

The Wings of jCOLIBRI: A CBR Architecture for Semantic Web Services

Juan A. Recio-García, Juan M. Gómez-Berbis, Belén Díaz-Agudo, Pedro A. González-Calero

Abstract—With the advent of Semantic Web Services, a paradigm that results from adding semantics to Web Services, the challenge remains in finding a suitable way of capturing the goals and objectives of the user and attaining a technological solution to fulfill them. In this paper, we aim at solving this problem combining Case Based Reasoning (CBR) and semantics for Web Services in the Web INdependent and Generic Services Architecture (WINGS).

The objective of WINGS is to improve the composition of semantic web services by taking into account the goal of the user. We use for this purpose CBR assuming that similar cases may exist and be reused after selection and adaptation. One of the main focus of our approach resides in the semantic match making piloted by an ontology.

As a proof-of-concept WINGS has been built above jCOLIBRI, a well-known framework to design CBR applications, and CBRonto, and ontology with knowledge of CBR.

Index Terms—Case Based Reasoning, Semantic Web Services, Semantic Annotation, Ontologies

I. INTRODUCTION

WEB Services are the most suitable medium of choice to publish applications and services for organizations. However, realization of its full potential requires further technological advances in services interoperation, discovery and composition. Particularly, the challenge remains in showing how multiple Web Services can be integrated to provide a new, value-added service to the end-user fulfilling the goals of the user. Adding semantics to Web Services attempt to foster automatic service discovery, matchmaking and composition since Web Services are described and published from a purely syntactical perspective that hampers any inference or similarity based strategies [1].

Semantic Web Services initiatives such as WSMO [2] or OWL-S [3] require an ontological description of the objectives a user may have (the concept of goal in WSMO and the concept of Service Profile in OWL-S from the service perspective as

Manuscript received May 12, 2008. This work is supported by the MID-CBR project of the Spanish Committee of Education & Science TIN2006-15140-C03-02.

J.A. Recio-García, B. Diaz-Agudo and Pedro Gonzalez-Calero, are with the Facultad de Informatica de la Universidad Complutense de Madrid, 28040, Madrid, Spain. (phone: +34 91 3947624; fax: +34 91 394 7529 e-mail: jareciog@fdi.ucm.es, belend@sip.ucm.es, pedro@sip.ucm.es)

J.M Gomez-Berbis was with Universidad Carlos III de Madrid, Spain. (e-mail: juanmiguel.gomez@uc3m.es).

what “the service can do for the user”). These descriptions are turned into the execution of a number of Web Services fulfilling the user goal. However, there are multiple caveats, inconsistencies and problems to relate goals with Web Services. First of all, lack of accuracy of capturing user preferences and relating it to domain-specific constraints are raised. Secondly, vital information about the services execution is not envisaged and knowledge from previous experiences or executions adequacy fails to be used. CBR provides a well-known and consistent established research corpus, methodology and formal scientific basis to approach these challenges. In this paper, we present the Web Services in the Web INdependent and Generic Services Architecture (WINGS), a novel approach which combines Natural Language Processing (NLP) techniques, semantic matchmaking and CBR to address those challenges and solve the user goal fulfillment problem regarding to searching, finding, interacting and integrating Semantic Web Services.

The objective of WINGS is to improve the composition of Semantic Web Services by taking into account the goal of the user. We use for this purpose CBR assuming that similar cases may exist and be reused after selection and adaptation. One of the main focus of our approach resides in the semantic match making piloted by ontology.

As a proof-of-concept we have applied WINGS to improve the configuration of a particular type of applications: CBR systems. We use jCOLIBRI a framework to develop CBR systems where each system is composed by Web Services. jCOLIBRI provides most of the code needed to represent structured cases, methods and similarity functions used in CBR systems. jCOLIBRI includes facilities to work with different case representations, namely, first order logics, data based records or XML files. It incorporates DLs reasoning capabilities and facilitates the development of Knowledge Intensive CBR (KI-CBR) applications based in ontologies. The main advantage of using jCOLIBRI is mainly an easier development of CBR systems. To reach this goal jCOLIBRI proposes a design process based on reusing existing CBR knowledge (terminology, designs, tasks, methods, implementations), on the integration and composition of new components, and the extension of existing components and their collaborations.

“The WINGS of jCOLIBRI” proposes creating new CBR systems semi-automatically based on the user description of goals of the system, and from cases describing CBR systems previously designed and configured. These systems are reused

after selection and adaptation. The knowledge to annotate CBR services and systems is taken from CBR_{Onto} an ontology that formalizes knowledge about CBR systems.

The remainder of the paper is organized as follows. Section 2 describes the fundamental principles of CBR and Semantic Web Services. In Section 3, the WINGS architecture is presented. Section 4 describes the use of WINGS to configure CBR applications in jCOLIBRI. Finally, Conclusions and Related Work are discussed in Section 5.

II. CBR AND SEMANTIC WEB SERVICES

Semantic Web Services initiatives define infrastructure that combines Semantic Web and Web Services technologies to enable automatic Web service selection and use. To get these goals, a fundamental component would be the markup of Web services to make them computer-interpretable [4]. Within this markup and semantic service descriptions, powerful tools should be enabled across the Web service lifecycle. This lifecycle includes automatic Web service discovery to locate either a Web service that provide a particular service, or a Web Service that is similar enough to be adapted to the current request; and Automatic Web service composition and interoperation that involves the automatic selection, composition, and interoperation of appropriate Web services to perform some task, given a high-level description of the task's goal.

At the same time, another research community has intensively been working about similarity based retrieval and adaptation of solutions to fit new situations: two key aspects in the working SWS lifecycle. Case-Based Reasoning (CBR) is one of the most successful applied AI technologies of recent years. CBR is based on the intuition that new problems are often similar to previously encountered problems, and therefore, that past solutions may be reused, directly or through adaptation, in the current situation. CBR systems typically apply retrieval and matching algorithms to a case base of past problem-solution pairs. Another very important feature of CBR is its coupling to learning. The driving force behind case-based methods has to a large extent come from the Machine Learning community. A strong effort has been done in the CBR community to solve the problems of similarity and adaptation in different contexts, with different approaches to case representation, organization and storage, and amount of knowledge, from knowledge intensive to data intensive approaches. Many successful research and industry results are presented every year in the European and International CBR conferences. During the last few years, Ontologies and Description Logics (DLs) have become systems of interest for the CBR community. There has been a body of work [5][6][7][8] on Knowledge Intensive-CBR, and on taking advantage of the DL reasoning mechanisms for the processes involved in the CBR cycle. Although a number of different approaches are considered, they all focus on the intuition that the formal semantics and the capability of DLs to maintain a terminological taxonomy are interesting properties to measure similarity and to manage a case base.

In this paper we bring together the efforts of the SWS and

CBR communities, and try to find synergies between both of them. Given a certain query describing the user goals, automatic Web service discovery typically uses very simple ways of similarity that could be greatly improved by the use of more complex CBR similarity methods. Besides retrieval, the SWS lifecycle could also benefit from other CBR processes like case reuse, revise and learning [11].

A first simple approach of using CBR in the SWS lifecycle is considering each atomic SWS as a case of a case base distributed through the Semantic Web. The use of the complete CBR cycle [11] would allow the inclusion of reuse methods where simple variations of the retrieved SWS could be automatically generated to better fit the user query. However adaptation can be exploited much more within complex solutions. Instead of simple SWS, the solution of each case could be a workflow composed of different SWSs that get certain goals when are executed in a certain sequence. A case could then be described by the goals that the solution satisfies, and also with information about the experience, either good or bad, of the use of this solution to solve one or more goals. This is the approach we are taking in the WINGS architecture that is described in section below. Section 4 describes how to use WINGS together with jCOLIBRI where each CBR system itself is composed from semantic web services.

III. WINGS

Both from the Semantic Web Services and CBR user standpoint, a number of issues are raised when trying to express the user wish and intentions. The challenge remains in finding a suitable way of capturing the goals and objectives of the user and attaining a solution to fulfill them. In the SWS terminology, the notion of goal encapsulates such challenge. A goal is a representation of an objective for which fulfillment is sought through the execution of a certain sequence of SWSs. If the wish of the user can be first extracted and captured and matched with a particular Case Description, the fulfillment of this goal could be achieved applying the Case Solution.

The case base is a set of cases. Each case has three components:

- Description of the goals that the solution satisfies. The vocabulary to describe the goals is taken from a domain ontology, like "buy" a "book", or "book" a "single room".
- Solution includes the sequence of SWSs. It represents both, the code of the services and the dependencies between services and goals.
- Result includes information about the experience, either good or bad, of the use of this solution to solve one or more goals.

The Web INdependent and Generic Services Architecture (WINGS) is a fully-fledged software architecture designed to interact with Semantic Web Services and CBR following the aforementioned four layers as depicted in Figure 1.

- User Goal Extraction: The user expresses usually in natural language what is concerned about. NLP techniques are used to extract concepts pertaining to the space of ontologies being used and a lightweight ontology is created.

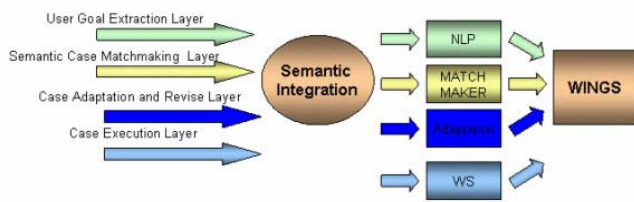


Fig. 1 - The WINGS ARCHITECTURE

- **Semantic Case Matchmaking:** Once we have these concepts extracted, the most suitable goal is selected and matched with the most similar case. These are the cases that are described by concepts that are similar to the concepts that describe the user query.
- **Case Adaptation and Revise.** Cases are composed by a sequence of atomic SWSs that provide a particular functionality. If the retrieved case does not fulfill exactly the query goals, then an adaptation process could modify its solution. We offer three ways of adaptation: remove an atomic SWS from the workflow, add a new SWS in the sequence, and substitute one of the SWS by other similar SWS.
- **Case Execution:** Once the most similar case is found, these Web Services are executed and, as a result, the goal of the user is achieved.

Each layer has a particular and precise functionality. Layers observe a dependency relationship from down to up. This means that the upper layers rely on some functionality of the lower layers. Layered architectures are used in different domains but particularly in complex engineering systems where each layer implements a different aspect of the information exchange. For example, the bottom layer usually represents the basic communication event at a physical level, e.g., the physical electrical signals of communication. In the following the four layers of the WINGS architecture will be detailed.

A. User Goal Extraction Layer

The User Goal Extraction Layer is heavily based on the GODO architecture language analyzer and concept extraction layer [12]. First of all, the language analysis challenge is to filter and process the input introduced by the user in natural language and determine the concepts (attributes and values) and relations included in it. The language analyzer is based on GATE [13] and intensive training on a concise and reliable knowledge base of concepts is compulsory to assure a quality threshold in the results. Fundamentally, the analyzer receives a text in natural language as a parameter and returns the concepts (attributes and values) within the text which are related to the ones pertaining to the ontologies space being used. A lightweight ontology using the concepts and relationships identified is built. For example, if we would like to buy a book about Tai Chi in Amazon, price being less than twenty dollars, the layer components would identify the concepts “book”, “price” and its relationship, which would outcome a small ontological structure.

B. Semantic Case Matchmaking Layer

We refer to Semantic Case Matchmaking as the strategies to compute the similarity between the ontological structure provided by the previous layer and the cases we have in our CBR case base. From this architecture layer viewpoint, we access the CBR knowledge base, previously annotated in OWL-DL and use subsumption to relate the similarity or “closeness” of the cases with the provided ontological structure. As a result, one of the cases is selected and issued to the next layer. To represent the *description* of each case we describe a structure in OWL that is classified and indexed by the domain ontology concepts representing goals. The case description is represented as a simple instance that is classified below the domain concepts that its solution satisfies and other information about the applicability of this solution. There are different approaches to similarity computation based on Ontologies [14][15]. *Classification based retrieval* builds a concept or an individual description using the restrictions specified in the query. This concept/individual is then classified, and finally all its instances/siblings are retrieved. The second approach, *computational based retrieval*, uses numerical similarity functions to assess and order the cases regarding the query. The use of structured representations of cases requires approaches for similarity assessment that allow comparing two differently structured objects, in particular, objects belonging to different object classes. Similarity measures for structure case representations are often defined by the following general scheme [16]: The goal is to determine the similarity between two objects, i.e., one object representing the case (or a part of it) and one object representing the query (or a part of it). We call this similarity object similarity (or global similarity). The object similarity is determined recursively in a bottom up fashion, i.e., for each simple attribute, a local similarity measure determines the similarity between the two attribute values, and for each relational slot an object similarity measure recursively compares the two related sub-objects. Then the similarity values from the local similarity measures and the object similarity measures, respectively, are aggregated to the object similarity between the objects being compared. Following the previous example, subsumption would determine that the best case to apply in the CBR knowledge base is Case Ni, “Search and Buy a product in Amazon”.

C. Case Reuse & Revise

Many authors agree about the fundamental role of case adaptation in the problem solving capabilities of CBR systems. In WINGS we propose an ontology based model and an adaptation scheme based on substitutions. This method is included in jCOLIBRI and described in [17]. Case adaptation plays a fundamental role in the ability of CBR systems to solve new problems. Case adaptation is a knowledge-intensive task and most CBR systems have traditionally relied on an enormous amount of built-in adaptation knowledge in the form of *adaptation rules* (if ... then do ...). Our approach relies on the explicit representation of general terminological knowledge about the domain. That way, certain adaptation knowledge is explicitly represented in the domain knowledge taxonomy, as it

indicates, for instance, that individuals that are close in the taxonomy are eventually interchangeable.

The adaptation scheme is based mainly on deletions and substitutions [18][19]. Dependencies within a case are explicitly represented in order to guide the adaptation. In WINGS there are explicitly represented dependencies between goals and services; and between services in the workflow. If an element is removed the dependent elements are also removed. If an element e_1 is substituted by other element e_2 then the dependent elements are substituted. The search for substitutes is guided by the ontology.

Adaptation is required when the retrieved case does not fulfill all the required goals given in the query, or when it solves goals that are different from the ones in the query. We need to apply the deletion and/or substitution adaptation operators. We propose an adaptation mechanism as a process that propagates changes from description to solution items, as follows:

The list L of SWSs in the solution that need to be adapted is obtained. Dependencies between goals and SWS are used. Each SWS in the solution is linked with one (or more) of the concepts in the case description that describe the goal of a certain SWS in this composition.

Every SWS in L is deleted or substituted by a proper new SWS. First, those that only depend on values from the case description, then, those that depend on other items of the solution that have already been adapted. Of course, circularity is not allowed in the dependency relation. Section 4 includes an example of the adaptation process.

D. Case Execution Layer

Once the case has been determined, two subsequent actions must be undertaken: refinement and execution. Refinement is the ability of narrowing down the scope of the abstract description of the case for the concrete situation being tackled. Execution is the software “plumbing mechanism” that invokes a number of Web Services in the particular sequence and order described in the Case, following a business logic i.e. a deterministic distribution of execution which results in a very specific state of the world after the execution.

This layer deals with significant overhead due to a number of orthogonal features potentially inherent to the fact of communication such as security, reliability, communication means and so on and so forth. Therefore, conceptual partitioning of this layer into another one dealing with these aspects might be envisaged in future versions.

In the example, Case Ni, “Search and Buy a product in Amazon”, can be decomposed into invoking the search Amazon Web Service, retrieving the results, verifying a number of constraints (availability, price, date of publishing, etc), executing the buy Amazon Web Service and, finally, polling the “delivery acknowledgment” Amazon Web Service to close the business transaction if and only if the delivery has taken place.

IV. USING WINGS TO CONFIGURE CBR APPLICATIONS IN JCOLIBRI

The approach shown in Section 3 improves the configuration

of applications composed by web services. It is a flexible schema that can be applied in many scenarios if there are two prerequisites:

1. A set of seed cases representing previously configured systems using SWS composition, and
2. A domain ontology used to describe the semantics of the services, i.e. their goals.

This section explains the use of WINGS to configure CBR systems using the jCOLIBRI framework [20]. jCOLIBRI¹ a well-known framework in the CBR community that allows the development of CBR applications. jCOLIBRI fits our architecture because it is based on the reuse of past designs and implementations of problem solving methods (prerequisite 1); and formalizes the CBR knowledge using a domain-independent CBR ontology (CBROnto)[21] which is mapped into the classes of the framework (prerequisite 2). CBROnto allows the description of the components of the framework, their extension and composition. It also improves the acquisition and organization of the knowledge required to retrieve and adapt cases.

jCOLIBRI offers a library of reusable methods (services) that are semantically annotated using the CBROnto terminology. In this case study, we use methods from the textual CBR extension [22] that allows the reasoning over cases represented as texts.

A. JCOLIBRI

jCOLIBRI is an object-oriented framework in Java for building CBR systems. jCOLIBRI is aimed at CBR system designers. Expert programmers can use java to instantiate the framework, although the easiest way of use is based on its graphical configuration tools.

The underlying ideas of CBR can be applied consistently across application domains. However, developing a CBR system is a complex task where many decisions have to be taken. The system designer has to decide among a range of different methods for organizing, retrieving and reusing the knowledge retained in past cases. This process would greatly benefit from the reuse of previously developed CBR systems. jCOLIBRI offers a predefined set of tasks and a library of PSMs that can be reused to build CBR systems.

The architecture of jCOLIBRI (see Figure 2) is separated into two layers [23]: The bottom layer contains a library of methods that solve the CBR cycle, and components to connect with the persistence media that contains the cases. The top layer wraps the bottom layer methods as Semantic Web Services. The methods (services) are described using CBROnto and composed using a DLs Reasoner. The inclusion of ontology capabilities (loading OWL ontologies computed with DLs reasoners) increases the performance in the retrieval and adaptation of cases embedded in an ontology.

The WINGS of jCOLIBRI is an extension coupled to the top layer of the architecture of Figure 2. A CBR application designer using the top layer of jCOLIBRI, configures a composition of SWSs from the library to create a new working

¹ <http://gaia.fdi.ucm.es/projects/jcolibri.html>

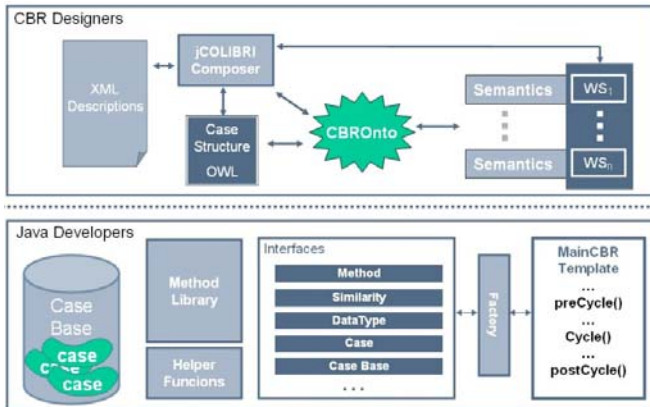


Fig. 2 - Two layers Architecture of jCOLIBRI

CBR system. These compositions are the set of seed cases for WINGS. Reusable methods in jCOLIBRI are semantically annotated using the CBROnto ontology.

CBROnto conceptualizes knowledge about CBR systems, and is formalized in OWL. The terminology of CBROnto allows describing, among other things, the structure of the cases of a CBR system, the organization of the case base, and the behavior of the methods. Defining the behavior of a method is based on concepts that represent the parameters, preconditions, results, etc.

WINGS facilitates the composition of new CBR systems by reusing previous compositions of successful CBR systems. In our particular use of WINGS together with jCOLIBRI, each CBR system (made of SWS compositions) is a case itself of the WINGS extension, that is CBR based, where similarity and adaptation of SWS workflows provide with successful configurations of new working systems. Although the double use of CBR could be confusing, the reason to choose CBR applications themselves as the cases is the availability of CBROnto: a very complete and detailed ontology for describing CBR software components.

Figure 3 shows a reduced view of CBROnto with the concepts that are used in the representation of a jCOLIBRI SWS. Imagine that we have a *retrieval* method that can be applied to recover *big case bases* that contain *cases* with a *compound (taxonomical) description* using a *linear organization*.

All these words in italics are concepts of CBROnto, so the semantic annotations of the suitable methods will include these concepts. The *execution context* is defined as the subset of CBROnto concepts used as labels to semantically annotate a certain CBR method or a complete CBR system.

We retrieve a suitable retrieval method RM_1 using the semantically annotated method and the semantic case matchmaking described in section 3B. Let's suppose now, we want to compose RM_1 with an *adaptation method* that is semantically described to be used with small case bases (due to efficiency problems with big case bases). According to its annotations RM_1 loads a big case base, the new method cannot be executed next. To check this restriction we can classify the execution contexts of both methods into the ontology (using a DLs reasoner) and obtain if they are compatible. Repeating this

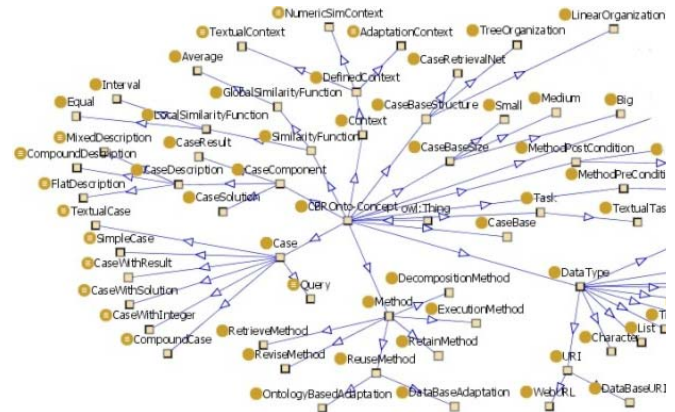


Fig. 3 – CBROnto

step we can find a correct configuration of our CBR system. With the use of WINGS the user can reuse these configurations to develop new CBR systems. The user describes his/her particular goals regarding the system design like: “*I need a CBR system for a huge case base that contains several texts and must be organized as a tree to improve the performance, I also would like the system to (...)*”, and then retrieves the most similar configuration. This configuration is found comparing the context created with the textual query and the context obtained calculating the most common subsumer of the contexts that describe each SWS of the configuration.

Once the configuration is found, WINGS changes it based on the reuse process described in section 3C. The reuse method removes or substitutes the pieces of the workflow (SWSs) that are incompatible with the query. As an example, let's suppose the user query includes the goal “*I would like the CBR system to (...) compute similarity of cases using the information contained in a ontology (...)*”. However, the retrieved CBR system includes a similarity method using a similarity table. WINGS changes the table-based similarity method by a ontology-based similarity method. Obviously, CBROnto also represents the requirements of each method, this is: a URL of an ontology or the URL of the file containing the similarity tables. Using this information the system can ask the user for the required parameters used by the new method.

V. CONCLUSIONS AND RELATED WORK

As the technologies associated to Business Process Modeling interactions, such as Web Services gain momentum being chosen by organizations to deploy their applications, the need for a particular application fulfilling the customer expectations becomes the Holy Grail. The goal of current Semantic Web Services [1] initiatives such as OWL-S [3] or WSMO [2] is to actually add semantics to current Web Services to match the expectations of the consumer of the web service. Currently, the most prominent compliant implementation of SWS is the WSMX [24] platform. In the WSMX architecture, a goal specifies the objectives that a client may have when he consults a web service and it also contains a list of preferences. These preferences represent constraints on non-functional properties of a web service i.e. they narrow the scope of the selection spectrum of a web service. Our approach is quite

complementary to the one presented is the aforementioned Semantic Web Services initiatives. However, the benefits and forthcoming of adding a CBR flavor to the approach have significant impact in achieving the user expectations. Although there are many synergies between CBR and SWS communities there are not many works where they are explicitly described. However, we have found recent efforts in this line. In [25] the authors present a similar approach to compose Web services satisfying the desired user goal. After the query goal is specified, the system retrieves and adapts service composition patterns that are executed. Then the goal and the created service composition pattern is stored for later re-use. It is a nice approach similar to ours, but it seems to be a preliminary work and even if they mention adaptation they do not describe how to do it in a real case study. Other similar work is [26] where a specific language is proposed to describe services compositions and a CBR approach is used to retrieve previous composition. This paper improves that approach by using ontologies to organize and describe the compositions of services and easing the definition of queries through a natural language interface.

This paper describes a CBR approach to assist in discovery and adaptation of web service compositions. It describes WINGS, an architecture for such an approach, and a case study in jCOLIBRI, a framework to design CBR systems, where each CBR system itself is composed from Semantic Web Services.

REFERENCES

- [1] D. Fensel and C. Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.
- [2] J. D. Cristina Feier. Wsmo primer. <http://www.wsmo.org/TR/d3/d3.1/v0.1/>, April 2005.
- [3] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. Mc-Dermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The owl-s approach. *Lecture Notes in Computer Science*, 3387:26–42, 2005.
- [4] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
- [5] G. Kamp. Using description logics for knowledge intensive case-based reasoning. In *Third European Workshop on Case-Based Reasoning (EWCBR '96)*, Lausanne, Switzerland, pages 204–218. Springer-Verlag, Berlin, 1996.
- [6] J. Koehler. An application of terminological logics to casebased reasoning. In J. Doyle, E. Sandewall, and P. Torasso, editors, KR'94: *Principles of Knowledge Representation and Reasoning*, pages 351–362. Morgan Kaufmann, 1994.
- [7] P. A. González-Calero, M. Gómez-Albarrán, and B. Díaz-Agudo. A substitution-based adaptation model. In *Challenges for CBR - Proc. of the ICCBR '99 Workshops*. University of Kaiserslautern, 1999.
- [8] B. Díaz-Agudo and P. A. González-Calero. An architecture for knowledge intensive CBR systems. In *Advances in Case-Based Reasoning (EWCBR '00)*. Springer, New York, 2000.
- [9] J. A. Recio-García, B. Díaz-Agudo, and P. A. González-Calero. A distributed cbr framework through semantic web services. In *Research and Development in Intelligent Systems XXII (SGAI 2005)*, pages 88–101. Springer, 2005.
- [10] B. Díaz-Agudo and P. A. González-Calero. *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, chapter An ontological approach to develop Knowledge Intensive CBR systems, pages 173–214. Springer-Verlag, 2006.
- [11] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(i), 1994.
- [12] J. M. Gómez, M. Rico, F. García-Sánchez, R. M. Béjar, and C. Bussler. Godo: Goal driven orchestration for sws. In *Proc of the 1st WSMO Implementation Workshop*, 2004.
- [13] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*, 2002.
- [14] B. Díaz-Agudo and P. A. González-Calero. Knowledge intensive CBR through ontologies. In *Proc of the UK CBR Workshop*, 2001.
- [15] J. A. Recio-García, B. Díaz-Agudo, P. A. González-Calero, and A. Sánchez. Ontology based CBR with jCOLIBRI. In *Proc. of the 26th SGAI Int. Conf. (AI-2006)*, pages 149–162, UK, 2006. Springer.
- [16] R. Bergmann and A. Stahl. Similarity measures for objectoriented case representations. In *Advances in Case-Based Reasoning (EWCBR '98)*, pages 25–36. Springer, 1998.
- [17] M. Gómez-Albarrán, P. A. González-Calero, B. Díaz-Agudo, and C. Fernández-Conde. Modelling the CBR life cycle using description logics. In *Proc. of ICCBR '99*, pages 147–161, UK, 1999. Springer.
- [18] P. A. González-Calero, M. Gómez-Albarrán, and B. Díaz-Agudo. A substitution-based adaptation model. In *Challenges for CBR - Proc. of the ICCBR '99 Workshops*. University of Kaiserslautern, 1999.
- [19] J. A. Recio-García, B. Díaz-Agudo, and P. A. González-Calero. A distributed cbr framework through semantic web services. In *Research and Development in Intelligent Systems XXII (SGAI 2005)*, pages 88–101. Springer, 2005.
- [20] J. A. Recio-García, A. Sánchez, B. Díaz-Agudo, and P. A. González-Calero. jCOLIBRI 1.0 in a nutshell. A software tool for designing CBR systems. In *Proc. of the 10th UK Workshop on CBR*, pages 20–28. CMS Press, 2005.
- [21] B. Díaz-Agudo and P. A. González-Calero. CBRonto: a task/method ontology for CBR. In S. Haller and G. Simmons, editors, *Proc. of the 15th International FLAIRS'02 Conference*. AAAI Press, 2002.
- [22] J. A. Recio-García, B. Díaz-Agudo, M. A. Gómez-Martín, and N. Wiratunga. Extending jCOLIBRI for textual CBR. In *Proc of the 6th Int. Conf. on CBR, ICCBR 2005*, volume 3620 of LNAI, pages 421–435, US, 2005. Springer.
- [23] J. A. Recio-García, B. Díaz-Agudo, A. Sánchez, and P. A. González-Calero. Lessons learnt in the development of a cbr framework. In *Proc of the 11th UK Workshop on Case Based Reasoning*, pages 60–71. CMS Press, 2006.
- [24] C. Bussler, E. Cimpian, A. Mocan, M. Moran, and M. Zaremba. WSMX: An Execution Environment for Semantic Web Services. *Position paper at W3C Workshop on Frameworks for Semantics in Web Services*, 2005.
- [25] G. Agre, T. Atanasova, and H. J. Nern. Case-based semantic web services designer and composer. In *Proceedings of the Conference Euromedia-2005*, pages 226–229, France, 2005.
- [26] B. Limthanmaphon and Y. Zhang. Web service composition with case-based reasoning. In *ADC '03: Proc. of the 14th Australasian database conference*, pages 201–208, Australia, 2003. Australian Computer Society, Inc.

J.A. Recio-García is an assistant teacher at the Complutense University of Madrid, Spain. He has been working there for 5 years. He is a computer engineer since 2003 and he is currently working in his Phd, supervised by 3rd and 4rd authors. He is the main developer of the jCOLIBRI framework.

B. Díaz-Agudo is a lecturer at the Computer Science Faculty of the Universidad Complutense de Madrid, Spain. She is a computer engineer since 1996, and received her Phd from the UCM in 2002. She has been working at this university for more than 10 years and has published over forty scientific international publications. She has been program committee, reviewer, attendee and organizer of multiple scientific events. His current research interests include Case Based Reasoning and Semantic Web.

P. González-Calero is a lecturer at the Computer Science Faculty of the Universidad Complutense de Madrid, Spain. He is a physicist since 1992, and received his Phd from the UCM in 1997. He has been working at this university for more than 15 years and published over sixty scientific international. He has been program committee, reviewer, attendee and organizer of multiple scientific events. His current research interests include Semantic Web, Software Engineering, and Game Learning technologies.

J.M. Gomez-Berbis is a Visiting Professor at the Department of Computer Science in the Universidad Carlos III de Madrid. He has obtained his PhD in 2006 from the Digital Enterprise Research Institute (DERI) at National University of Ireland, Galway. He received his Master Thesis in Software Engineering at the Swiss Federal Institute of Technology (EPFL) in Lausanne (Switzerland) and an Msc. in Telecommunications Engineering from the Universidad Politécnica de Madrid (UPM). His current research interests include the Semantic Web, Semantic Web Services, Business Process Modeling, formal methods for eCommerce and eBusiness.