computer Science, Vanderbilt University Tennessee, US.
- Messmer, B. T. and Bunke, H. (1999). A decision tree approach to graph and subgraph isomorphism detection. *Pattern recognition*, 32(12):1979–1998.
- 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*, 25:285–292.
- Schmidt, D. C. and Druffel, L. E. (1976). A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *Journal of the ACM (JACM)*, 23(3):433–445.
- Sharanya Jayaraman, S. V. (2013). Comparative survey of query processing on graph databases. Project report, Florida State University.
- Smyth, B. and McKenna, E. (1999). Footprint-based retrieval. In *Case-Based Reasoning Research and Development*, pages 343–357. Springer.
- Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42.
- Voss, A. (1997). Case design specialists in FABEL. *Issues and Applications of Case-based Reasoning in Design*, pages 301–335.
- Wessel, R., Blümel, I., and Klein, R. (2008). The room connectivity graph: Shape retrieval in the architectural domain.