Knowledge Engineering for Distributed Case-Based Reasoning Systems

Kerstin Bach

Department of Computer and Information Science
Norwegian University of Science and Technology, Trondheim, Norway
http://www.idi.ntnu.no

Abstract. This chapter describes how to identify and collect human knowledge and transform it into machine readable and actionable knowledge. We will focus on the knowledge acquisition for distributed case-based reasoning systems. Case-based reasoning (CBR) is a well-known methodology for implementing knowledge-intensive decision support systems [1] and has been applied in a broad range of applications. It captures experiences in the form of problem and solution pairs, which are recalled when similar problems reoccur. In order to create a CBR system the initial knowledge has to be identified and captured. In this chapter, we will summarise the knowledge acquisition method presented by [4] and give an running example within the travel medicine domain utilising the open source tool for developing CBR systems, myCBR.

Keywords: Case-based Reasoning, Knowledge Acquisition Knowledge-Based Systems, Knowledge Engineering, Distributed Knowledge Acquisition

1 Introduction and Motivation

Following our approach of Collaborative Multi-Expert Systems [2] the knowledge sources, which are used to store and provide knowledge, are mostly distributed. When dealing with complex application domains it is easier to maintain a number of heterogeneous knowledge sources than one monolithic knowledge source. Therefore we propose a Knowledge Line that holds a map of topics, which split rather complex knowledge in smaller, reusable units (knowledge sources). Moreover, the knowledge sources contain different kinds of information as well as there can also be multiple knowledge sources for the same purpose. Therefore each source has to be described in order to be integrated in a retrieval process which uses a various number of knowledge sources. In the following we describe how multiple CBR system holding case of different topics, but aggregate their results eventually, can be built. This knowledge acquisition method is targeted for capturing heterogeneous knowledge sources and build a distributed CBR system that maintains its knowledge in homogeneous containers that are queries on demand. Therewith it fits in the requirements of knowledge acquisition for the SEASALT architecture [14] as well as the CoMES approach [2].

The goal of the knowledge modularization is building as independent modules as possible in order to reduce the complexity of each individual CBR system. Modules can be described as task specific program parts as they are described in [12]. The knowledge modularization in SEASALT aims are minimizing dependencies between Topic Agents by identifying modules that are coherent within themselves. Modules, further on, can be combined as required within a **Knowledge Line**. The result of a problem solving system which is based on SEASALT is always a solution that consists of partial solutions, which originate in heterogeneous partial domains.

For generating a solution within a Knowledge Line a Knowledge Engineer must define the overall contexts for ensure the composition of an overall result. For that reason the Knowledge Modularization is a basis of the Knowledge Composition.

Crucial for the effective application of such knowledge intensive systems is the organization of knowledge. For that reason the conceptional (development) phase requires special attention, because after that phase the development of the CBR systems can be carried out individually. The Knowledge Line contains on the one hand the Knowledge Map for organizing information about each module (Topic Agent) that is required for combining the snippets, which are the partial solutions of which the final result consists of.

2 Background and Related Work

Two related methodologies to the work presented here are DISER [17] and INRECA [8]. While both, DISER and INRECA are methodologies for creating single CBR systems, the process of building a knowledge line instead describes the systematic development of decentralized, CBR systems.

DISER describes a methodology for developing experience-based systems in general. It especially focuses on the integration of the CBR system in an enterprise rather than providing information from Web 2.0 sources to laymen. Furthermore, the knowledge line requires knowledge engineers to execute key tasks while in [17] enterprise executives are addressed with the goal to integrate the system in existing socio-technical processes. With SEASALT instantiations, we are focusing more on the technical realization rather than the social interaction between stakeholders. When recalling DISER's development aspects, then a Knowledge Line can be positioned in the vision to pilot phases. The presented methodologies addresses also maintenance aspects, but has not a particular phase, because novel information is constantly fed into the systems due to the stream-like data in web forums.

INRECA on the other hand was particularly focused on developing CBR systems and provides experiences for the development itself on various abstraction levels. Because of the specialization to CBR, common experiences derived from the development of various CBR systems, can be generalized and shared. INRECA does not cover any experiences for developing distributed systems such as the previously described snippet descriptions. Eventually a Knowledge Line

can be seen as an addition to the INRECA recipes since it presents an approach for distributed CBR systems, which is has not been covered before.

On INRECAs *Common Generic Level* contains more abstract processes for the development of any kind of CBR system. The herewith introduced methodology focuses on a subset of systems that are based on heterogeneous CBR systems. For that reason, only the abstract process presented in Figure 3 would fit that generic level. It can for instance be applied for distinguishing whether a distributed system is required to fit the expectations or not. The *Specific Project Level* is not directly addressed with the work previously present, but the experiences made during the instantiation of an application such as docQuery could provide extensions to that level.

3 Knowledge Modularization

In this section we are introducing a process model for the knowledge modularization within a Knowledge Line. This process model, describes the goal-oriented development of decentralized, heterogeneous case bases and supports the Knowledge Engineer with a structured approach for modularizing knowledge based on available data. The approach presented in this work does not aim at distributing knowledge for performance reasons, instead we specifically extract information for the respective knowledge sources from Internet communities or to have experts or Knowledge Engineers maintaining one knowledge base. Hence, we are creating knowledge sources, especially Case-Based Reasoning systems, that are accessed dynamically according to the utility and accessibility to answer a given question. Each retrieval result of a query is a part of the combined information as it is described in the CoMES approach. The representation of the knowledge modularization is the Knowledge Map, which will be described in the following subsection. The successive subsection will introduce the Knowledge Line process model and describe the containing sub-processes.

3.1 Knowledge Map

The **Knowledge Map** organizes all available knowledge sources that can be accessed by a Coordination Agent that creates individual requests and combines information. The term Knowledge Map has been introduced by [9] describing working knowledge in an organization. In the aforementioned book they describe a knowledge map from the organizational point of view in which human experts are mapped to topics or expertise fields in order to ensure that everybody in a company knows who is an expert in a certain domain. We transfer this concept in an intelligent agent framework that coordinates different knowledge sources.

A Knowledge Map (KM) consists of a number of Topic Agents TA that are depending on each other and each consist of a software agent A on top of a knowledge base KB. Thus it can be defined as follows:

$$KM = \{TA_1, TA_2, TA_3, \ldots, TA_n\} \ \textit{with} \ \ TA = (KB, A) \tag{1}$$

A Topic Agent is a knowledge-based system itself and the software agent queries it. The Topic Agent collaborates with the Coordination Agent that navigates through the Knowledge Map and asks subsequent questions to the individual Topic agents thus creating an individual path through the map. There are dependencies $Dep_{constraint}$ between the Topic Agents which define that sequence and influence the retrieval executed by one of the subsequent Topic Agents. A dependency exists if one agent's output serves as another agent's input and thus enforces a subsequent query. Since the dependencies between Topic Agents can take any form, we decided to implement the Knowledge Map as a graph where each Topic Agent is represented by a node and directed edges denote the dependencies.

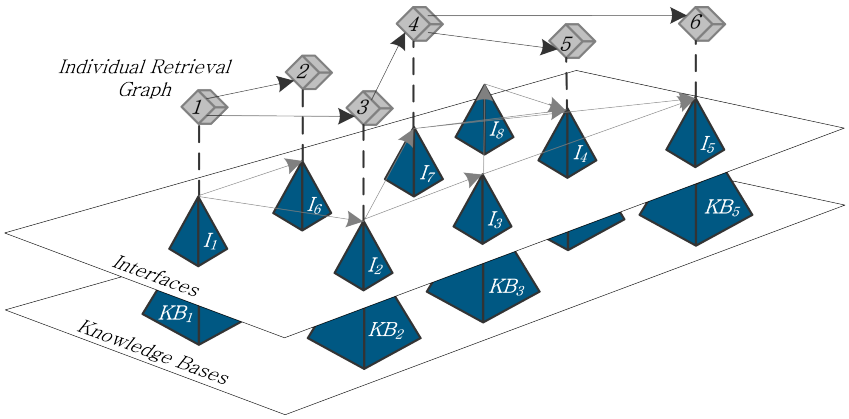

Fig. 1. Knowledge Map containing Topic Agents and Knowledge Bases

Figure 1 shows a Knowledge Map containing the knowledge bases and software agents as well as an example for a possible path through the knowledge sources.

3.2 Pre-Processing

Based on a customers requirement for a target system, the Knowledge Engineer defines the systems functionalities as well we the overall goal. We assume that the overall goal of the knowledge-based system is the composition of results retrieved from multiple, heterogeneous case bases. Examples for this are the menu creation in the cooking domain [6,10] or the travel medicine domain as it will be used as an example further on.

Within docQuery, we require to use information of various CBR systems such as regional information, information on diseases, information on vaccinations and medications or activities. There are dependencies between each case base, but each one covers its own domain: For instance the case base disease contains

information about infectious diseases that work outside of docquery, i.e. in a medical information system, as well. The dependencies or links between case bases are necessary to compile first a retrieval graph over the knowledge map and eventually compose the solution integrating all retrieved cases.

Knowledge Source Sounding After identifying the scope of each individual case base, existing and available knowledge sources have to be investigated whether they can provide the required data and quality. The major goal of this first step is getting an overview of

- What kind of knowledge already exists?
- What has to be reviewed?
- What has to be acquired in order to fit the given requirements?

Knowledge sources are mainly data collections, but also cover experts for certain domains.

For the docQuery use case we looked at what type of knowledge is naturally available. This led us to the CBR systems Region, Disease, Medicament, Hospital and Activity. It is obvious that these information can be plugged in any other application as well. For example, the Region case base can provide information in a travel agency scenario. Based on the discussions with domain experts we further identified the need of the case bases Chronic Illnesses and Associated Condition. Chronic Illnesses and Disease are separated because of their handling in a knowledge map and the information provided about them. While chronic illnesses are information to be specified by the user, docQuery provides prevention of diseases. Further, explanations on chronic illnesses are focused on how to travel with that handicap, while the disease are potentially unknown and we try to avoid the user with an infection. Therefore disease and vectors have to be explained, while chronic illnesses are known to that particular person. The associated condition case base mainly contains prevention information for any kind of occasion. These information can be linked to most of the other case bases and therefore this case base has to be queried towards the end of the querying process.

This example shows that the identification is at least a 2-step-process in order to define the scope of each CBR system. However, for defining attributes and links usually more iterations are necessary. These discussion are led by the Knowledge Engineer, which increases his/her awareness of the domain. Common sources that should be initially considered and were used within the development of docQuery are DBpedia[1] or Google Knowledge Graph[2].

Identification and Representation of Snippets The modularization of the required knowledge is carried out based on the knowledge engineer's understanding of the domain aiming at fitting the given requirements. The main task is the

[1] `http://dbpedia.org/`
[2] `https://developers.google.com/knowledge-graph/`

identification and definition of homogeneous and independent modules that keep their semantic even when they are seen as singletons (*independence*). Each module has to be combinable in order to complement another module (*compatibility*) and requires rules or constraints ensuring a valid combination of the modules (*validity*). The requirements *independence* and *compatibility* are focusing on the information sources while *validity* aims at procedural knowledge applied combining the results.

Within SEASALT the previously described modules are called snippets since the concept is similar to [13]. After their identification, the knowledge sources providing data for the snippet cases have to be determined. Mainly the Knowledge Engineer has to decide which information from sources are included. For the composition of information that prevent travelers from diseases. Since our modularization happens with respect to different sub-domains, the individual case bases that are the result of this modularization are not absolutely independent of each other. Instead they have a net of dependencies between them that indicates what other case bases are affected by changes in one individual case base. These dependencies also affect some of the aforementioned maintenance tasks and split them into two different kinds: those which have to take other case bases into consideration and those which do not. A case factory agent that performs the task of maintaining a case base's uniqueness, minimality, incoherence or consistency can do this without knowing about other case bases. An agent that inserts new cases or adds new data to existing cases is a different matter. For instance if a new case is inserted into the diseases case base, or an existing disease breaks out in a new region, the inserting agent has to check, whether every risk region indicated in the respective disease's regions attribute is actually included in the country case base's underlying knowledge model, otherwise this diseases case will never be retrieved. Another example of such dependencies would be, if one of the regions can not be associated, a domain expert will have to be contacted and asked for specifications, since we assume that in this case there is either a simple typing error, a synonymous name for a region, or the new region will have to be added to the ontology by a domain expert. The same is true for new data being added to the medicament case base. Here the area of application has to yield at least one result in the disease case base, otherwise the new data have again to be passed to a domain expert for a review. Although this approach is rather maintenance-intensive, our medical application scenario requires this very conservative case addition strategy in order to preserve the system's overall accuracy and the case factory approach allows us to realize this strategy and also adapt to new dependencies, should they arise [3].

As an example, travel medical data contain information about countries, diseases, medications, vaccinations as well as descriptions, guidelines, and experiences. Therefore the knowledge in docQuery will be provided in case bases and each case base will contain one specific topic with its own domain model and maintenance agents/jobs. However, following the modularized structure of knowledge in docQuery, CBR agents will be used for each individual topic agent providing information. Aiming at higher accuracy each case base will serve its

own topic and the case format will exactly fit for the type of knowledge. Furthermore the case bases will contain similarity measures, adaptation knowledge and the vocabulary to represent and retrieve cases. In travel medicine it can be dangerous to use CBR for the whole set of information, because the combination of medications, vaccinations, side effects, contraindications, etc. regarding the traveler's health history have to be correct, without any contradicting information. Instead of that we will apply CBR for each topic and do the combination of the responses afterwards using the constraints given in the response sets. Each issue handled in a case base will be provided using CBR methods and the strength of CBR, finding similar information on a given topic, will ensure a higher quality of information provision.

Now we will exemplify four docQuery case bases that are each representing one topic and explain the dependencies between them. The selected case bases are examples to explain our approach and for the implementation of docQuery there will be at least six more case bases. The case base country will contain specific country information a traveler has to consider planning a journey. Furthermore the information will be separated in the sections a traveler has to pay attention to before, during, and after the journey. The country information also includes the required vaccinations and additional information, for example guidelines for a healthy journey. The case base disease holds more than 100 diseases considered in a travel medical consultation. It concentrates on diseases that might affect a traveler on a journey, for instance Malaria, Avian Influenza, or Dengue. A disease in this case base is characterized by general information on the disease, how to avoid the disease, how to behave if one has had the disease before, and how to protect oneself. The third case base we will introduce is medicament with details about medicaments and their area of application (diseases, vaccinations, age, etc.). Basically it contains information about active pharmaceutical ingredients, effectiveness, therapeutic field, contraindication, inter-dependencies, and the countries in which those medicaments are approved. In diseases we do not store information on chronic illnesses, be-cause they will be modeled in their own case base. Instead we focus on diseases which can affect travelers during their journey. The fourth case base will hold activities which are used within docQuery to provide safety advice for intended activities when planning a journey. For travelers, activities are the major part of their journey, but may involve certain risks for which safety advice is needed and furthermore while asking for their plans they usually describe their activities which we can use to provide better guidance. Examples of such activities are diving, hill-climbing or even swimming.

Complete travel medical information will contain knowledge of all four case bases enhanced with descriptions, guidelines, and previous experiences. The combination of the information retrieved from each case base will be done by a Coordination Agent as it can be seen in the SEASALT architecture (Figure 2). The coordination agent will request each agent and based on the agents' response and the given information by the traveler the next request containing all constraints to another topic agent will be created and send.

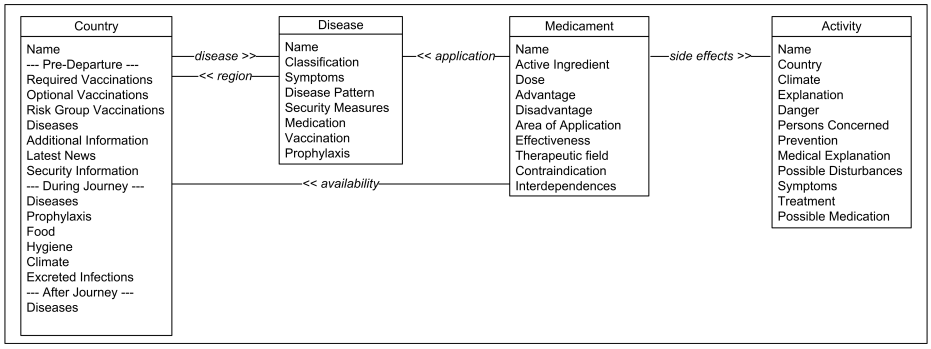

Fig. 2. Case representations for four case bases that provide knowledge for the topic agents

4 Knowledge Modularization Methodology

In section 3 we described how to implement the knowledge modularization for the travel medicine domain. In this showcase we explained how to proceed in order to develop CBR systems which will be deployed in the SEASALT architecture. In this section we will discuss a general methodology how to modularize a domain for setting up a knowledge line.

For the development of CBR systems within SEASALT we use snippet descriptions[3] for describing pivotal factors (or knowledge) as topic agents. Like in traditional CBR systems each snippet description requires its own representation including their knowledge containers. With regard to decentralization and distribution of the content (i.e. the snippets) the Knowledge Engineer has to define the topics, the CBR system's specifications and the linkage between topics. Further on it has to be ensured that a composition of several snippets, which are still meaningful, can be achieved.

The goal of this section is not to describe how to design a CBR system, it presents a methodology for the development of distributed CBR systems, which are based on the SEASALT architecture. In contrast to INRECA and DISER we are not aiming at providing a complete methodology that can directly be implemented in an enterprise. More precisely we show how to analyze an application domain and identify topics and snippet descriptions.

The overall process can be segmented in subprocesses, which is a sequence of its own, will also be discussed in detail. Depending on the application domain and the available information, each subprocesses can be revisited if necessary. The overall process is pictured in Figure 3. For the representation of the processes,

[3] In the remaining parts of this work we will differentiate between *snippet descriptions* and *snippet*. Snippet descriptions are the conceptual representation of knowledge and are equivalent to a case representation. Snippets on the other hand are instances of snippet descriptions and therewith equivalent to cases.

we used the specification language BPMN[4], which has been modified in order to fit our purpose.

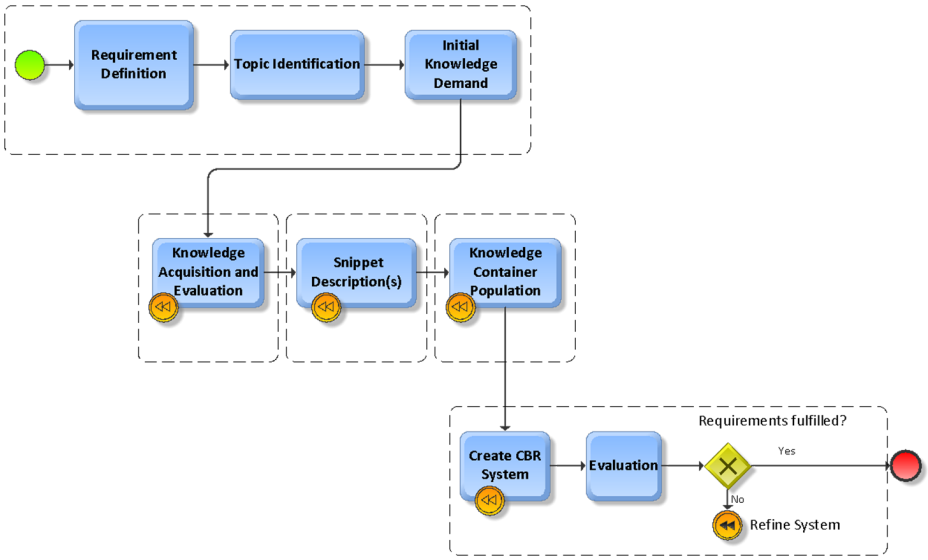

Fig. 3. Abstract Knowledge Modularization process

The development process starts with the identification of the expectations the stakeholders have in order to derive key aspects that have to be covered by the software system. Based on the available knowledge and the insights obtained, the domain has to be separated to determine the snippet descriptions and their associated knowledge sources (from which the snippets/cases are generated). Once the snippet descriptions, snippet sources and interactions are designed, the CBR system can be implemented, evaluated, and, if necessary, incrementally improved.

4.1 Requirement Specification

This introductory phase of the system development is activated by the demand of creating a new knowledge-based system (e.g. by a customer). Furthermore, this step will be the basis for all further developments. The tasks that have to be fulfilled are the determination of the requirements from which the topics for the knowledge distribution can be derived in order to define the required knowledge (see Figure 4).

The Knowledge Engineer should carry out a goal-oriented development and therefore a thoroughly requirements acquisition has to be carried out. Together

[4] `http://www.bpmn.org/`

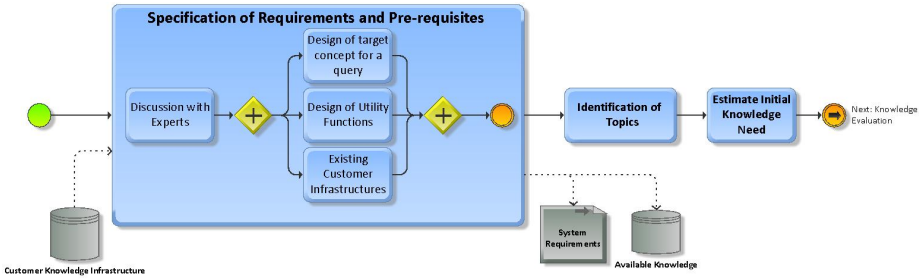

Fig. 4. Requirement Specification Process

with stakeholders, first goals should be identified, and based on them the requirements are iteratively refined. Relevant aspects in this phase are demanded and desired functionalities. Further, also standard factors such as numbers of expected users or the expected access.

In the end of this phase a common concepts describing the system should be available. This must contain the specification of a query and the expected prototypical result as well as information from which topic agent each sub-result should be retrieved. The latter briefly defines the required snippet descriptions, which will be recalled later.

In parallel information sources already available have to be identified and tested whether they can be included in the new system. Within an enterprise often databases or data warehouses can be accessed. If this is possible, the Knowledge Engineer has to ensure that the service will be available, required information is accessible and the provenance of information can be trusted (as well as the stakeholders trust them). For relevant information sources that will potentially feed data in a system, it has to be specified how they are structured and stored. The result of this task is a picture of the complete available knowledge.

Based on the available knowledge topics have to be identified in order to make use of the advantages of the decentralized CBR systems defined in the Knowledge Line. Each case base covers a heterogeneous topic. The definition of topics represents the main thematic areas and are directly related to the information/expectation about the system provided by stakeholders or initiators.

For each topic the required knowledge has to be estimated, in order to fulfill its task within the Knowledge Line. This estimation focuses on semantic estimations rather than the volume. This might lead to a constellation that topics have to be covered, where only little information is available. This will lead into a status of *increased knowledge demand* for this particular topic. On the other hand, if sufficient information is available, this knowledge demand can also be classified as *covered*.

4.2 Knowledge Evaluation and Acquisition

Following the initial phase of collecting system requirements and elaborating available knowledge, in this phase an evaluation of that knowledge is carried out. The result of the evaluation will lead into further processes of more goal oriented knowledge acquisition for certain topics. After adding new knowledge (bases) this phase has to be carried out again in order to either identify the need of further investigation or carrying on to the next phase. The repetition of this phase is carried out until sufficient knowledge is available. Afterwards the snippet descriptions are defined.

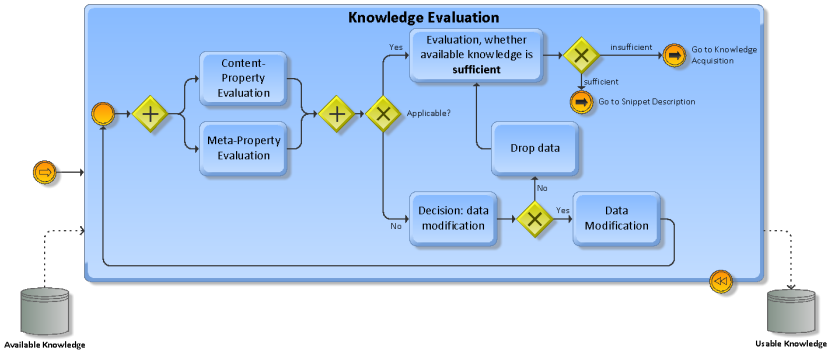

Fig. 5. Evaluation of available knowledge

The knowledge evaluation task (see Figure 5) is carried out as testing and validating of the available knowledge until the previously defined goals are met. In the subsequent runs of this phase, only the newly added knowledge, which originates from the knowledge acquisition, has to be evaluated and existing knowledge does not necessarily have to be evaluated twice. More importantly, it has to be ensure by the evaluation that the newly added knowledge extends the existing knowledge and whether the required knowledge is now available.

During the evaluation it has to be ensured that the containing knowledge covers the topic adequately as well as the content is correct and up-to-date (Content-Properties). Furthermore access possibilities and restrictions as well as the used data representations have to be captured (Meta-Properties). If the evaluation results in the fact that the newly included knowledge does not fit for the topic, it has to be decided whether the new knowledge sources can/should be discarded or the knowledge has to be enhanced.

The evaluation ends with the decision whether the knowledge demand for the overall system is covered by the available knowledge sources. If there is still a lack of knowledge, the next step has to be the knowledge acquisition (see Figure 6). If everything required is covered, it is procceded with the identification and definition of Snippet Descriptions (see Section 4.3).

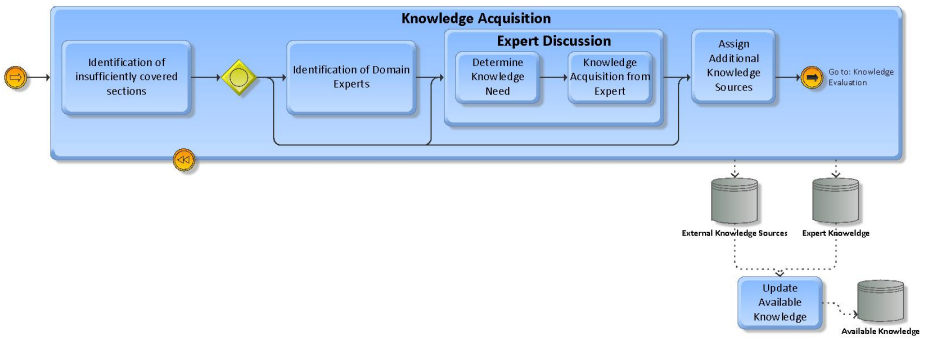

Fig. 6. Knowledge Acquisition

While the knowledge evaluation basically only reviews the knowledge in order to detect knowledge lacks, the knowledge acquisition identifies in which topic what kind of knowledge is missing. After that identification the according expert – if available – has to be consulted. If such an expert is not available s/he has to be found or the topic can not be covered as area of expertise. In this case it is still possible to include the available information, but marked with less confidence.

If an expert has been identified, the expert will be in charge to define the lack of knowledge. During this discussion, knowledge can be directly acquired. This can be procedural or contextual knowledge that clarifies relations and enables the knowledge engineer to enrich the existing knowledge. Furthermore this discussion with the expert can lead to novel knowledge sources. Usually it is not expected that an expert will formalize and insert the required knowledge. Within SEASALT, we would expect the Knowledge Engineer to carry out this task and have an expert reviewing the knowledge as well as the final results.

The result of this subprocess is accumulated expert knowledge and knowledge sources, which will enrich the existing knowledge containers. After this step this knowledge is re-evaluated in the overall context in order to ensure that the knowledge still matches the scope of the overall system and the interaction between topic agents will be still possible.

4.3 Identification and Definition of Snippet Descriptions

Once the previous cyclic processes of knowledge evaluation and acquisition end up with the fact that the available knowledge is sufficient to represent the identified topics, relevant snippet descriptions are defined. For each snippet its representation has to be determined and based on that its correctness.

The identification of snippet description can be differentiated between descriptions derived explicitly from the system requirements and descriptions implicitly described in the available domain knowledge. Snippet descriptions which are based on the system requirements usually cover desired functionalities or topics. These functionalities are incomplete and focus only on particular aspects

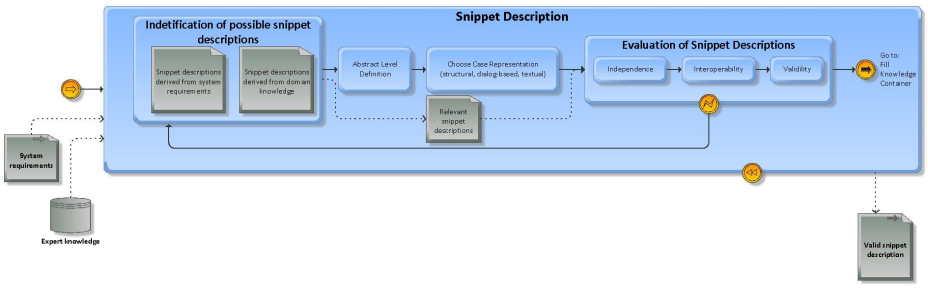

Fig. 7. Identification Snippet Descriptions

that, from our experience, match the expert's area of work/expertise. Especially when topics are combined and the combination possibilities are discussed, more topics arise. However, each topic that comes up does not necessarily need to be represented as such in the knowledge line. The knowledge engineer has to decide whether this aspect is relevant (and therewith becomes an additional topic), it can be merged into an existing topic, or has to be withdrawn. In the end there will be a set of relevant snippet descriptions, which will the be classified as *relevant for the domain* or *additional information*. This process can therewith change the demands of knowledge for each topic, but also helps to reduce the effort, if topics are identified, which are not that relevant or can be covered differently. On the other hand, this process can also be applied if an extension of the functionalities is necessary, because it defines the demand of knowledge and the knowledge acquisition is later on carried out by the standardized process.

For each of the included snippet descriptions, the level of abstraction has to be defined. According to [7], there are three levels of abstraction for cases: *Concrete Cases* represent the real cases with very less loss of information and therewith are the lowest level of abstraction. *Abstract Cases* are reduced in their complexity. For abstract cases the loss of information is immanent and the degree of abstraction can vary as well as concrete cases and abstract cases can create a hierarchy. *Generalized Cases* represent a collection of cases with common features. Based on the level of abstraction, the kind of case representation will be defined.

After finding appropriate level of abstraction and case representation, the correctness of the therewith defined snippet descriptions has to be ensured. Each snippet descriptions has to fulfill the following domain-dependent criteria: independence, interoperability and validity.

The *independence* criterion requires that each snippet description is an individual, semantically coherent unit. This means each snippet description should cover a topic that would also work for itself and therewith each snippet also provides a logic piece of information and does not necessarily need the complete set of snippets to be semantically understandable. Further on, the *interoperability* ensures that snippets can be combined with others while the links or dependencies describe the kind of combination [3]. It is obligatory that each snippet

description has to be linked at least once to another description in order to be included in the knowledge line. The interoperability herewith describes the semantic relationships between topics. The final criterion, *validity*, eventually monitors that snippets are technically combinable – it checks common representations, identifiers and constraints. To satisfy this criterion it has to be ensures that combination procedures are available and well defined.

In case one of the three above mentioned criteria is not met, either the missing information has to be added or the snippet description has to be redefined. If all requirements are satisfied, the definition of snippet descriptions is finished and the collection of valid snippet descriptions is available for the next phase - the implementation.

4.4 System Implementation

This phase first defines how the future system is populated with the acquired knowledge, before the CBR system itself is implemented.

For each of the previously defined snippet descriptions one or more knowledge sources have to be assigned for the case base population. Depending on the domain, the available snippets, the Knowledge Engineer has to decide how many sources are assigned to which snippet description and how much overlap of snippets is allowed. Furthermore, in order to be able to combine information, it is necessary to include certain information in more than one snippet description - the deployment of these information is also defined and implemented within this phase.

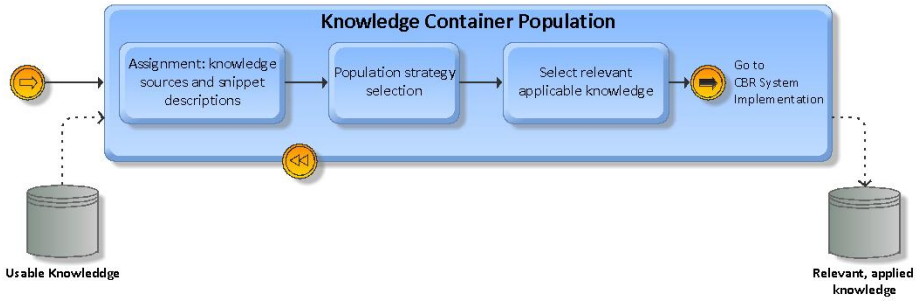

Fig. 8. Knowledge Container Population

Next, the technical procedures for the population have to be defined. This also specifies whether the knowledge to be included is stored locally in the CBR systems or queried on demand. This highly depends on the given infrastructure and availability of knowledge sources in the particular domain.

The technical procedures closely relate to the properties of a knowledge source, which have been determined in the knowledge evaluation phase. These

properties define whether and automated population is possible and how data updates are feasible. Since each snippet description is independent this has to be specified for each individual snippet description and includes access mechanisms and protocols.

After the technical specifications have been defined, the content-based selection has to follow. Not all usable information of the knowledge sources are usually relevant for a snippet. Therewith, the goal is to determine that the content of a particular snippet description fits the expectations and can be applied.

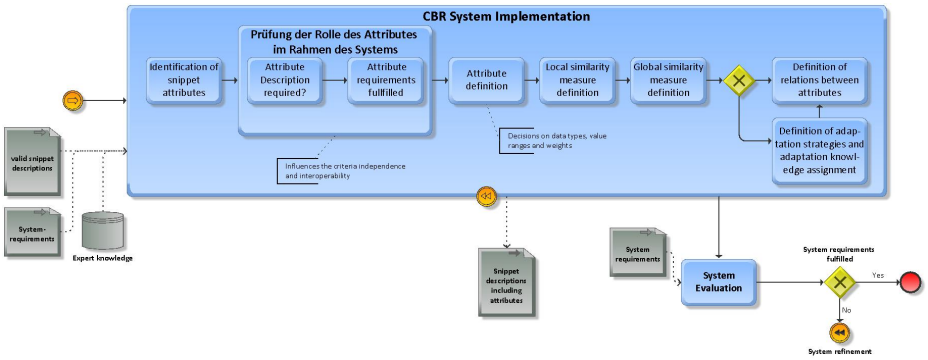

Fig. 9. Implementation of the CBR system

Once all relevant sources are known, accessible and the target representations are linked the implementation and population of the CBR system can start. In the beginning the attribute descriptions for each snippet have to be identified in order to create a case representation. Based on the system requirements, the knowledge engineer has to decide which are the required attribute descriptions. This decision should be based on the independence criteria described in 4.3 and aim at only including necessary attributes.

After all relevant attribute descriptions are sorted out, the knowledge engineer has to analyze whether the case description is sufficient for its purpose within the application. The analysis ensures that the attribute descriptions as as modular as possible and required and furthermore the attribute values can be combined during the knowledge composition. In this process, also the data types and value ranges are defined as well as the local similarity measure(s).

We assume that we have amalgamation functions to describe the global similarity. Therefore the relevance of each attribute description for the case (or snippet) has to be determined and assigned. In the final step optional adaptation mechanisms are defined. In general, the previously mentioned steps describe the process of building a CBR system once all required information is known and available.

The closing step of this process is the evaluation of the created CBR system regarding the requirements. If the demands set for the system are met, the mod-

ularizing phase is completed. If there are mismatches between the expectations and the performance, the system has to be revised.

There are various rescue points where the revision can start. First the type of mismatch has to be identified, before the process can move to the according task. If there are general drawbacks regarding the incoming data, the knowledge evaluation and acquisition has to be refined (see section 4.2). Alternatively, decisions made regarding the design of snippet descriptions and population processes can be revised which can influence the type of information produced by the overall system. Eventually, also the technical implementation can cause mismatches to the expectations and therewith the implementation might has to be revised as well. A detailed study of the Knowledge Line modularization methodology in a different domain can be found in [11].

5 Software Engineering of the Knowledge Line

While the Knowledge Line describes a process of how to collect and organize knowledge, it requires tools to capture and implement it in order to use it within an intelligent system. For the entire process we have successfully used myCBR [5,16], which allows to model the knowledge as well as to test the case-based retrieval.

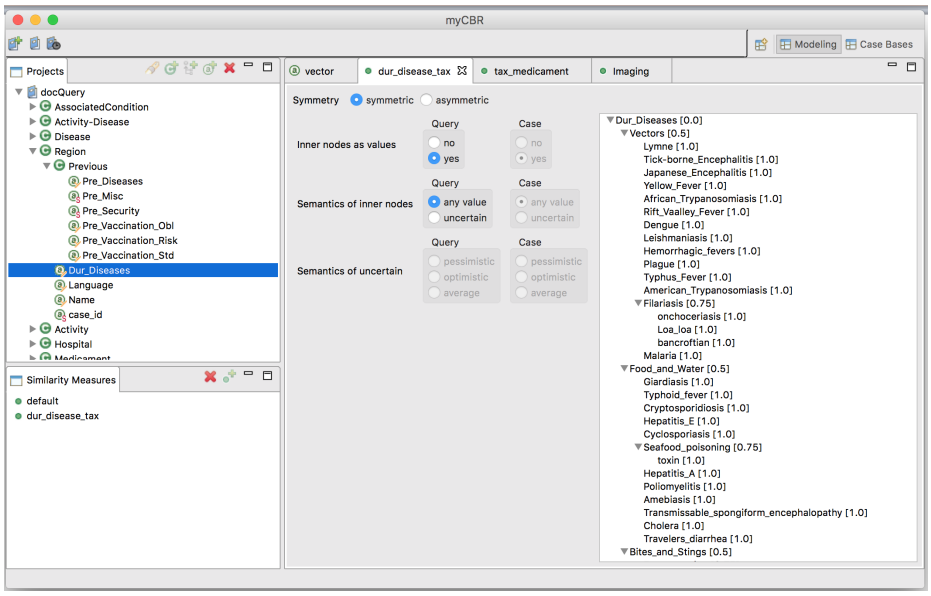

Fig. 10. myCBR model view showing the different case bases on the upper left, the similarity measures for one attribute on the lower left and on example taxonomy in the main screen

Figure 10 shows the myCBR view that is used to create the knowledge model for case bases. The tool supports the creation of various concepts of which each of them can hold individual case bases. Therewith it is possible to visualize the included knowledge structure along with the similarity measures during discussions with experts. Further on, the tool also allows to carry out the similarity based retrieval for each concept and therewith directly evaluate how changes made to the knowledge model and especially similarity measures affect the retrieval result. Also various case bases can be created and kept in parallel, which allows different testing scopes.

Once the CBR system is designed, the resulting project can be integrated in any kind of Java application. The myCBR back end comes as a jar file and together with the project file the entire CBR system can be run independently from the workbench.

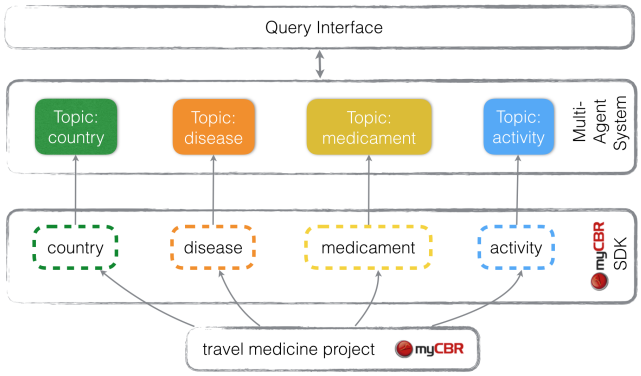

Fig. 11. myCBR-based architecture of a Knowledge Line implementation in the travel medicine domain

Figure 11 shows a possible architecture of a myCBR-powered Knowledge Line application. We assume that we have one project that contains all knowledge models and case bases, each topic agent can individually be instantiated and represented by a topic agent. Numerous topic agents are coordinated by a multi-agent system (MAS) that collaboratively compose a solution. The dependencies that guide the knowledge composition process are implemented as an MAS und JADE (see [15] for details).

On top of the MAS, we suggest a query interface that receives and organizes the query process. The query interface typically has knowledge of the case representations within the MAS and can therewith provide queries for the CBR system in the right structure, while the MAS manages the internal case base dependencies. The query interface can be implemented using the Spring frame-

work [5] which sends queries to the MAS and exposes a RESTful web API which can be used by various types of front ends.

Further on, the population of case bases for each topic can be implemented individually, so each case base creation can follow a customized process of accessing external knowledge sources, apply information extraction and natural language processing, if necessary, for feeding in knowledge snippets.

6 Summary

In this chapter we present an knowledge engineering approach that describes how a complex domain with heterogeneous knowledge components can be systematically transferred into a distributed CBR system. The resulting CBR system is a multi-agent system that utilizes several homogeneous CBR engines to hold and provide knowledge on demand. We describe the conceptual process of identifying, organizing and implementing such a knowledge-based system.

Eventually we give an overview how the concept can be implemented using existing open source tools and frameworks. To illustrate the entire process we use the travel medicine domain and discuss the challenges it provides.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications, 7(1), 39–59 (1994)
2. Althoff, K.D., Reichle, M., Bach, K., Hanft, A., Newo, R.: Agent based maintenance for modularised case bases in collaborative multi-expert systems. In: Proceedings of AI2007, 12th UK Workshop on Case-Based Reasoning. pp. 7–18 (Dezember 2007)
3. Althoff, K.D., Reichle, M., Bach, K., Hanft, A., Newo, R.: Agent based maintenance for modularised case bases in collaborative multi-expert systems. In: Proceedings of AI2007, 12th UK Workshop on Case-Based Reasoning. pp. 7–18 (dec 2007)
4. Bach, K.: Knowledge Acquisition for Case-Based Reasoning Systems. Ph.D. thesis, University of Hildesheim, München (2012), iSBN 978-3-8439-1357-
5. Bach, K., Althoff, K.D., Newo, R., Stahl, A.: A case-based reasoning approach for providing machine diagnosis from service reports. In: Ram, A., Wiratunga, N. (eds.) Proc. of the 19th Intl. Conference on Case-Based Reasoning (ICCBR-2011), London, UK. LNCS, vol. 6880, pp. 363–377. Springer Verlag, Heidelberg (Sep 2011)
6. Bach, K., Reuss, P., Althoff, K.D.: Case-based menu creation as an example of individualized experience management. In: Maier, R., Kohlegger, M. (eds.) Professional Knowledge Management. Conference on Professional Knowledge Management (WM-2011), From Knowledge to Action. pp. 194–203. LNI 182, Köllen Druck & Verlag GmbH, Bonn (Mar 2011)
7. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, Lecture Notes in Computer Science, vol. 2432. Springer (2002)

[5] `https://spring.io/`

8. Bergmann, R., Althoff, K.D., Breen, S., Göker, M.H., Manago, M., Traphöner, R., Wess, S.: Developing Industrial Case-Based Reasoning Applications: The INRECA-Methodology, LNCS, vol. 1612, chap. Selected Applications of the Structural Case-Based Reasoning Approach, pp. 35–70. Springer (2003)
9. Davenport, T.H., Prusak, L.: Working Knowledge: How Organizations Manage What they Know. Harvard Business School Press (May 2000)
10. Ihle, N., Newo, R., Hanft, A., Bach, K., Reichle, M.: Cookiis - A Case-Based Recipe Advisor. In: Delany, S.J. (ed.) Workshop Proceedings of the 8th International Conference on Case-Based Reasoning. pp. 269–278. Seattle, WA, USA (July 2009)
11. Marter, S.: Case based coordination agents - knowledge modularization and knowledge composition (Fallbasierte Koordinationsagenten – Wissensmodularisierung und Wissenskomposition für dezentrale, heterogene Fallbasen). Master's thesis, Institute of Computer Science, University of Hildesheim (2011)
12. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM 15, 1053–1058 (December 1972)
13. Redmond, M.: Distributed cases for case-based reasoning: Facilitating use of multiple cases. In: AAAI. pp. 304–309 (1990)
14. Reichle, M., Bach, K., Althoff, K.D.: Knowledge Engineering within the Application Independent Architecture SEASALT. In: Baumeister, J., Nalepa, G.J. (eds.) Int. J. Knowledge Engineering and Data Mining, pp. 202 – 215. Inderscience Publishers (Jan 2011)
15. Reuss, P.: Concept and implementation of knowledge line retrieval strategies for modularized, homogeneous topic agents within a multi-agent-system (konzept und implementierung einer knowledge line – retrievalstrategien fuer modularisierte, homogene topic-agenten innerhalb eines multi-agenten-systems). Hildesheim : Stiftung Universität Hildesheim, Institut für Informatik, Bereich Intelligente Informationssysteme, Master thesis (August 2012)
16. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of cbr applications with the open source tool mycbr. In: ECCBR '08: Proceedings of the 9th European conference on Advances in Case-Based Reasoning. pp. 615–629. Springer-Verlag, Berlin, Heidelberg (2008)
17. Tautz, C.: Customizing Software Engineering Experience Management Systems to Organizational Needs. Ph.D. thesis, Universität Kaiserslautern (2000)