CBROnto: A Task/Method Ontology For CBR

Belén Díaz-Agudo and **Pedro A. González-Calero**
Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
{belend,pedro}@sip.ucm.es

Abstract

Our approach to Case-Based Reasoning (CBR) is to build integrated systems that combine case specific knowledge with models of general domain knowledge. In this paper we describe CBROnto, the CBR ontology we have developed, as a task/method ontology. CBROnto specifies a modelling framework to describe reusable CBR Problem Solving Methods based on the CBR tasks they solve and the knowledge requirements needed to apply them.

Introduction

Even though any Case-Based Reasoning (CBR) system relies on a set of previous specific experiences, its reasoning power can be improved through the use of general knowledge about the domain. Our approach to CBR is to build integrated knowledge based systems (KBS) that combine case specific knowledge with models of general domain knowledge. Our ongoing work is the development of COLIBRI (Cases and Ontology Libraries Integration for Building Reasoning Infrastructures), an environment to assist during the design of knowledge intensive CBR (KI-CBR) systems (Díaz & González 2000; 2001).

In knowledge engineering approaches such as Role Limiting Methods (McDermott 1988), CommonKADS (Schreiber *et al.* 1994) or Components of Expertise (Steels 1990), a KBS is viewed as consisting of separate but interconnected collaborating components. Typically, components of a KBS include domain knowledge and Problem Solving Methods (PSMs), that represent commonly occurring, domain-independent problem-solving strategies.

In (Díaz & González 2000) we defended the acquisition of domain knowledge for KI-CBR systems by reusing domain ontologies. In this paper, we consider problem solving knowledge. The core of the COLIBRI architecture is CBROnto, an ontology incorporating common CBR terminology and problem solving knowledge. CBROnto serves as a domain-independent framework to design KI-CBR systems. It is both an ontology including general terminology related to CBR systems, and an ontology of CBR tasks and methods. In this paper we focus on the task/method view of CBROnto. We describe CBROnto method description language, and how CBROnto solves two problems that are typically found when applying PSMs. The first problem is due to the fact that PSMs may use different terminology than the one used in the domain. CBROnto bridges the gap between domain knowledge and PSMs using a mapping mechanism that is based on Description Logics (DLs) concept classification. The second problem refers to PSM knowledge requirements. Again, DLs classification is the mechanism we use to check if the method knowledge requirements are satisfied by the application context – made of the domain knowledge and the case base. This paper begins by introducing DLs, the COLIBRI system and CBROnto. The rest of the paper describes CBROnto's ontology of tasks and methods and its method description capabilities, emphasizing knowledge requirements representation.

Description Logics

DLs based languages are commonly used to implement ontologies, and it is the technology we use to formalize aspects of representation and reasoning. DLs, rooted in the KL-ONE family (Brachman & Schmolze 1985) and the frame systems, are characterized by their expressiveness and clearly defined semantics. We use LOOM (Mac Gregor & Bates 1987), a particular DLs implementation, as the supporting system on top of which COLIBRI is built.

DLs capture the meaning of the data by concentrating on entities (grouped into concepts) related by relationships. More important than the DLs representational characteristics are its reasoning mechanisms. The most important is the checking of incoherencies and the organization of terms (concepts and relations) into taxonomies that the system automatically builds from the term definitions. DLs reasoning mechanisms are based on *subsumption*, to determine whether a description is more general than another, and *instance recognition*, to determine the concepts that an individual satisfies and the relations that a tuple of individuals satisfies. *Contradiction detection*, both for descriptions and assertions about individuals, completes the basic set of reasoning mechanisms provided by DLs systems.

COLIBRI/CBROnto

COLIBRI helps to design KI-CBR systems that combine specific cases with various knowledge types and reasoning methods. The major problem associated with the knowledge intensive approach to CBR is the so called knowledge acquisition bottleneck. Our approach to knowledge acquisition (Díaz & González 2000) is based on reusing knowledge

from an ontology library to create complex, multirelational knowledge structures to support the CBR processes.

As the next step, the KI-CBR system should be able to take advantage of the acquired domain knowledge. COLIBRI views KI-CBR systems as consisting of collaborating knowledge components, and distinguishes different types of knowledge (Van Heijst, Schreiber, & Wielinga 1997). *Ontologies* describe the structure and vocabulary of the *Domain Knowledge* that refers to the actual collection of statements about the domain. *Tasks* correspond to the goals that must be achieved. *PSMs* capture the problem-solving behavior required to perform the goals of a task. And *Inferences* describe the primitive reasoning steps in the problem solving process.

COLIBRI uses CBROnto as a unifying framework that structures and organizes different types of knowledge in KI-CBR systems according to the role that each one plays. CBROnto captures CBR semantically *important* terms, includes CBR dependent but domain-independent terms, and aims to unify case specific and general domain knowledge representational needs.

Mappings through DLs Classification

As PSMs are used to accomplish tasks by applying domain knowledge, the external *context* of a PSM is formed by the task to be solved and the domain knowledge to be applied. When we want to use PSMs to build a KBS, we have to connect the PSMs with both the tasks and the domain knowledge (Gómez & Benjamins 1999). Since PSMs are generic and reusable components, they may not always perfectly fit in a context, or in other words, there may be gaps.

In our model, it is not necessary to consider task-method gaps because tasks and PSMs are defined using CBROnto as unifying terminology. Instead, we take care of gaps between methods and domain knowledge. They exist mainly for two reasons.

The PSMs may use different terminology than that of the domain knowledge in which case a "renaming" process can bridge the gap. When designing a KI-CBR system using COLIBRI, after domain knowledge acquisition, the system designer performs an integration phase based on classifying the domain terms (concepts and relations) with respect to CBROnto terms. Due to the inheritance mechanism only the top level terms in the hierarchies should be classified. This mechanism allows CBROnto's methods to capture the problem-solving behavior in a domain-independent manner, referring only to CBROnto terms that are correspondingly linked to the domain knowledge terms by classification.

McDermott (McDermott 1988) coined the term *knowledge roles* to refer to the way in which problem solving knowledge requires domain knowledge of certain types. PSMs represent different strategies to solve a task, and these strategies determine the roles that domain-dependent knowledge plays. Our approach relies on the use of CBROnto terminology and DLs classification to "type" knowledge elements according to their role in the CBR methods. The roles that domain knowledge terms play in the CBR methods depend on how they are classified below CBROnto terms, i.e., classification integrates the acquired domain knowledge

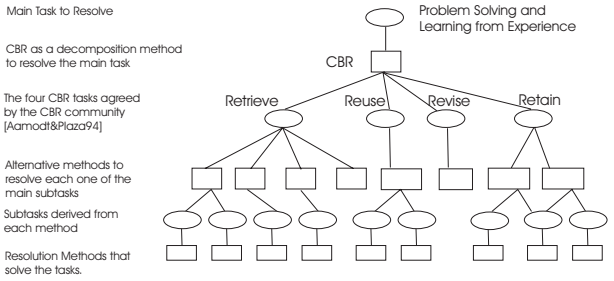

Figure 1: CBR Task Method Structure (Chandrasekaran 1990)

with the task/method knowledge of CBROnto and it defines the role that domain knowledge plays in the PSMs.

The second reason for the domain-method gap is due to the fact that knowledge required by a PSM may not be fully given by the available domain knowledge, in which case additional knowledge needs to be acquired. We describe in this paper how CBROnto method applicability in a certain context can be checked thanks to the described integration phase together with a declarative description of contexts of application and the knowledge requirements of PSMs.

Issues involved in reusing PSMs from a library include finding a suitable method and checking whether it is applicable in the current context. In the CBROnto library of methods, selection is simple because methods are organized around the tasks they resolve. That is why in this paper we are looking primarily at method applicability checking. Next section describes CBROnto task/method ontology and its method description language. We especially focus on the representation of the knowledge requirements for the PSMs.

CBROnto as a Task and Method Ontology

A useful way to describe problem solving behavior is in terms of the tasks to be solved, the goals to be achieved, the methods that will accomplish those tasks, and the domain knowledge that those methods need. A description along these lines is referred to as a knowledge level description. Although various authors have applied knowledge level analysis to CBR systems, the most relevant work is the CBR task structure developed in (Aamodt & Plaza 1994) influenced by the Components of Expertise Methodology (Steels 1990). At the highest level of generality, they describe the general CBR cycle in terms of four tasks: *Retrieve* the most similar case/s, *Reuse* its/their knowledge to solve the problem, *Revise* the proposed solution and *Retain* the experience. Each one of the four CBR tasks involves a number of more specific sub-tasks. There are methods to solve tasks either by decomposing a task in subtasks or by solving it directly. In our approach, we do not identify a common subtask structure associated to all the methods. Instead we use the notion of a task structure proposed in (Chandrasekaran 1990). As shown in Figure 1, the task structure identifies a number of alternative methods for a task, and each one of the methods sets up different subtasks in its turn. This kind of task-method-subtask analysis is carried on to a level of detail where the tasks are primitive with respect to the available knowledge.

PSMs capture and describe problem-solving behavior in an implementation and domain-independent manner. CBROnto includes a library of PSMs associated to the main CBR tasks. CBROnto describes CBR PSMs by relating them to terms and relations within its ontology of tasks, methods and domain characteristics. The method ontology includes terms of the method description language that are used to formalize PSMs. It defines concepts and relationships that are used by the methods. CBROnto also includes a simple task ontology influenced by Aamodt and Plaza's task structure that defines the terminology related to the CBR tasks from a domain and method independent point of view. Methods in our library are organized around the tasks they resolve.

Our approach relates to other systems like HICAP (Muñoz-Avila, Aha, & Breslow 1999) that solves problems with HTNs (Hierarchical Task Networks) retrieving methods for decomposing tasks into subtasks. The main difference is that HICAP considers methods to solve problem-specific tasks, and cases in the case base are methods themselves that are retrieved in a case based sense. Our task/method structure only refers to CBR tasks and methods (and not problem-specific tasks and methods).

We represent the task-method structure of Figure 1 by using two related concept hierarchies. The task hierarchy is rooted by the CBR_TASK concept that has only one direct instance, iCBR_task, representing the generic task of "solve a problem and learn from the experience". The method used to solve this task is CBR itself, that is represented by iCBR_method, a direct instance of CBR_METHOD that is the root concept in the method hierarchy . There are four subconcepts of the CBR_TASK concept regarding the four basic CBR tasks: RETRIEVE_TASK, REUSE_TASK, REVISE_TASK and RETAIN_TASK. Each one of these concepts represents a subtask that results from the application of the decomposition method iCBR_method to solve iCBR_task.

Depending on the method applied to solve a task, it results on a different set of subtasks to be solved themselves. In the task taxonomy, below the concepts corresponding to the four basic CBR tasks, there are concepts for all the subtasks resulting from the different methods. For example, below RETRIEVE_TASK we find concepts for all the subtasks resulting from all the retrieval methods. Each one of the methods represents the information about the subtasks applied for it.

As a corresponding taxonomy, below the CBR_METHOD concept there is a subconcept hierarchy classifying different types of methods regarding the task they solve. The two hierarchies allow the representation of certain ontological assumptions in CBROnto. For example, instances of RETRIEVAL_METHODS (either direct or not) relate with instances of RETRIEVE_TASK. As COLIBRI helps designing KI-CBR systems, we have studied and included in CBROnto those methods that intensively take into account the available general domain knowledge. By now, CBROnto library of methods includes several approaches successfully applied and described in the CBR literature. In (Díaz & González 2001) we have described the CBROnto declarative framework that successfully integrates various retrieval methods.

Method Description Language

Most approaches consider that a PSM consists of three related parts. The *competence* is a declarative description of *what* can be achieved. The *operational specification* describes the reasoning process, i.e *how* the method delivers the specified competence if the required knowledge is provided. And the *requirements* describe the knowledge needed by the PSM to achieve its competence (Gómez & Benjamins 1999).

Some approaches like CommonKADS (Schreiber *et al.* 1994) specify much of how the PSM achieves its goals, i.e. the reasoning steps, the data flows between them, and the control that guides their execution. As we focus on PSM applicability assessment we consider what the method does, i.e. the task it solves, and its knowledge requirements, and leave control-flow issues to informal documentation and method implementation code. This allow us to use a black box type of method reuse.

Our approach to the specification of PSMs competence and requirements makes use of ontologies and provides two main advantages. First, it allows formal specifications that add a precise meaning and enables reasoning support. Second, it provides us with important benefits regarding reuse because task and method ontologies can be shared by different systems. Each method in the library is represented as a CBR_METHOD instance (either direct or not) that relates with:

- The method name.
- The method informal description.
- The instance of CBR_TASK (either direct or not) representing the method competence.
- The instance of REQUIREMENTS representing the method knowledge requirements.
- The instance of INPUT_REQUIREMENTS representing the input requirements that must be satisfied to apply the method.
- The instance of REASONING_TYPE recording successful uses of the method in certain kind of CBR systems.
- The instance of METHOD_FUNCTION representing the Lisp function that implements the method.

DLs reasoning mechanisms allow us to reason with the PSMs formalized in this way. Method internal reasoning processes are not formalized as part of the descriptions used to reason with. Instead, PSM descriptions relate to Lisp functions that implement them. Input to a PSM is a list of instances, representing either values or structured individuals, whose defining concepts reside in the CBROnto method ontology.

As an example, we use one retrieval methods that is particularly well-suited when we have taxonomical domain knowledge available. It uses a representational approach that assigns similarity meaning to the path joining two individuals in the case organization structure. Retrieval is then accomplished by traversing the subsumption links starting from the query, whose position at the hierarchy has been recognized by the DLs mechanisms. This retrieval method is represented in CBROnto method description language as:

```
(tell (:about iRetrieve_instance_classification_method
        Instance_Classification_Method Decomposition_Method
        (method_name "instance classification")
        (method_informal_description "This method retrieve
instances according to the distance in the representational structure ")
        (method_competence iRetrieveTask)
        (method_requirements iInstanceRequirements)
        (method_io iInstanceIO)
        (functional_specification iInstanceFunction)
        (subtask iClassifyInstance)
        (subtask iRetrieveSiblings))
```

Method Requirements Representation

PSMs are defined in a domain independent way although they include certain knowledge requirements determining their applicability in a particular context. We propose a declarative approach to describe method knowledge requirements, that allows to check the applicability of a method in a certain context.

Below the CBROnto REQUIREMENTS concept there is a taxonomy representing the knowledge requirements for each type of method. These requirements represent situations in which this type of methods are applicable. Each method individual is related with an instance of one of the REQUIREMENTS subconcepts of the hierarchy according to the corresponding ontological assumptions.

Requirements descriptions include information about:

- The domain knowledge model. For example, depth and width properties of the concept and relation taxonomies are relevant to apply the retrieval methods using classification. (Díaz & González 2001)
- The case base size. For example, the number of cases is relevant to assess the efficient use of the retrieval method by similarity computation.
- The case types. For example, to apply an adaptation method there must be cases with solution.
- The knowledge roles that "type" domain knowledge terms according to their classification below CBROnto terms. For example, a term plays the "goal" role in the retrieval method by goal matching (Díaz & González 2001) if it is classified below the GOAL CBROnto concept.
- Other CBROnto related knowledge, for example, instances of the SIMILARITY MEASURE concept determine the applicability of the retrieval method by similarity computation, instances of the RELEVANCECRITERIA concept determine the applicability of the retrieval method by relevance criteria (Díaz & González 2001), the adaptation cases (instances of the ADAPTATIONCASE concept) determine the applicability of the case-based adaptation method, etc.

CBR Systems Development with COLIBRI

Method competence is the task the method solves. In a particular KI-CBR system, several alternative methods can be applied to solve each task. In our model, each task can be linked with a preferred method instance. Task-method relation links CBR_TASK instances with CBR_METHOD instances maintaining CBROnto ontological assumptions. If a CBR system designer fixes a preferred method to solve the task, then a task_method link is asserted between the corresponding individuals. Figure 2 shows the retrieval task

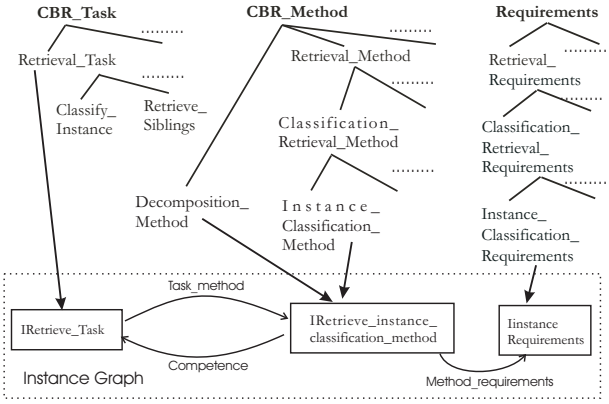

Figure 2: task-method and requirements

(individual iRetrieve_task) when the chosen method is iRetrieve_Instance_Classification_Method. This representation allows an automatic, general and recursive task resolution mechanism.

The task solver checks if the task has a task_method relation and if it does, the reached individual is used as the method to solve the task. If it does not, the solver searches for methods that solve the task using the method_competence relation. To select among several applicable methods for a task we use the mechanism that is described in next section. Decomposition methods divide the task in subtasks and the resolution process is applied recursively for each subtask. Resolution methods finalize recursion and solve the task:

Resolve (iT)
1.Get the method individual to resolve the task: iM
2.Get the method functional specification iEF
4. If iM is a decomposition_method,
  Apply iEF to get the sequence of subtaks
  to be solved: iST_1 , ..., iST_n
  ResolveSeq(iST_n , ResolveSeq(iST_{n-1} , ... Resolve(iST_1)...))
Else *% iM is a resolution method*
  Apply iEF to solve the task iT

Selecting Applicable Methods for a Task

The system designer chooses the methods to solve the different tasks using an interactive process. When a task does not have a preferred method, the task solver obtains all the methods whose competence subsumes the task and checks their applicability, i.e. if their knowledge requirements are fulfilled by the current context. Afterwards, the system designer chooses one between the applicable methods.

To assess method applicability, the current context is characterized by a concept that is automatically constructed, and that describes the available domain knowledge and the case base. This description is classified in the subsumption taxonomy to test if the requirements concept, associated to the method, subsumes it or not. The method is applicable when the method requirements concept subsumes the context concept. Besides, if a method is not applicable, this mechanism makes possible to explain why the method does not fit the situation and what additional knowledge would be needed to apply the method.

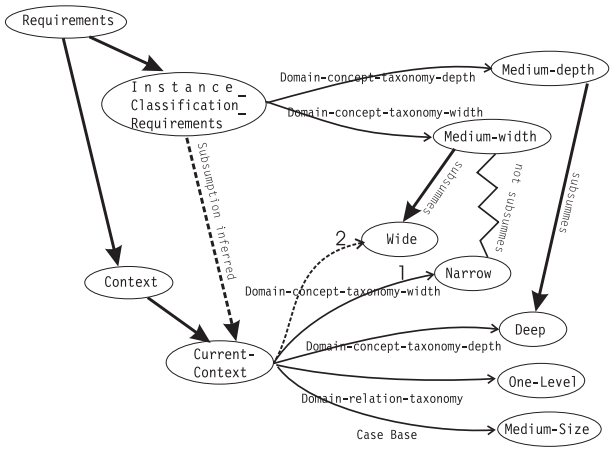

Figure 3: Checking method applicability

For example, the concept CURRENT_CONTEXT of Figure 3, represents a situation where there are a DEEP but NARROW domain concept taxonomy, a relation taxonomy with only ONE LEVEL and a case base of MEDIUM SIZE. We know that the retrieval method by instance classification behaves adequately when the domain concept taxonomy has enough concepts, i.e the domain concept taxonomy depth and width are at least of medium level.

In this situation the method is not applicable because, the MEDIUM-WIDTH concept does not subsume NARROW, and then the requirements concept does not subsume CURRENT_CONTEXT. The declarative representation allows the system to compare the two concepts and find the dissimilarity between them to determine what additional knowledge must be provided to make the knowledge usable by the method, either by adding new knowledge (as in this example) or classifying it below CBROnto terms adequately. CBROnto relies on the LOOM classifier to reason about what knowledge requirements subsume the current context representation. After adding new concepts to increase the concept taxonomy width, that becomes WIDE now, figure 3 shows that the subsumption mechanism infers (dotted line) that the method is applicable in the current context, because its requirements concept subsumes the current context concept.

Conclusions

In this paper we have described CBROnto as an ontology that includes task and method knowledge about CBR. CBROnto language to describe methods is used to formalize the CBR PSMs, that are organized in a library around the tasks they resolve. We have made an special point of the representation of the method knowledge requirements, and have described how DLs mechanisms allow reasoning with PSM descriptions to check their applicability regarding an external context formed by the domain knowledge and the cases. Besides, DLs mechanisms make it possible to explain why the method does not fit the situation and what additional knowledge would be needed to apply the method.

Even though DLs mechanisms have proven to be useful in CBR systems, they cannot do much without a good model of the domain on which to set the mechanisms to work. Our approach uses DLs mechanisms within domain ontologies and CBROnto as a CBR task/method ontology. To bridge the gap between domain knowledge and methods, we have described a mapping mechanism based on DLs classification and inheritance that allows PSMs to be described using method terminology from CBROnto in a domain-independent manner, although they intensively take into account the domain knowledge.

Although the process of developing COLIBRI and CBROnto is not finished, we have included by now the main design ideas behind its architecture. Next step is testing its use in real CBR applications. Our current work is studying the applicability of COLIBRI/CBROnto in a CBR system to generate Spanish poetry.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 7(i).

Brachman, R. J., and Schmolze, J. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9(2):171–216.

Chandrasekaran, B. 1990. Design problem solving: A task analysis. *AI Magazine* 11:59–71.

Díaz, B., and González, P. 2000. An architecture for knowledge intensive CBR systems. In *Advances in Case-Based Reasoning – (EWCBR'00)*. Springer-Verlag.

Díaz, B., and González, P. 2001. A declarative similarity framework for knowledge intensive CBR. In *Procs. of the International Conference on CBR (ICCBR'01)*. Springer.

Gómez, A., and Benjamins, R. 1999. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *IJCAI99 workshop on Ontologies and Problem-Solving Methods, Sweden*.

Mac Gregor, R., and Bates, R. 1987. The LOOM knowledge representation language. ISI Reprint Series ISI/RS-87-188, University of Southern California.

McDermott, J. 1988. Preliminary steps towards a taxonomy of problem-solving methods. In Marcus, S., ed., *Automating Knowledge Acquisition for KBS*. Kluwer.

Muñoz-Avila, H.; Aha, D. W.; and Breslow, L. 1999. HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Orlando, FL: AAAI Press.*, 870–875.

Schreiber, T.; Wielinga, B. J.; Akkermans, J. M.; Van de Velde, W.; and de Hoog, R. 1994. CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert* 9(6).

Steels, L. 1990. Components of expertise. *AI Magazine* 11(2):29–49.

Van Heijst, G.; Schreiber, A.; and Wielinga, B. 1997. Using explicit ontologies in knowledge based systems development. *International Journal of Human and Computer Studies* 46(2/3).