Thinking With Containers: A Multi-Agent Retrieval Approach for the Case-Based Semantic Search of Architectural Designs

Viktor Ayzenshtadt[1,2], Christoph Langenhan[3], Saqib Bukhari[2], Klaus-Dieter Althoff[1,2], Frank Petzold[3] and Andreas Dengel[2]

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

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

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

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

Abstract: To provide the retrieval of information, that can be considered useful during the design conceptualization process, with advantages of distributed artificial knowledge, an approach, that distributes retrieval-related and knowledge maintaining tasks among autonomously working and case-based self-learning agents and agent groups, can be used. In this work we present the distributed retrieval system MetisCBR for the architectural design domain, where agents work in groups (containers) on resolving of user queries built with a semantic description model Semantic Fingerprint. The main aim of our approach is to carry out a basis for a considerable retrieval tool for architects, where the combination of case-based reasoning and multi-agent methods helps to achieve valuable and helpful search results in a comprehensive building design collection.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The early conceptualization phase of a building is the stage of the architectural design development, during which an architect can increase the efficiency of the working process by communicating with a computer-aided helper system. Such a system should be able to present new ideas and helpful recommendations for inspiration and guidance in combination with similar building designs to a currently composed one.

The study of buildings that have a similar context to the design problem at hand, or that are based on a similar initial premise, is seen as a way of approaching a design problem and developing a possible course of action. Prior built projects or architectural designs serve here as a knowledge base that contains both examples of spatial constellations as well as solutions for specific architectural situations.

Plans, models, and the documentation of architectural designs and existing buildings and urban environments represent an extensive architectural knowledge base which contains implicit design knowledge that is valuable for the design process. The problem is, that this knowledge is not available in an explicit form, without first analyzing and interpreting the information available.

To handle this architectural knowledge, for the purpose of providing a retrieval component that efficiently interprets this data, the helper systems often use *case-based reasoning* (CBR) methods as the underlying retrieval technique. This aims to achieve high concentration on the quality of results, and to make the system knowledge-intensive and extensible.

For implementation of such retrieval features, many approaches were proposed and applied to the existing systems in the last three decades. The most significant of them are listed in (Heylighen and Neuckermans, 2001) and (Richter et al., 2007).

In this work we present *MetisCBR*, a multi-agent and case-based retrieval approach for similar architectural designs. This approach uses inter alia case-based learning agents as retrieval units in a *distributed container-based* application. MetisCBR was implemented as a prototype and integrated into the existing infrastructure in context of a basic research project *Metis – Knowledge-based search and query methods for the development of semantic information models (BIM) for use in early design phases*, an interdisciplinary project of CBR, *multi-agent systems* (MAS), and *computer-aided architectural design* (CAAD), partially funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

This paper is structured as follows: first, the background and the interdisciplinarity of the research

Ayzenshtadt, V., Langenhan, C., Bukhari, S., Althoff, K-D., Petzold, F. and Dengel, A.
Thinking With Containers: A Multi-Agent Retrieval Approach for the Case-Based Semantic Search of Architectural Designs.
DOI: 10.5220/0005683801490156
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 1*, pages 149-156
ISBN: 978-989-758-172-4
Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

project *Metis* will be briefly presented. The next section contains the overview of related approaches and studies. Next, a short overview of the system structure will be presented. The next section characterizes the proposed agent types and their corresponding tasks in detail. In the next section the prototype of the approach and its context and validation within the research project will be presented. The conclusion summarizes the work and the future project planning.

2 BACKGROUND

Technological developments and globalized working processes have transformed the building design process. However, current digital semantic building models are no longer able to adequately represent the increasing complexity of modern architectural projects. The potential of combining agent-based, case-based, and graph-based methods with novel semantic models, for example, for energy calculations or spatial research strategies, is not fully exploited. To find essential ways to utilize this potential, a basic research project *Metis* was initiated. The distributed CBR retrieval system *MetisCBR* we present in this work is one of the integrant parts of the project. The interdisciplinary nature of the project provides a necessary mixture to overcome the problem of timely finding similar building designs as source of inspiration to guide the design process of buildings.

CAAD considers computer-related topics regarding buildings and delivers insights into ways an architect can work/interact with such a system or how she would evaluate search results.

CBR, as the discipline responsible for working with intensive knowledge data, is the means for the actual finding of similar building designs. CBR's all-purpose nature and a multitude of different CBR retrieval approaches provide efficient tools for the accomplishment of the project's main tasks.

MAS play a role of executing units in the project. Being a means for distributed decentralized problem solving, MAS, and the corresponding research area, provide a necessary collection of techniques for distributed and concurrent retrieval processes.

To combine the above named fields in order to work together in a common system, a *building information model (BIM) infrastructure* was built for the project. It integrates every project-related service, including a building model server, a graph database with building designs represented as graphs, a content management system, and software prototypes.

3 RELATED WORK

The system we present in this work is set in context of the above mentioned research project *Metis*, and is related to the CAAD and BIM applications with case-based reasoning as the underlying retrieval technique.

The project-related work includes (Siebert, 2014), a research work that contains the initial setting of our system where a concept of agent-based experience sharing system is applied to the architectural domain.

A semantic building structure description model *Semantic Fingerpint* (Langenhan and Petzold, 2010) was initiated to provide a structure to idea that building floor plans can be described by abstracting the building data at the level of building, storey, and room, as well as their respective topological relationships. Based on this model, a subgraph matching-based retrieval algorithm was proposed in (Ahmed et al., 2014), where an information extraction process of floor plans is also contained and later used in the *Metis* project for the graph database.

Applications for Android and iPad, and a touch-table interface, were developed for query construction with modern haptic HCI methods. A web-based user interface (Bayer et al., 2015) is part of the project as well. This application uses the *(AGraphML)* specification (Langenhan, 2015) for the query construction.

Finally, a distributed domain model that builds a CBR base for the retrieval process of *MetisCBR* was introduced in (Ayzenshtadt et al., 2015).

CBR-based approaches with architectural emphasis can be classified as a subdomain of *case-based design* (CBD). In these systems, the retrieval functionality is one of the most essential system modules.

For example, in FABEL (Voss, 1997) functional entities called *specialists* are used to execute the retrieval tasks. The *specialists* work with special *aspect-specific* representations of cases and queries. These representations are produced by applying a transformation process to the user query or to a currently comparing case of the case base. *Retrieval specialists* determine similar cases by comparing the corresponding aspects of cases and queries.

CBArch (Cavieres et al., 2011) is a supporting framework for the conceptualization phase of architectural design. With emphasis on the early energy performance evaluation of buildings with commercial background, it provides an approach for a completely CBR-based solution for such cases.

Other systems, such as DYNAMO (Heylighen et al., 2002), SEED (Flemming, 1994), PRECEDENTS (Oxman and Oxman, 1993) or CaseBook (Inanc, 2000), can also be mentioned as closely related, as they serve the same purpose as our system.

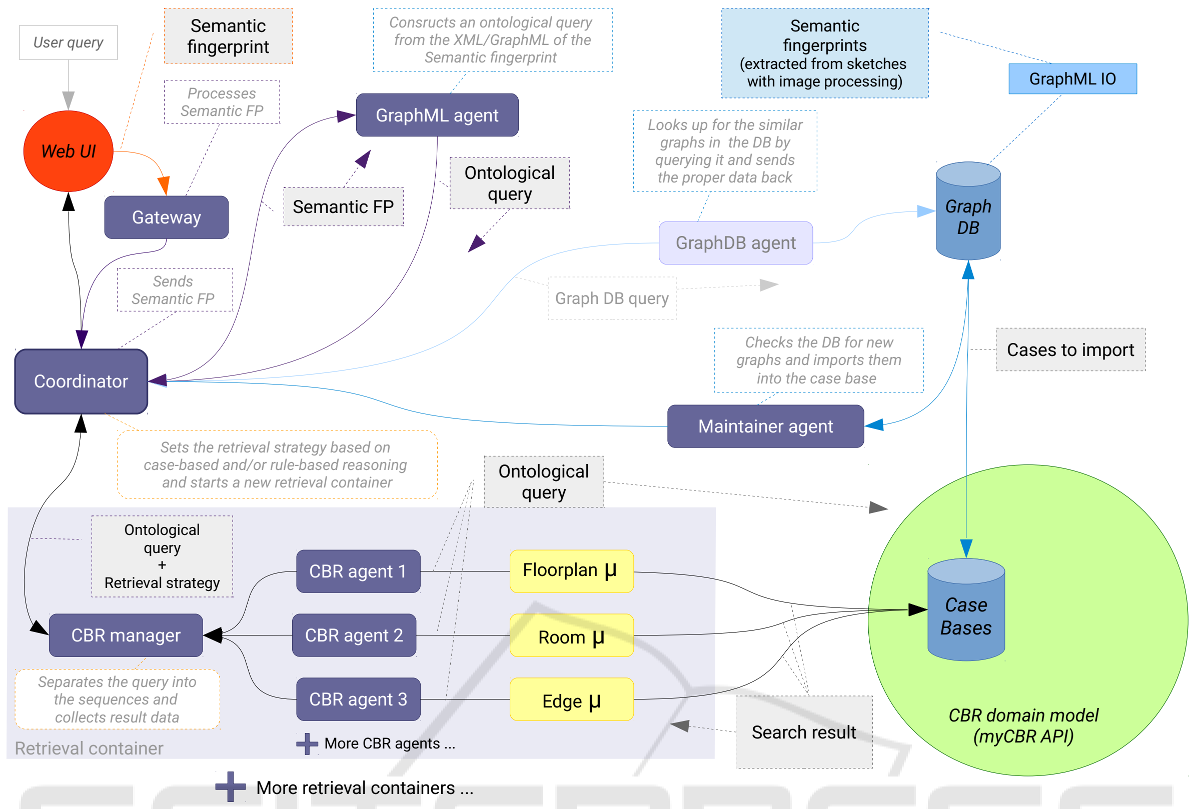
User query
Semantic fingerprint
Processes Semantic FP
Consructs an ontological query from the XML/GraphML of the Semantic fingerprint
GraphML agent
Semantic fingerprints (extracted from sketches with image processing)
GraphML IO
Web UI
Gateway
Semantic FP
Ontological query
Looks up for the similar graphs in the DB by querying it and sends the proper data back
Graph DB
GraphDB agent
Graph DB query
Sends Semantic FP
Coordinator
Checks the DB for new graphs and imports them into the case base
Cases to import
Maintainer agent
Sets the retrieval strategy based on case-based and/or rule-based reasoning and starts a new retrieval container
Ontological query
Ontological query + Retrieval strategy
CBR agent 1
Floorplan μ
CBR manager
CBR agent 2
Room μ
Case Bases
CBR agent 3
Edge μ
Separates the query into the sequences and collects result data
Search result
CBR domain model (myCBR API)
Retrieval container
More CBR agents ...
More retrieval containers ...

Figure 1: The system overview of MetisCBR.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

4 THE SYSTEM OVERVIEW

In this section we provide a brief overview of the proposed approach *MetisCBR*. The main task of the entire multi-agent system is to process a query in the AGraphML format that was sent by a user *(an architect)* from a web-based query construction interface (not part of this approach, see also Sections 3 and 6).

As depicted in Figure 1, the system is constructed in a circular form, where the *gateway* marks the entry point into the system, after a user query was sent from the web interface.

Successfully passing the gateway agent that conducts an initial error analysis, the query reaches the *coordination component* that is responsible for the correct accomplishment of the entire case-based retrieval process. Its first action is to send the query to the parsing agent that transforms it from the architectural GraphML into an ontological representation in *SL* content language to provide human-readability as well as suitability for communication among agents according to (Caire and Cabanillas, 2002).
Back at the coordination point, the query is analyzed and sent to a *retrieval container* that conducts the actual resolving of this particular query. The coordinator selects the proper retrieval strategy and conducts the setup of the container, where a *retrieval manager* and at least one *case-based retrieval agent* that operates with at least one *similarity measure* μ are mandatory units for successful processing of the retrieval stage in the case base(s) of the CBR domain model.

After finishing the actual retrieval process, the achieved result data is sent back from the container's manager to the coordinator, who forwards it to the gateway that transforms it into the proper processing format (JSON) and sends the result data object to the web interface component for displaying.

Parallel to the user query resolving process, new cases *(building designs)* are being imported into the case base(s) of the underlying CBR domain model. This is done by another essential component of the proposed MAS, the *maintainer*, whose main task is to extract cases from the BIM infrastructure's graph database and to import them into these case base(s).

5 AGENT TYPES AND TASKS

This section presents detailed descriptions of the agent types that are directly or indirectly involved in the process of retrieval of similar building designs. Their tasks will be also described in detail.

5.1 Agent Categories

The categorization of agents into classified groups helps to achieve a better understanding of the entire system functionality. For our case-based multi-agent approach with retrieval as main task, we defined the following categories of agents: *case-based*, *managing*, *connecting* and *service* agents. An agent may belong to one or many categories.

Case-based agents are *directly* involved in the retrieval process of similar architectural designs *and* possess an internal reasoning and learning component with rule-based and/or case-based reasoning abilities. Retrieval executing agents of this category (see Section 5.3.2) are able to apply different similarity measures when searching for designs (or parts of it) in the case base(s) of the CBR domain model described in (Ayzenshtadt et al., 2015) (see also Sections 3 and 6).

Managing agents ensure that the entire system (including the retrieval process and building designs import into the case base(s)) is working properly, without failures, such as abruptions or sudden bottlenecks. The agents of this class monitor ongoing internal processes, share their statuses/intentions, and forward queries and results to the corresponding agents.

Connecting agents play the role of connection points between the MAS and linked external services. They make sure that the connection and communication of *MetisCBR* with these environments (e.g., the query builder interface or the graph database) is established, and query or case streams can be forwarded to the destinated system components.

Service agents are helper agents that execute tasks that do not belong directly to retrieval, management, or connection task domains. The agents of this class are involved in the accomplishment of intermediate steps of the retrieval process (e.g., query parsing).

5.2 Coordination

For the efficient coordination of the retrieval process within our system, a corresponding agent (the *coordinator*) was implemented. Its role is to manage the entire case-based retrieval stage, after the user query has been received from the gateway. Therefore, this agent belongs to both categories of *case-based* and *managing* agents, and has a number of particular tasks.

Retrieval Strategy Selection. The selection of the proper retrieval strategy is the coordinator's most essential task. It is based on the combination of the properties of the query, previous experiences, rules, and user-specified data from the query.

For example, if the user's current objective is to conduct a retrieval without much detailed settings for the search, the coordinator can derive the objective from the user-specified data (in form of a specific attribute) and select the associated retrieval strategy to resolve the query. Otherwise, the common strategy can be applied. The objective recognition and setting is an example of applying the rule-based reasoning, the first one of the reasoning abilities of the coordinator. The second one is the case-based classification of the query, where selection of the strategy is mainly the task of the learning component of the coordinator.

Learning is a significant property of an agent that aims to improve its ability to make decisions based on previous experiences. Therefore, the coordination component of our retrieval system, being such an agent, uses its case-based learning ability as an essential feature to select the proper procedure for query resolving. Each query that the coordinator gets from a user, together with the achieved results, is registered as a case with the IB2 algorithm (Aha et al., 1991) that controls the filling of the coordinator's internal case base and provides the coordinator with the learning ability. These cases are the base for the reasoning process that gets activated if the rule-based classification misses the explicit specification from the user side. During the CBR-based classification, the corresponding floor plan information of the query (e.g., room count and room types) is used to find the most similar case in the coordinator's internal case base, and apply a strategy that was used in this case.

Retrieval Container Activation. After the strategy for the retrieval has been selected, the coordinator activates a new retrieval container (see Section 5.3). The setting of the container is defined by the coordinator – it selects the retrieval agents that will execute the actual retrieval steps and starts the activity of the container. In *MetisCBR* the retrieval strategy is essential for selection of the agents – *retrieval agent types are bound to the strategy*, but *not relevant* for the selection of the strategy. The coordinator is also able to terminate one or more currently running retrieval containers. This happens when a container has finished its retrieval activities, *and* the user has finished her retrieval session. In case of unexpected interruption of the connection between the user interface and the MAS, the retrieval is continued and the results are persisted in an intermediate object.

Forwarding of the Query Modifications. If it is the user's wish to modify the current query during the ongoing retrieval, the coordinator receives the modified query from the gateway agent and forwards the changes while notifying the manager of the corresponding container about these changes.

Sending of the Results to the Gateway. When the actual retrieval phase is finished, the coordinator receives the results from the manager of the container and provides the gateway agent with these results.

Assignment of the Query Parsing Task. The coordinator assigns the *GraphML agent* (see Section 5.6) the task of parsing the current query, which it gets back transformed as an ontological query in SL content language format.

Receiving the Import Status. From the maintainer agent the coordinator receives the current status of the extraction/import process and coordinates the currently running retrieval tasks with respect to this status (see also Section 5.5).

5.2.1 Coordination Techniques

The efficient accomplishment of the above described tasks requires possession of corresponding coordination techniques. The coordination component of our retrieval system is developed with the integration of combination of some of the coordination scenarios described in (Nwana et al., 1996).

Organizational Structuring is applied when the coordinator creates and destroys retrieval containers and assigns strategies to them. It is the main coordination technique in *MetisCBR*.

Contracting (more precisely, a sub-feature of contracting, the *sub-distribution* of the tasks) is applied when a manager of a retrieval container assigns parts of the query to be resolved by corresponding agents.

Centralised Multi-agent Planning is activated when the maintainer sends an import status to the coordinator (see Section 5.5)

5.3 Retrieval Container

In a retrieval system, where concurrent queries are expected to be a usual case, a big advantage is to implement a structure that is able to handle each of the queries separately to achieve more efficient processing. For the actual building design retrieval step of our approach, we use a *container-based* structure, where each *retrieval container* during its activity can be seen as a completely autonomous multi-agent sub-system that is nested inside a parent MAS, and can only be terminated if its retrieval process and the user session are both finished.

The working cycle of a retrieval container starts with receiving of an ontological query from the coordinator and ends after the retrieval of the assigned query is finished. Between those two steps, a case-based resolving of the query takes place inside the container. Thus, every container has three goals to complete during its working cycle: *receiving* of the query, *resolving* of the query by means of applying CBR methods, and *replying* to the coordinator with a message that contains the achieved results.

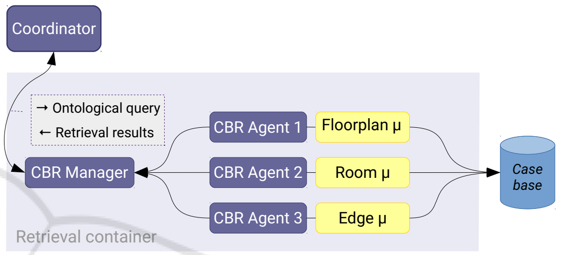

Figure 2: Exemplary configuration of a retrieval container.

An exemplary configuration of a single retrieval container can be seen in Figure 2. Every container consists of a mandatory *CBR manager* – an agent that manages resolving of the current query – and at least one mandatory *CBR retrieval agent* – an entity that is responsible for the actual search for similar cases. The following sections provide a detailed description of both agent types.

5.3.1 CBR Manager

As mentioned above, the CBR manager is a mandatory agent inside an active retrieval container. It organizes the current case-based retrieval of an ontological query that was received from the coordinator. Therefore, this agent belongs to the category of *managing agents* (see Section 5.1), and was primarily created to take load off the coordinator, that can in turn concentrate on its strategic and system managing tasks.

The tasks of the manager include two primary classes: *communication with the coordinator* and *query separation*. During the communication with the coordinator, (i.e., receiving an ontological query and sending the results back), the manager completes *receiving* and *replying* steps of the container goals. The separation process cuts the given query into the number of parts according to the selected retrieval strategy and forwards these parts to the corresponding agents.

The secondary class of tasks of the manager consists of helper operations, which are accomplished only if required by the strategy as well. Those helper operations can include some computational tasks, such as determination of rooms and room connections that originate from the same floor plan.

5.3.2 CBR Retrieval Agents

CBR retrieval agents execute the last and most important part of the retrieval process – they apply CBR retrieval methods for determination of the most similar cases to the query or its parts in the case base(s) of the underlying *attribute-value based* CBR domain model (described in detail in (Ayzenshtadt et al., 2015)). To achieve this goal, they use similarity functions prepared for each agent, special *query types*, and an internal reasoning mechanism to deal with incoming queries from the manager. CBR retrieval agents belong to the *case-based agents*, where several agent types of this class currently exist.

Floor plan Agent. This agent's aim is to search for cases that contain *floor plan meta data* from the AGraphML specification and additional related information such as room count and room types. Floor plan agent works with following query types:

- `Floorplan Query` – every attribute of the meta data and additional data will be taken into account for the similarity measurement.
- `Floorplan ID List Query` – returns cases, if their corresponding IDs exist in a defined ID list.

Room Agent. To find similar rooms to a given one this agent uses every room attribute from the domain model. Its only query type is currently `Room Query` that accomplishes this similarity computation task.

Edge Agent. This retrieval agent takes every edge attribute into account to compute a similarity measure between two room connections. The corresponding query type is the `Edge Query`.

5.3.3 Learning of CBR Retrieval Agents

CBR retrieval agents are able to learn from previous queries in the same manner as the coordinator does. Each of the retrieval agents possesses an internal reasoning mechanism, where a special *internal* CBR domain model exists for each retrieval agent's knowledge base. Special attributes are determined to hold information about previous queries the agent has resolved (e.g., room count from the query and from the found case(s) for the *Room agent*).

5.3.4 Container Identification

Retrieval container can be identified via an UUID as defined in the RFC 4122. The coordinator assigns an UUID to each container it creates, the same ID is also temporarily assigned to the query that the container is working with to be able to forward modifications to the associated container.

5.4 Gateway

As described above in the overview section, the *gateway agent* is an entry point to the MAS from the user side, and an interface for forwarding queries and results in corresponding directions. The gateway agent belongs to the category of *connecting agents* (see Section 5.1).

Gateway agents of such type are common in systems with structure similar to ours. For example, in (Greenwood et al., 2005) a gateway agent is a part of a component that is used for the OWL-S-based semantic interaction between a web client from an external web service environment and an agent environment.

The activity area of our gateway agent is divided into two main sections: *accepting* of user queries and forwarding them to the coordinator, and *receiving* results from the coordinator and forwarding them to the web user interface. Both task sections are controlled by special agent behaviors (see Figure 3).

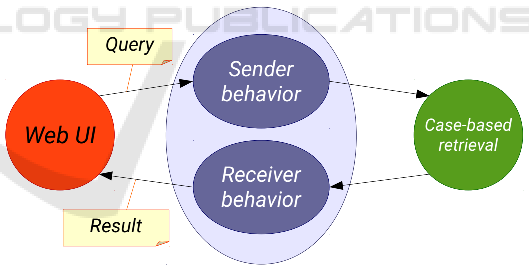

Figure 3: The behaviors structure of the gateway agent.

Sending a formalized and validated user query in AGraphML format to the coordinator is the main purpose of creating of the *Sender behavior*. During the validation of the query the gateway agent traverses through the data in order to find formal errors, such as missing meta data or false (or missing) room or room connection definitions. Empty queries will not be forwarded to the coordinator. Incomplete queries, where data is fragmentary or only partly missing, will be forwarded to the coordinator, however, missing attributes will not be used for the retrieval.
Receiving result data from the coordinator and delivering it to the web-based user interface is the activity

field of the *Receiver behavior* of the gateway agent. Similar to the process of the sender behavior, the result data (in SL content language) will be validated as well, following by the transformation into the JSON format (plain HTML is also possible). It is optional if results will be returned in sequences or at once.

5.5 Maintainer

Maintainer is an agent that is responsible for the correct functionality of the extraction and import of the graph-based designs from the BIM infrastructure's graph database into the case base(s) of the underlying case-based domain model of the system. The maintainer belongs to the class of *managing agents*.

During the three-step case base(s) update process the maintainer executes tasks defined for each step. In the first phase it queries the graph database for new graph representations of building designs. In the second phase, the maintainer requests the RESTful service (implemented in the BIM infrastructure) for each graph of the previous result set and gets back an AGraphML-formatted graph. In the last step, the building graph is imported to the case base(s) of the domain model with a special import function.

For the essential workflow of the system, the maintainer is also able to set and share an *import status*, which currently can be one of the following: `Import Ready`, `Import Active`, `Import Paused`, and `Import Done`.

The coordinator receives the status and may execute actions according to it. For example, if the system performance is critical the coordinator may stop some retrieval containers or request to stop recently started extraction/import process to allow for a better system resource sharing.

5.6 GraphML Agent

As previously mentioned (see Section 5.2), the *GraphML agent* is the system entity that is responsible for parsing of the incoming architectural GraphML queries into ontological queries in *SL* content language format. The ontological queries get their name from the underlying communication ontology that defines the communication vocabulary among agents. By means of applying an ontology to the communication, the ontological query becomes part of the message content as demonstrated in (Caire and Cabanillas, 2002).

6 SYSTEM EVALUATION

For the above mentioned basic research project *Metis*, a working prototype of the proposed retrieval system *MetisCBR* was developed to demonstrate its suitability for the "real-world" application. Java language based frameworks *JADE* (for the implementation of the described agent system) and *myCBR* were used to develop the prototype. It is also implemented as one of the sub-systems of the BIM infrastructure.

The prototype works with the BIM infrastructure's content management system (that contains *graphical* representations of building designs) and from it derived graph database that contains every building design as *graph representation* that is then extracted and imported as specified in Section 5.5. In Section 3 mentioned web-based user interface is planned to be used as the main *graphical* architectural GraphML query builder for the retrieval system. Furthermore, the proposed system and the software prototype are part of the master thesis (Ayzenshtadt, 2015).

For the validation and demonstration of the abilities of the approach, an evaluation of the system was conducted, that is completely described in (Ayzenshtadt et al., 2015). The evaluation process included the querying of the system with an exemplary query consisting of 5 rooms and 6 connecting edges with two different similarity measures: with and without attribute weighting.

The *MAS task area* of the evaluation should show that *MetisCBR* is already able to handle search requests and return results according to the similarity measures.

Considering the early phase of the system implementation, the agents were able to properly handle the user queries. The coordinator was able to recognize a new query, to create a corresponding container, and to assign the strategy and the query type. The manager of the container was able to recognize if it is required to separate the query, and then assigned the corresponding parts to the CBR retrieval agent(s). The retrieval agent(s) retrieved the case base with the similarity measure that was set for the current query (manually, on this stage of development). The manager collected the results and sent them to the coordinator that then forwarded them to the gateway agent. A couple of minor technical problems occurred during the import process, but could be traced by inspecting the consistency service of the BIM infrastructure.

The results of the evaluation of the CBR task area, including the information about retrieval strategies, similarity measures, building design dataset used, average and highest similarity, and the similarity distribution, are described in (Ayzenshtadt et al., 2015).

7 CONCLUSION

In this paper we presented *MetisCBR*, a distributed case-based approach for the retrieval of similar building designs in the corresponding case base(s). The system consists of different agent types, where the coordinator controls the retrieval process and applies rule-based or case-based reasoning to select the proper retrieval strategy, retrieval containers execute concurrently the actual case-based retrieval processes, the gateway agent connects the system with an external service (web-based query builder) and the maintainer agent manages the import of new cases into the case base(s) of the underlying CBR domain model. Service agents contribute to the smooth run of the system by executing small intermediate tasks.

The future work includes the full integration of the graphical user interface to the presented MAS. Index-based retrieval method that queries the graph database directly is also planned to be added, a new *GraphDB* agent will be integrated to the MAS to accomplish this alternative retrieval task. This will also allow for evaluating and hopefully deeply integrating these different kinds of retrieval approaches.

ACKNOWLEDGEMENT

This work for the *Metis* project was supported by the DFG (*German Research Foundation*).

REFERENCES

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

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. (2015). *Concept and Implementation of a Distributed Case-based Retrieval Approach for the Support of Architectural Conceptualization Phase*. Master thesis. University of Hildesheim.

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.

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*.

Caire, G. and Cabanillas, D. (2002). Jade tutorial: application-defined content languages and ontologies. *TILab SpA*.

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 AAAI 2011 Spring Symposium (SS-11-02)*, pages 19–25.

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

Greenwood, D., Nagy, J., and Calisti, M. (2005). Semantic enhancement of a web service integration gateway. In *Proceedings of the Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE 2005), Utrecht, The Netherlands*.

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

Heylighen, A., Schreurs, J., and Neuckermans, H. (2002). Enter instead of submit. *DesignNet-Knowledge e Information Management per il design*, pages 171–182.

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.

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.

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., 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*.

Siebert, M. (2014). *Concept of a CBR System for the Architectural Conceptualization Phase Support*. Project report.

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