An Architecture for Knowledge Intensive CBR Systems*

Belén Díaz-Agudo and Pedro A. González-Calero

Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
{belend, pedro}@sip.ucm.es

Abstract. In this paper we describe a domain independent architecture to help in the design of knowledge intensive CBR systems. It is based on the knowledge incorporation from a library of application-independent ontologies and the use of an ontology with the common CBR terminology that guides the case representation and allows the description of flexible, generic and homogeneous CBR processes based on classification.

1 Introduction

Any knowledge-based system (KBS) achieves its reasoning power through the explicit representation and use of different kinds of knowledge about a certain domain. Although in a CBR system the main source of knowledge is the set of previous experiences, our approach to CBR is towards integrated applications that combine case specific knowledge with models of general domain knowledge, mainly about the domain terminology. The more knowledge is embedded into the system, the more effective is expected to be. The major problem associated with this knowledge intensive CBR approach is the so called knowledge acquisition bottleneck (common for every KBS).

We make an issue of domain knowledge acquisition and study how the ontological engineering community efforts could help us to acquire the knowledge needed in a knowledge intensive CBR application. The goal of our current work is to formalize this approach to CBR and provide a tool and a methodology to assist during the design phase of CBR application development. Our main contribution is the definition of a domain-independent architecture to help in the integration of ontologies for CBR applications. The core of this architecture is CBROnto, an ontology incorporating the common CBR terminology that is used to guide the domain ontologies integration, and that will be the base of a future system to support the design of knowledge intensive CBR applications. This paper discusses the issues involved, but the whole system implementation is far from its final form.

Section 2 introduces the ontological reuse we propose for the CBR knowledge acquisition process and Section 3 describes CBROnto, the CBR ontology we have

* Supported by the Spanish Committee of Science & Technology (CICYT TIC98-0733)

E. Blanzieri and L. Portinale (Eds.): EWCBR 2000, LNAI 1898, pp. 37–48, 2000.

developed. In Section 4 we sketch the main ideas of the CBR processes defined within our architecture. Finally the conclusions, advantages and shortcomings of the current framework are discussed.

2 The Synergy of the Ontological and CBR Communities

The word *ontology* is used as a technical term by different groups to mean slightly different things. The more well-known definition for *ontology* is: "An ontology is a specification of a conceptualization" [8]. To clarify, an ontology:

- Expresses the consensus knowledge of a community of people.
- Defines the basic terms and relations comprising the vocabulary of a topic area, and contains precisely defined terms that can be used to describe and understand more complex descriptions.
- Can be reused and serve as a starting point to construct different knowledge-based applications.

Most of the KBSs (including the KB CBR systems) have some reusable ontological content but it is often influenced by the specific task, the restrictions of the representation language, and the specific inference procedures employed.

We state that ontologies can be useful for designing knowledge intensive CBR applications because they allow the knowledge engineer to use knowledge already acquired, conceptualised and implemented in a formal language, reducing considerably the knowledge acquisition bottleneck. Moreover, the reuse of ontologies from a library also benefits from their reliability and consistency [6].

We know of little interactions among the CBR community and the ontological community although the knowledge in an ontology is specially well-suited to be shared, and many CBR systems codify this kind of domain knowledge. Ontologies may help in the creation of complex, multirelational knowledge structures to support the CBR processes.

2.1 The Ontology Server

The Ontology Server (OS) (see [6] for a complete description and references) is a set of tools and services that support the building of shared ontologies between geographically distributed groups. It was developed in the context of the ARPA Knowledge Sharing Effort by the Knowledge System Laboratory at Stanford University. This server is an extension of the language Ontolingua[1]. The ontology server architecture manages a library of ontologies to be reused and provides a HTML interface to build, modify and browse ontologies; lexical and syntactic analyzers to avoid incompleteness, inconsistencies and redundant knowledge; and a set of translators to various knowledge representation languages as CLIPS, CML RE, EPIKIT, LDL, KIF, LOOM, OKBC, PROLOG. The OS and Ontolingua have been accepted by the knowledge-sharing community as the main tool to implement ontologies mainly due to its complete set of translators.

[1] http:\\www-ksl-svc.stanford.edu:5915\ or http:\\granvia.dia.fi.upm.es:5915\

2.2 Description Logics

Ontologies must be codified in a formal language. Description Logic based languages (DLs) are commonly used to implement ontologies, and it is the technology we use in our model to formalize aspects of representation and reasoning. DLs, rooted in the KL-ONE family [3] and the frame systems, are characterized by its expresiveness and clearly defined semantics. The implementation language we use is LOOM [9], one of the destination languages of the OS translators.

DLs capture the meaning of the data by concentrating on entities (grouped into classes or concepts) related by relationships. This intuition is shared by formalisms such as semantic data models, semantic networks or frame systems. More important than the DLs representational characteristics are its reasoning mechanisms. The most important characteristic is the checking of incoherencies and the organization of the concepts on a taxonomy that the system automatically builds from the concept definitions. This is possible because of the clear and precise semantic of concept definitions that avoid the user to put the concepts in the correct place of the hierarchy (as is the case in frame systems, which provide inheritance but not classification).

DLs reasoning mechanisms and deductive inferences are based on *subsumption* and *instance recognition*. Subsumption determines if a term is or not more general than another, and instance recognition finds all the concepts that an individual satisfies. Furthermore, completion mechanisms perform logical consequences like inheritance, combination of restrictions, restriction propagation, contradiction detection, and incoherent term detection. These mechanisms will be used during the ontology integration, the case representation and, in general, as the base for all the CBR processes.

2.3 How the Ontologies Are Used

We are interested in three types of ontologies [6] from those provided by the OS: (1) the *domain ontologies* provide the vocabulary for describing a domain and interpreting a description of a problem in that domain; (2) the *task ontologies* provide the vocabulary for describing terms involved in the problem-solving processes, which could be attached to similar tasks that may, or may not, be in the same domain; and (3) the *common sense ontologies* include a wide-range amount of foundational knowledge as time, space, or causality.

The activities performed by the CBR application designer to model a domain, and to formalize it as LOOM knowledge base, are summed up as follows.

1. The designer begins with a preliminary idea of what domain is to be modelled, and selects from the library those ontologies that are potentially useful. For example, if we were modelling the *used-car* domain, a sensible choice would be the VEHICLES ontology which comprises knowledge about "*vehicles which are typically bought and sold through the classified ads ...*"[2]

[2] Excerpt from the OS documentation.

Our tool may suggest the designer some other ontologies depending on the ontologies previously chosen. In the example, the incorporation of the VEHICLES ontology will suggest the PRODUCT-ONTOLOGY inclusion, which "*defines the terms used for describing products, objects that are typically bought and sold ...*"[2] This, in its turn, will lead the system to consider the terminology within the SCALAR-QUANTITIES and STANDARD-UNITS ontologies.

2. The domain terminology from the ontologies has to be integrated as two term hierarchies: the concept hierarchy rooted by the THING concept, and the relation hierarchy, rooted by the BINARY-TUPLE relation. The designer chooses where, within those hierachies, the ontology components have to be placed, The system, in its turn, combines and propagates through inheritance the restrictions included in the terms, and eventually signals contradictions and incoherencies. Then, it is again the designer who must solve the integration problems detected. Anyway, we must point out that the issue of coherent integration of definitions from different ontologies is still an open problem [13].
3. Due to the fact that ontologies are very general and reusable, sometimes all the definitions inside an ontology are not useful for our concrete domain model. The elimination of not relevant terms is not essential but, in our approach, will effect on the final system efficiency and quality because the search space will be smaller and contain only relevant terms. Notice that the selection of a definition from an ontology can provoke the automatic inclusion of others interrelated definitions that can not be eliminated and conversely, the elimination of some definitions could cause others to be erased.
4. Mechanisms are also provided to allow the inclusion of new definitions, just in case some useful specific definitions for our domain are not included in the chosen ontologies. Unfortunately, when there are not appropriate ontologies to be reused, an effort is needed to build a new ontology (or knowledge base).

In its final form, our tool will allow the term hierarchies graphic visualization (the OS doesn't provide with this functionality) and select, merge, eliminate, move, add or modify their definitions.

3 The CBR Ontology

As it was described in Section 2, our approach proposes the use of an ontology library to build the domain model for knowledge-rich CBR applications. To take advantage of this domain knowledge, the CBR knowledge needed by the processes, or at least part of it, should be expressed in a similar way. We have developed an ontology for CBR (CBROnto) that provides the vocabulary for describing the elements involved in the CBR processes. CBROnto serves two purposes: the integration between the domain ontologies and the CBR process knowledge; and as a domain-independent framework to design CBR applications.

With this approach, the designer of a knowledge rich CBR application does not only borrow domain terminology from the ontology library but also CBR

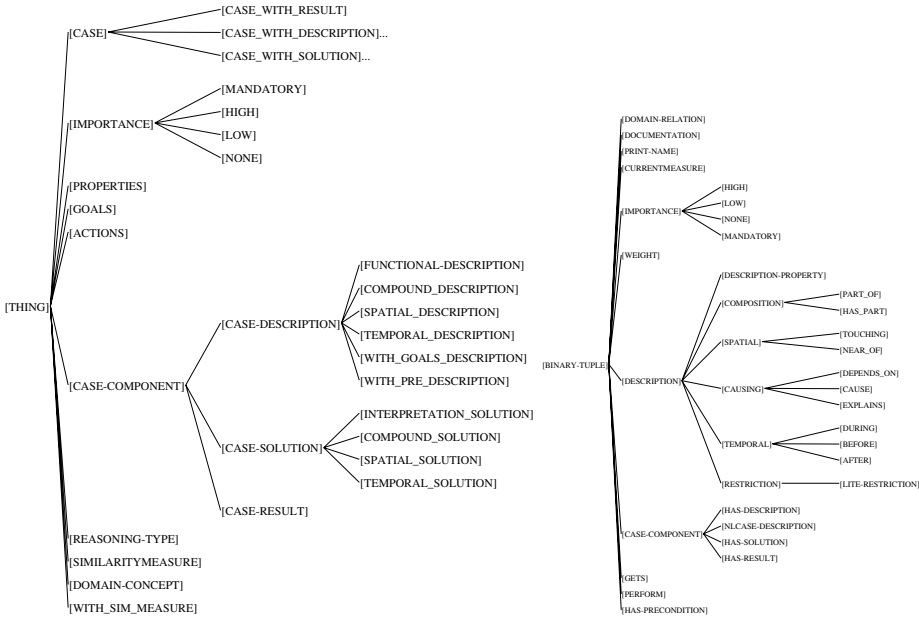

Fig. 1. The process support knowledge

terminology from CBROnto. CBROnto seeks to capture semantically *important* terms and the representation primitives commonly used in the case-based representation languages. It should be categorized inside the *task ontologies* because it provides a vocabulary for describing terms involved in the problem-solving CBR processes. Figure 1 shows a fragment of the CBROnto hierarchies.

CBROnto reveals our current view of certain CBR dependent but domain-independent terms that make possible different types of CBR, and that are used as the junction between the domain knowledge and the processes we define with a domain-independent perspective (see Figure 2). After the domain modelling, the phase of integration is based on classifying the domain terms with respect to the CBROnto terms. That mechanism allows the CBR processes being domain independent because they only refer to the CBROnto terms, that are correspondingly linked to the domain terminology by the classification mechanism.

Next subsections describe the main characteristics of CBROnto, the place it takes during the case representation, and how it is used to integrate the domain knowledge to be utilized by domain-independent CBR processes.

3.1 Case Representation

The cases in the case base should be described somehow by mean of the vocabulary provided by the domain model. The issue of case representation involves deciding the type and the structure of the domain knowledge within the cases. Efficiency pushes many CBR systems to use simple case representations that typ-

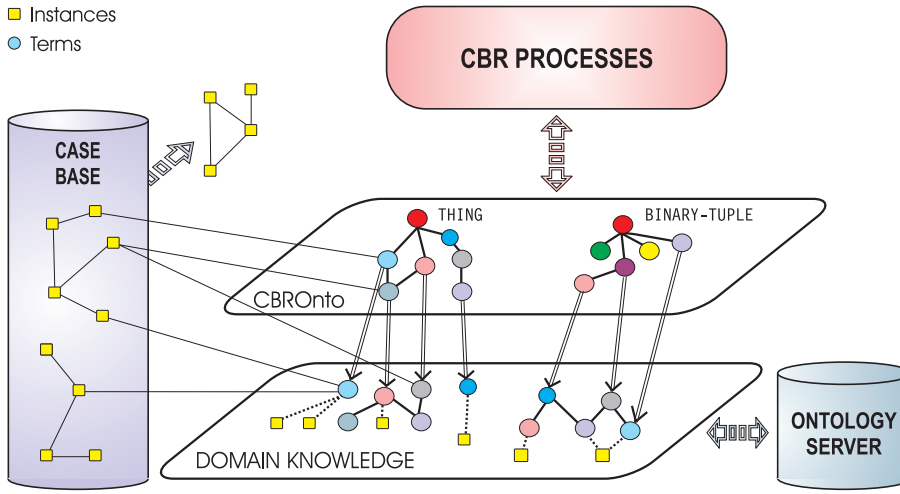

Fig. 2. CBROnto as a join between the domain terminology and the CBR processes

ically contain two sets of attributes, problem and solution features, and where there are no relationships or constraints between the features of a case. That is not our choice. We don't want to restrict the cases to be monolithic units of a fixed format because our framework is intended to be suitable for different types of CBR [4]. Our aim is to propose a rich framework to represent cases based on the terminology from the CBROnto together with a reasoning system that works with such representations.

The Case Representation Language In the CBROnto origins, the first decision was the definition of a *primitive* concept `CASE`. We will call *case-type* concepts to the `CASE` subconcepts. That way, cases can be represented as instances of the case-type concepts and will be described by using both the domain vocabulary provided by the domain model, and the CBR vocabulary provided by the CBROnto. Cases are represented as instances of different `CASE` subconcepts, so they won't have, in general, the same structure. Besides, the concrete cases (`CASE` instances) may add other proper features to the fixed structure inherited through the case-type concepts.

The designer will define case-type concepts to represent the new types of cases. The CBROnto vocabulary is used to guide the definition of these concepts by providing with CBR semantically important terms as `has-description`, `has-solution`, `has-result`, `similarityMeasure`, `weight`, `goal`, `precondition`, or `description-property`.

Our case representation language is based on the Loom instance definition language and on the CBROnto terminology. This representational framework allows complex structures and does not restrict the possible relations among the parts of a case, facilitates the definition of cases having different structures, is

able to handle incomplete cases and allows default values (by inheritance). The case instances may be related with other individuals, and in particular with other case instances, i.e. a case can be related with cases that are cases themselves.

We propose the use of an instance of the concept `CASE-DESCRIPTION` to represent the description of a case. The `has-description` relation links a `CASE` instance with the individual representing the description of this case. CBROnto includes the following components to describe a `CASE-DESCRIPTION` instance. The use of different components builds different types of case descriptions.

- The kind of reasoning the case will be used for. We use (by now) the following: `diagnosis`, `evaluate`, `explain`, `design`, `solve`, and `search`. They are represented as instances that will be linked to the case description instance by means of the `kind_of_reasoning` relation.
- The goals achieved by the case. The goals will be specific for the concrete domain and will be represented as instances of the CBROnto `GOAL` concept.
- The restrictions to be considered before applying the solution.
- Other suitable properties to describe the case: `composition`, `causing`, `temporal`, or `description-property` relations. (see Figure 1 (right)).

That way, the CBR processes can take advantage of this explicit definition of certain parts of the case structures, through the concept and relation hierarchies and the DLs inference mechanisms.

The solution of a case is represented as an instance of the `CASE-SOLUTION` concept. The `has-solution` relation links a `CASE` instance with the `CASE-SOLUTION` individual representing its solution. We use a general perspective, because the solution of a case depends very much on the kind of reasoning the case represents. The kind of solution is represented by classifying the solution individual below the `CASE-SOLUTION` subconcepts. It can be a designed component, a layout, a plan, or a diagnostic or interpretation for the current situation. As an example of the CBROnto terms that might be used to describe a `CASE-SOLUTION` instance we cite the *spatial*, *temporal*, *composition* or *causing* relations to represent the temporal sequence of reasoning steps used to solve the problem represented by the case; the spatial layout of the pieces used to design a component, or the adaptation (dependency) knowledge used to build this solution and links to the cases used to make it. To finish with the case main parts, the result of a case might include components as the success or failure of the case, the explanation of a failure, or links to other possible solutions.

3.2 The Domain and CBROnto Integration

As we have introduced, after the domain modelling phase, there is an integration phase where the CBR application designer relates the specific domain knowledge with the CBROnto terms. This section aims to explain the basic mechanisms used to integrate the domain and the CBROnto term hierarchies.

We use DLs classification to relate the specific domain terms with the CBROnto terms. Suppose a domain relation that is used to describe a property of the

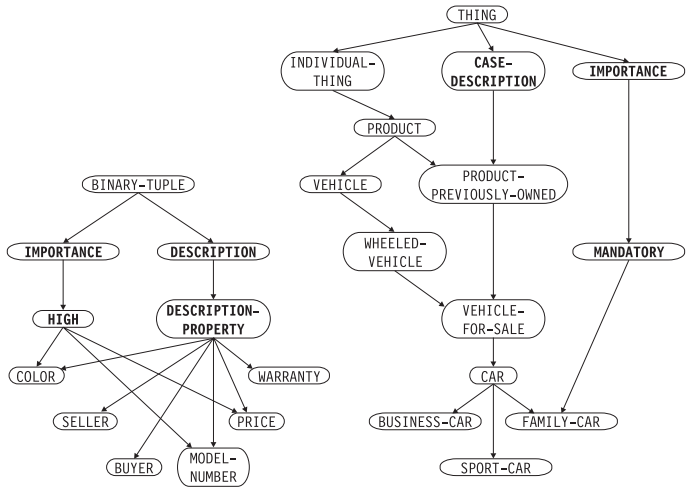

Fig. 3. Integration mechanisms based on classification (CBROnto terms in bold)

domain cases. For example, the `color` relation. With our framework, it will be classified as a subrelation of `description-property` because is a relation used to describe a domain property (see Figure 3). The same mechanism is used to classify other kinds of relations as `temporal`, `composition` or `spatial`.

We are using relation classification here, but the mechanism is similar in the concept hierarchy. For example, once we have modelled the used-car domain by means of the VEHICLES and PRODUCT Ontologies we want to represent the different types of cases. We would like to have cases representing second-hand products and without solution. Each case can include one or more products. We are building the case-type concept `CASE-PRODUCT-CLASS` with the structure of Figure 4 (left). The integration mechanism classifyies the `PRODUCT-PREVIOUSLY -OWNED` domain concept below the CBROnto `CASE-DESCRIPTION` concept. With this representation the `PRODUCT-PREVIOUSLY -OWNED` instances (from the `Product` ontology) are used as the description components of the `CASE-PRODUCT-CLASS` cases. Due to the classification mechanism, instances of `FAMILY-CAR`, `SPORT-CAR`, `BUSINESS-CAR`, `CAR`, and `VEHICLE-FOR-SALE`, are also appropriate instances to be used to describe a `CASE-PRODUCT-CLASS` case.

Also based on classification, our framework provides with a way to express preferences between the terms. This mechanism can be used for many purposes, and either by the designer, the final user of the designed CBR application, or by the organization processes. The importance for the case descriptors used during retrieval can be expressed by classifying them under the `IMPORTANCE` terms: `MANDATORY`, `HIGH`, `LOW`, and `NONE`. The domain independent CBR processes will prefer the domain relations classified under the `HIGH` relation and avoid the `NONE` classified ones. Figure 3 illustrates the use of the `HIGH` relation to strengthen the `color`, `model-number` and `price` domain relations; and the use of the `MANDATORY`

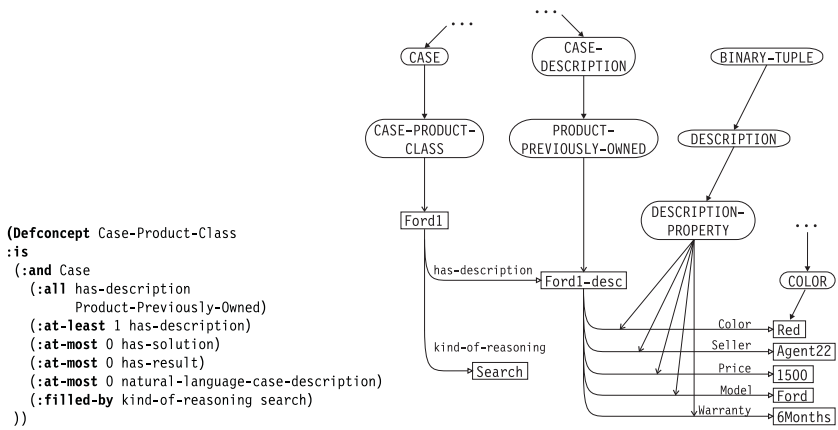

Fig. 4. Case definition example

concept to indicate that only the `FAMILY-CAR` type of `CASE-DESCRIPTION` should be considered for this retrieval.

The designer doesn't handle this low-level classification mechanisms nor classify one by one every domain term. Due to the inheritance mechanism only the top level terms in the hierarchies should be classified. Besides, we are developing a graphical environment to help this integration between the domain and the CBROnto terminology.

4 The CBR Processes

The described representational framework facilitates general and homogeneous CBR processes that refer to the CBR terminology and not to the specific domain terminology (see Figure 2). In this sense, the CBR processes are domain-independent but they are guided by the domain terminology organized below (in the subsumption hierarchies) the CBROnto terms. We are working nowadays in the development of the CBR processes and we are no elaborating here their details but only enumerating some of the alternatives we are considering.

Several alternatives can be chosen in our system to *index* the cases. The straight one is the use of the domain terminology as the case organization structure. That's the approach we used in [5]. Other approach [14] is let the designer explicitly define indexes as new DLs concepts. The links between indexes and between cases and indexes are automatically (semantically) computed. The alternative we are mainly using is the computation of a different index structure by inductive techniques guided by the domain knowledge.

With regard to *retrieval* and *similarity* assessment, the straight possibility [1] is the use of the LOOM query language to enable the user to describe the current situation and interests. Also, in the line of [12,14] a similarity term (concept) could be explicitly computed (and automatically classified) to represent in a declarative way the similarity and differences between the cases, expressed with

```
System question:        GIVE A NAME FOR THE NEW CASE
User answer:            Ford1
System question:        CLASSIFY THE NEW CASE CHOOSING FROM THE LIST.
System internal:        (Get-subconcepts CASE)  Response: Case-Product-Class
User answer:            Case-Product-Class
System question:        HOW DO YOU DESCRIBE YOUR CASE?
System internal:        (Get-subconcepts PRODUCT-PREVIOUSLY-OWNED)
; The concept PRODUCT-PREVIOUSLY-OWNED is used due to the restriction
;(:all has-description PRODUCT-PREVIOUSLY-OWNED) in the case-product-class concept
Response:               PRODUCT-PREVIOUSLY-OWNED, VEHICLE-FOR-SALE, CAR,
                        FAMILY-CAR, SPORT-CAR, BUSINESS-CAR.
User answer:            FAMILY-CAR
System internal:        (createm 'Ford1-desc FAMILY-CAR)
                        (tell (:about Ford1 Case-Product-Class
                                       (:all has-description FAMILY-CAR)
                                       (has-description Ford1-desc)))
Response:               Recognition changes in the KNOWLEDGE BASE:
                        entry:  Ford1-desc    CASE-DESCRIPTION
                        entry:  Ford1         CASE-PRODUCT-CLASS
                        entry:  Ford1         CASE_WITH_DESCRIPTION
                        entry:  Ford1         CASE
System internal:        (get-subrelations DESCRIPTION)
Response:               COLOR, SELLER, BUYER, MODEL, PRICE, WARRANTY
System question:        WHAT'S THE COLOR OF THE CASE?
System internal:        (retrieve ?x (range color ?x))
Response:               |concept| (COLOR)
                        (get-instances (fc COLOR))
                        (GRAY BLUE BLACK RED GREEN WHITE YELLOW ORANGE)
User answer:            RED
System internal:        (tellm (:about Ford1 (COLOR Red))
.....................
```

Fig. 5. User interaction simulation

the domain terminology. Another possibility is the representational approach that assigns similarity meaning to the path joining two individuals. We are using CBROnto to define different similarity components depending on the used terms: the *structural similarity* will be computed based on the composition relations (part-of, has-part), the *semantics similarity* is due to all the concepts and relations describing the meaning of the case, the *contextual similarity* depends on the case context relations and the *adaptation similarity* will use the dependency knowledge.

4.1 A Case Definition Example

This section exemplifies the process of incorporating a new case to the case base. The used-car domain case structure represented by the `CASE-PRODUCT-CLASS` concept (see Section 3.1 and Figure 4 (left)) is very simple and doesn't illustrate all the representational possibilities, for example, cases with solutions and results, or more complex descriptions described by not simple properties. However, it exemplifies and facilitates the comprehension of the domain-independent and classification based mechanisms that provide access to the domain terminology through the CBROnto terms.

Figure 5, simulates the user interaction to instantiate the CBROnto concepts and relations. The key issue in the example is that the system questions are dinamically generated by querying the knowledge base with domain-independent questions referring only to the CBROnto terms (*system internal* in the figure).

Notice that during the example, the domain terms are always reached because they are classified below the CBROnto terms.

The user chooses the `FAMILY-CAR` type of `CASE-DESCRIPTION`, and the system creates the individual `Ford1-desc`, which is an instance of the `FAMILY-CAR` concept; and an individual called `Ford1`, which is an instance of the concept `CASE-PRODUCT-CLASS` and that is related with `Ford1-desc` by the `has-description` relation. The meaning is that the `Ford1-desc` individual represents the description of the case `Ford1`. The next step is describing the `Ford1-desc` instance. The system will access the relation hierarchy and formulate a question for each `DESCRIPTION` subrelation. When possible, the system offers a set of fillers according to the range of each relation (as is the case with the `color` relation). Figure 4 (right) shows the Ford1 case resultant from this user interaction.

5 Conclusions and Related Work

In this paper we have not aimed to describe all the terms within the CBROnto, mainly because we don't think it is complete but it is evolving with our current work. It only makes explicit certain CBR terms that are useful as a junction between the domain knowledge and the CBR processes defined with a domain-independent perspective. The use of domain ontologies provides a CBR application with the vocabulary for describing a domain and interpreting a description of a problem in that domain. The use of domain ontologies guides the construction of cases (and queries) and constitutes a warehouse of vocabulary to solve lexical, semantic and synonym problems. Besides, it avoids misunderstandings if cases are given by different sources and allows for a seamless integration of cases without requiring all the cases to have the same structure. As the main drawback of our approach we cite the ontology integration problem. When there are not appropriate ontologies to be reused, an effort is needed to build new ontologies or to integrate definitions from different ontologies [13]. Anyway, we consider this effort is not waste time if this knowledge is reused for other applications.

Although we aim to build a tool capable to fit many CBR approaches, we don't expect to contribute in the design of simple CBR applications, where cases can be attribute-value vectors, and where many optimised technologies exist. Our contribution is expected to be in the knowledge intensive applications where our system will allow for a quick way of design and prototyping a CBR application and study the results of the incorporation of certain domain knowledge.

Other works [1,10,14] use DLs to represent the cases and the domain knowledge for CBR systems, but none of them do it with a domain and application independent view as ours. As we did in [5,7], these works presume the existence of a DL knowledge base in the application domain, typically built ad hoc for a concrete application.

Other CBR systems codify specific domain ontologies. In [2] an ontology is developed for the representation of cases and adaptation knowledge for a CBR system that helps in the estimation of effort for software project. The ontology manages software project terms like task, project, resource and deliverable. Their

proposal fits in our architecture if the built project-effort ontology is included in the library of domain ontologies. More related with our CBROnto is the Multis Ontology (see [6] for references), a task ontology that defines the terminology for scheduling. We intend to use it to enhance CBROnto in the design of planning CBR applications. Also we plan to integrate the context ontology developed in [11] for its use in contextualized problem solving and learning.

The development of this system brings several lines of CBR future research. We will perform empirical studies comparing our system with other CBR shells, mainly for the CBR system design time, and the efficiency, quality of the results, suitability and effectiveness of the designed applications.

References

1. Ashley K. & Aleven V., 1993: "A logical representation for relevance criteria", in *Topics in CBR* (Wess S., Althoff K. & Richter M., eds.), Springer-Verlag. 45, 47
2. Aarts R. J., 1998: "A CBR Architecture for Project Knowledge Management", in *Advances in CBR* (Smyth B. & Cunningham P., eds.), Springer-Verlag. 47
3. Brachman R. J., McGuinness D. L., Patel-Schneider P. F., Resnick L. A., & Borgida A., 1991: "Living with CLASSIC: When and How to Use a KL-ONE-Like Language". In *Principles of Semantic Networks*. Morgan Kaufmann Publishers. 39
4. Gebhardt F., VoB A., Gräther W., Schmidt-Belz B., 1997: *Reasoning with Complex Cases*. Kluwer Academic Publishers. 42
5. Gómez-Albarran M., González-Calero P. A., Díaz-Agudo B. & Fernndez-Conde C., 1999: "Modelling the CBR Life Cycle Using Description Logics", in Procs. of the 3rd International Conference on Case-Based Reasoning (ICCBR'99). K.-D. Althoff, R.Bergmann & L. K. Branting (Eds.). 45, 47
6. Gómez-Pérez A., 1998: "Knowledge Sharing and Reuse". *The handbook on Applied Expert Systems*. By Liebowitz. ED CRC Press. 1998. 38, 39, 48
7. González-Calero P. A., Gómez-Albarran M., & Díaz-Agudo B., 1999: "Applying DLs for Retrieval in Case-Based Reasoning", in Procs. of the 1999 Description Logics Workshop (DL'99). 47
8. Gruber, T. "A translation Approach to portable ontology specifications". *Knowledge Acquisition*. Vol, 5. 1993. 38
9. Mac Gregor, R., 1991: "The evolving technology of classification-based knowledge representation systems", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge* (J. Sowa, ed.), 39
10. Napoli A., Lieber J., & Courien R., 1996: "Classification-Based Problem Solving in CBR", in *Advances in CBR* (Smith I. & Faltings B., eds.), Springer-Verlag. 47
11. Ozturk P. & A.Aamodt, 1998: "A Context Model for Knowledge-Intensive Case-Based Reasoning ", *International Journal of Human-Computer Studies*. Vol.48,3. 48
12. Plaza E., 1995: "Cases as Terms: A feature term approach to the structured representation of cases". In Procs. ICCBR-95. 45
13. Pinto H. S., Gómez-Pérez A. & Martins J. P., 1999: " Some Issues on Ontology Integration", in IJCAI-99, Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. 40, 47
14. Salotti S. & Ventos V., 1998: "Study and Formalization of a CBR System using a Description Logic", in *Advances in CBR* (Smyth B. & Cunningham P., eds.), Springer-Verlag. 45, 47