CONFIGURING ONLINE PROBLEM-SOLVING RESOURCES WITH THE INTERNET REASONING SERVICE

Monica Crubézy,[1] Wenjin Lu,[2] Enrico Motta,[2] and Mark A. Musen[1]
[1]*Stanford Medical Informatics, 251 Campus Drive, Stanford CA 94305-5479, USA.*
email: {crubezy, musen}@smi.stanford.edu
[2]*Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, UK.*
email: {e.motta, w.lu}@open.ac.uk

Abstract: Existing services on the World-Wide Web tend to be "integral." For instance, online services for data analysis are available, but usually it is neither possible to modify the underlying reasoning system, nor to configure it for a different domain, nor to integrate different services to produce new functionalities. Our research goal is to develop the technological framework needed to provide sophisticated online problem-solving resources, configurable for different applications. We describe the design and initial implementation of the Internet Reasoning Service—a Web-based front-end enabling developers to prototype knowledge-based applications quickly out of reusable components in distributed libraries. Our aim is to make reliable problem-solving technology available to a wider audience and to provide the level of intelligent support needed to allow rapid generation of Web-based reasoning services.

Key words: Problem-solving methods, ontologies, reusable components, knowledge-based systems, knowledge acquisition, adapters, Web services.

1. INTRODUCTION

Many resources on the Web are now dynamic services that provide some functionality for end users, such as booking a plane, and analyzing gene data. These services, however, tend to be "integral:" usually, it is neither possible to modify the underlying reasoning system of these services, nor to configure it for a different domain, nor to integrate different services to produce new functionalities. The necessary technologies and frameworks for service identification, configuration and interoperability on the Web still remain to

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35602-0_35

M. A. Musen et al. (eds.), *Intelligent Information Processing*

be developed. To foster this objective, the competence of each Web service must be specified, so that brokering agents can locate services, match them to user needs and configure them to realize specific user tasks. This approach is central to the IBROW project[6] [2]. In this context, we focus on configurable reasoning services: our goal is to develop the methodology and tool support needed to provide sophisticated problem-solving facilities (e.g., a heuristic-classification technology) on the Web, which different users (e.g., archeologists, biologists) can configure for specific applications. Note that our approach differs from the recently appeared "Web services," that tend to offer only predefined, limited reasoning functionalities, very specific to a given domain context (see Section 4).

Our approach stems from research on knowledge-based system (KBS) development by reuse [1, 3, 4, 6, 7, 11]. This area has identified domain-independent problem-solving strategies, or *problem-solving methods (PSMs)* [1, 6, 7, 11, 12], that provide standard ways of addressing stereotypical knowledge-intensive problems, or *generic tasks* [4], such as diagnosis, design, and classification. To foster the reuse of PSMs, structured *libraries* have been developed, in which the methods are indexed and retrieved for different domains and purposes [1, 3, 6, 11-13]. Furthermore, comprehensive frameworks take a PSM-centered view to support KBS development from reusable components [7, 11, 14, 17]. This process involves modeling a user problem as a generic task, and configuring an appropriate PSM, by integrating the generic components with domain-specific information.

This reuse-centered approach is now the most established and robust paradigm in knowledge engineering. However, the level of actual component reuse is still quite low, due to several pragmatic barriers: libraries of problem-solving components are few and far between, often they are not available online, and different libraries are not interoperable, neither at the knowledge level nor at the symbol level. Furthermore, existing frameworks for KBS development do not offer sufficient guidance for users to select and configure the reasoning component of their system.

We describe here the *Internet Reasoning Service (IRS)*, a Web-based front-end that enables developers to prototype knowledge-based applications quickly out of reusable components in distributed libraries. Based on a well-defined process model, the IRS overcomes the limitations of KBS-development frameworks by providing semi-automated support for each step of this application-configuration process. We have implemented the IRS as a Web-based server, integrated with the OCML reasoning platform [11] and the generic Protégé-2000 knowledge-modeling environment [14]. In this paper

[6] http://www.swi.psy.uva.nl/projects/ibrow/home.html

we provide an overview of the IRS and we show how it was used to produce a classification problem solver for the domain of Roman pottery archeology.

2. THE INTERNET REASONING SERVICE

Problem-solving libraries contain reusable software components, which can be configured into a running application to solve users' problems. These components are primarily tasks and PSMs, which model the reasoning building blocks of a KBS. For example, we have developed a library of reusable components for heuristic classification problem-solving [12]—the task of matching an unknown object, denoted by a set of 'observables,' to a 'solution space' of classes of similar objects. The IRS supports users in operating a set of distributed libraries of problem-solving resources to achieve tasks in their domain. The IRS provides different levels of support, from interactive browsing facilities for the manual selection and configuration of reasoning resources, to intelligent, semi-automated assistance in building an executable application.

The IRS relies on the knowledge-level descriptions of the problem-solving components in the libraries. The *Unified Problem-solving Method-development Language* (UPML)[7] is a Web-compatible framework that we have designed in IBROW for modeling the reusable components of a knowledge system [15]. UPML distinguishes three kinds of components: (1) *tasks*, which are functions that a KBS can achieve (e.g., classification), (2) *PSMs*, which are methods that implement the reasoning process of a KBS to realize a task (e.g., an optimal heuristic classifier), and (3) *domain models*, which are views on domain knowledge, as used by tasks and PSMs (e.g., archeology). Each component (e.g., a domain model) defines a particular *ontology*—a specification of the important concepts involved in the component model (e.g., the notion of 'solution space' for the classification task). PSMs and tasks specify a set of *input* and *output roles*. Each PSM also defines a functional *competence* in the form of a set of logical *preconditions* and *postconditions*. For instance, the competence of an optimal heuristic classifier states that the output solutions are optimal with respect to the input solution space. Each task is further characterized with a logical *goal expression* and a set of *assumptions* on domain knowledge. These components of a KBS can be configured and combined together in different running systems through the creation of explicit architectural elements—*adapters* [7]. In particular, *bridge* adapters connect two kinds of components, such as a PSM and a domain model, by way of *mapping relations* between

[7] http://www.cs.vu.nl/~upml/

the ontologies of both components. This way, a bridge between the optimal heuristic classifier PSM and the archeology domain maps the notion of 'abstractor' introduced by the PSM and a data-abstraction function for archeology observables.

The process of configuring the reasoning component of a KBS involves several activities: mapping generic tasks and PSMs to a domain model (e.g., mapping the generic classification task to a database of archeological artifacts to produce an artifact-classification application), mapping PSMs to tasks (e.g., selecting a particular abstraction method for performing the data-abstraction subtask of classification), or, in general, adapting existing components (e.g., specializing the data-to-solutions matching component of the classification process by introducing fuzziness in the matching process). More precisely, we have identified a clear process model for the IRS, that involves eight steps, from task selection to application execution (*Figure 1*). The manual completion of each step requires significant skills from users: the goal of the IRS is to guide users through the entire configuration process.

In the following, we detail each step of the IRS process model. Although our approach is agnostic of any specific problem-solving library, we illustrate the use of the IRS with examples taken from our experiment in configuring components of a classification problem-solving library [12] for the archeology domain. In this domain, the user is an archeologist who wishes to develop automated support for classifying Roman-pottery artifacts, based on information collected about the artifacts.

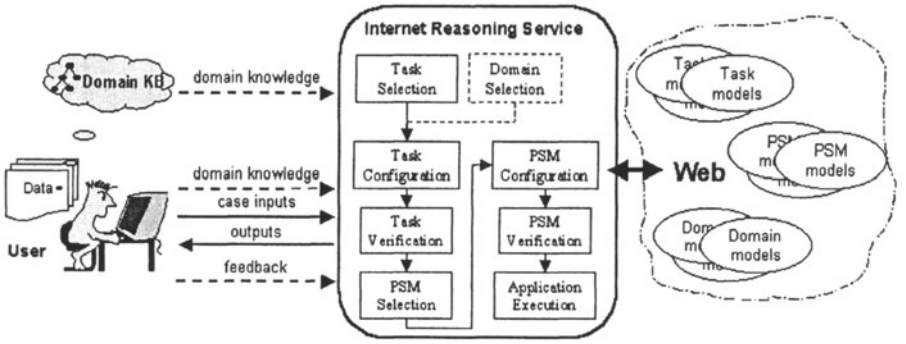

Figure 1. The IRS: use scenario and process model.

Task selection. IRS users first need to characterize the goal of their application in terms of one of the tasks known to the IRS, e.g., *classification.* Task selection is hard: it means adopting a particular viewpoint to impose over an application. For instance, the solutions to the benchmark Sisyphus-I

office-allocation problem [9] included both i) approaches that modeled the problem as a design problem and ii) approaches that modeled it as a classification problem. In other words, a specific application does not necessarily fall into a particular task model. Rather, it is a task model that provides a conceptual viewpoint for characterizing an application and for focusing the knowledge-acquisition process. The IRS supports users in browsing and navigating UPML-based descriptions of the various tasks in the available libraries, along with examples of the ways tasks have been applied previously. Users select a task by choosing a refinement of a high-level task (e.g., *optimal-solution classification*).

Task configuration. In this crucial step, users configure the selected task (e.g., *single-solution classification*) for their particular domain (e.g., archeology) by providing relevant domain knowledge to fill the input–output roles defined by the task. The IRS provides support adapted to the understanding that users have of the configuration process, allowing for a direct reuse of pre-existing configurations or more advanced experimentation. First, the IRS supports the direct acquisition of the value of an input role, according to the task ontology. For example, users can define a new *match criterion* (a mechanism to assess the degree-of-fit between data and solutions) in terms of the task ontology. Alternatively, users simply can select a default value for the input role from the library. Finally, if domain knowledge does not conform to the task ontology, the IRS supports users in constructing a *mapping relation* between the task role and domain knowledge. A domain-task mapping relation defines the transformation of a piece of domain knowledge into an instantiated role for the task [16]. In our experiment, we mapped the role *solution-space* to a hierarchy of archeological artifact types, and the notion of *observables* to attributes of artifacts, such as 'provenance,' 'material,' etc. In the application-execution step, these attributes are instantiated by case-data values. Output roles clearly are not instantiated with actual knowledge structures at this stage; however, mappings also may be required for outputs to conform to the domain ontology. Creating explicit architectural elements that isolate the configuration knowledge for a set of components maximizes the reusability of these components. However, this activity requires that users understand the connections and adaptations needed on the knowledge components to be assembled into a working system. The IRS reduces the complexity of the configuration process by providing structured mapping templates that users instantiate for their particular case. Nevertheless, more research is needed to ease the configuration process further.

The IRS generates a *task-domain bridge* for the relevant task and domain ontologies, to store the set of mapping relations created during the task-

configuration step. The IRS allows users to backtrack to the task-selection step if their domain does not match the expectations of the task (e.g., the task expects single-valued observables when domain observables are multi-valued), and to select another refinement of the high-level task.

Domain selection. In this phase, users specify the domain knowledge that they intend to use in their application. The IRS provides a range of pre-existing domain models stored in available UPML-compliant libraries, from which users simply can make a selection. The advantage of selecting an available domain model is that, in many cases, no configuration effort will be needed, given that mappings themselves are often reusable. In our archeological experiment we can see that the mappings discussed earlier tend to be generic (observables ⇒ artifact features; solution space ⇒ hierarchy of artifact types). Alternatively, the IRS enables users to provide domain knowledge directly by filling-in knowledge-acquisition forms. Another possibility is that users provide their own knowledge base. In this case, users need to solve the related interoperability problems, both at the knowledge and at the symbol level (see Section 4).

Task verification This step consists in checking the assumptions that the selected task defines on domain knowledge. The IRS performs this step automatically by running an assumption-checking engine on the task input roles—possibly obtained from domain inputs through mapping relations—and the assumptions of the task. The IRS notifies users about the inputs that do not satisfy the assumptions of the task. In this case, the IRS guides users back to the task-configuration step, so that they can re-specify the erroneous inputs. However, it is important to note that not all assumptions are necessarily verifiable. For instance, our task model for *single-solution classification* includes the assumption that only one solution exists in the target domain. Clearly this assumption cannot be verified in the general case.

PSM selection. This step focuses on selecting a PSM that can realize the configured task. Similarly to the task-selection step, the IRS provides users with the list of PSMs that match the configured task and supports users in browsing the description of each PSM. The IRS uses different means to compute the list of PSM candidates for realizing the task. First, the UPML description of the library may include a set of *PSM–task bridges*. Each bridge encodes a connection between a task and a PSM that can realize that task, together with a set of mapping relations between the task and PSM ontologies. Such a bridge connects the *single-solution classification* task and the *heuristic-admissible-solution classifier* PSM in our classification library. Alternatively, the IRS carries out a competence-matching process, by reasoning about the competence of PSMs and the goal of the current task. In

general, such reasoning requires full first-order logic theorem-proving support, which is not part of the current implementation of the IRS. Finally, users themselves can choose among all available PSMs in the library.

PSM configuration. This step is similar to the task-configuration step, at the level of the selected PSM: the IRS guides users in specifying the domain entities that fill-in the input–output roles of the PSM. Some of the roles for the PSM are "inherited" from the configured task, through a corresponding *PSM–task bridge*. If not already provided in the library, the IRS supports the creation of such a bridge to map the inputs and outputs of the configured task to the ones of the selected PSM. In addition, the selected PSM may define supplemental roles. For example, the *heuristic-admissible-solution classifier* PSM defines the notion of an *abstractor*—a function that computes abstract observables from raw data. In our archeology domain, we defined such a function to abstract from the specific geographical site in which the artifact was found (i.e., the 'provenance' feature) to a more coarse-grained area, more useful to classify the artifact. The IRS supports the acquisition of domain-method mapping knowledge in a way similar to the domain-task mapping during task configuration. The IRS finally stores the result of the PSM configuration as a *PSM-domain bridge* created for the user's domain.

PSM verification. This step focuses on verifying that the selected PSM can be applied to the task and the domain in accordance with the assumptions of the PSM and the results of the configuration process. This step is essentially the same as the task-verification step. For instance, in the case of our chosen heuristic classifier, the assumption-checking engine will verify that the solution hierarchy (i.e., the hierarchy of artifact types) contains no cycles. As a result, the IRS notifies users about the domain inputs that do not satisfy the assumptions or preconditions of the PSM. In this case, the IRS guides users back either to the task-configuration step or to the PSM-configuration step, to re-specify the inputs that are not satisfied.

Application execution. This final step consists in running the configured PSM to realize the specified task, with domain case data entered by the user. The IRS first acquires case data from the user and instantiates the case inputs of the PSM by interpreting the domain–task, task–PSM and domain–PSM mapping relations. The IRS also checks the preconditions of the PSM and task on the mapped case data. The IRS then invokes the PSM code with the mapped inputs, by running a code interpreter either locally or remotely. Knowledge about the location and type of PSM code is stored in *pragmatics* fields of the UPML description of the PSM. Finally, the IRS fills-in the domain outputs with the results of PSM execution, possibly transformed with domain–PSM mapping relations defined at PSM-configuration time.

3. INITIAL IMPLEMENTATION OF THE IRS

We have implemented the IRS design as a prototype tool, which exploits knowledge, Web and HCI technologies already available to our two groups. We first have implemented the IRS generic design in Java, as a Web front-end based on the OCML modeling language [11]. We also have implemented the IRS design as a service integrated to the widely-used Protégé-2000 knowledge modeling environment [14].

The IRS Web-based front-end. Our first implementation of the IRS capitalizes on knowledge and Web technologies available at the Knowledge Media Institute, mainly the OCML modeling and execution language [11] and a large library of problem-solving components, compliant with the UPML framework, and accessible online through a knowledge-level API. We use OCML to describe both the UPML properties of knowledge components and their actual contents. The IRS is directly connected to the LispWeb server, an online interpreter that processes OCML requests.

We have developed an initial implementation of the IRS, which supports all activities in the process model described in Section 2. Browsing support and ontology-driven knowledge-acquisition facilities enable users to instantiate role values directly, or to select pre-existing ones from a library. The IRS supports mapping through pre-defined OCML constructs, as well as 'free-form' constructs for expressing arbitrary mappings [11]. The IRS also provides library-specific high-level mapping templates, such as a 'Map to Hierarchy' template for the classification library, which allows the user to simply state that a particular domain hierarchy instantiates the role 'solution space'. The IRS also supports assumption-checking and a limited form of competence matching, which relies on pre-existing relations between tasks and PSMs. Finally, the IRS supports the application-execution step by invoking the OCML environment to interpret the PSM code on the case data inputs that the user enters in special-purpose forms.

Integration of the IRS design with Protégé-2000. We also have implemented the IRS design as an extension to Protégé-2000[8], a tool developed at Stanford for ontology development and ontology-driven knowledge-base construction [14]. Our IRS extension interfaces domain knowledge bases to libraries of tasks and PSMs, specified in UPML with a Protégé-based UPML editor [15]. This extension implements the user interface of the IRS with the ontology-driven, graphical knowledge-acquisition support of Protégé, to enable users to select and configure application components.

[8] http://protégé.stanford.edu

Most importantly, Protégé brings additional support to the IRS for the two configuration steps, by providing a structured methodology for mapping roles of reasoning components to corresponding domain entities. Our methodology provides a typology of mapping-relation templates—a *mapping ontology* [16], which covers a wide range of mapping relations, from simple renaming mappings, to complex numerical or lexical expressions of entities. Protégé guides users in the configuration steps based on this mapping ontology and the input–output definition of the task or PSM. In addition, Protégé incorporates a *mapping interpreter* [16], which runs the mapping relations with the domain entities to create PSM-level entities.

The IRS extension of Protégé is connected to the IRS Web-based server, to which it delegates the steps of task and PSM-verification, as well as the execution of the configured PSM with case data that users enter in knowledge-acquisition forms.

4. DISCUSSION

Our research is related to existing work in the area of Web and knowledge technologies. Our comparison highlights the strengths of our approach together with the main issues still to be addressed.

Configurable Web services. Recent industrial efforts aim to provide online, configurable services on the Web. They address the syntactic-level interoperability of Web services with standardized protocols for component-functionality description and communication (e.g., UDDI[9], WSDL[10], .Net[11], Jini[12]). Software components can thus be discovered, usually by a centralized look-up service. Although these distributed Web services can cooperate to achieve a certain goal, they typically provide only predefined functionalities.

Semantic approaches stem from research in earlier agent technology. Web services declare their capabilities and requirements, and special-purpose agents mediate those services to users by locating, matching and connecting them [18]. Shared ontologies of Web services and plan-like procedures, described in the recently-designed DAML-S[13] language, further enable agents to perform tasks for users, by dynamically chaining primitive steps realized by Web sites [10]. These services, however, implement relatively fine-grained, ad-hoc procedures that solve limited problems. Furthermore, such approaches assume that users and services share the same domain of

[9] http://www.uddi.org/
[10] http://msdn.microsoft.com/xml/general/wsdl.asp
[11] http://www.microsoft.com/net/
[12] http://www.sun.com/jini/
[13] http://www.daml.org/services/

discourse, ignoring the issue of mapping domain, task and method ontologies. Thus, such Web services are less reusable in different domains.

Our goal also is to provide configurable services on the Web. However, the services that we provide are complete, robust, and reusable methods for achieving common knowledge-intensive tasks in various domains. Our user-centered, ontology-driven approach to component configuration provides a full framework to model problem solvers, that the IRS then uses for constructing knowledge-based applications for different contexts.

Competence matching. Currently we do not address adequately the issue of retrieving PSMs that can realize the user-specified task. The list of possible candidate PSMs can be computed by a competence-matching process. This process involves reasoning over the competence of both the configured task and the available PSMs: candidate PSMs are selected if their postcondition statement fulfills the goal of the task, and if their preconditions do not contradict the assumptions of the task. Agent-based approaches such as RETSINA include an efficient competence-matching process, which retrieves agents, based on the services that they advertize in the LARKS language [18]. However, LARKS provides relatively weak representational machinery to express the formal competence of agents. A similar approach is under investigation by other members of the IBROW consortium. An important issue is to design contents-specification formalisms, which strike the right balance between expressiveness and efficiency in support of competence matching. The IRS must also address the possible mismatch between the ontologies of the task and of the available PSMs. Although usually not as different as the ontologies of domains and tasks, the ontologies of tasks and PSMs sometimes need to be mapped through explicit relations, which in turn need to be interpreted as part of a competence-matching process.

Interoperability. Component interoperability is at the center of the IRS—and, more generally, the IBROW—approach. Typically, each problem-solving library uses a specific knowledge-modeling language to express properties of its components, and includes pieces of code in a particular programming language. The federated UPML framework alleviates this difficulty only at the modeling level. Knowledge-level interoperability requires shared ontologies and either common modelling languages or the use of standard knowledge-level APIs, such as OKBC [5]. Protégé brings to the IRS its support for OKBC-compliant languages. At the execution level, configured components—including possible domain knowledge provided by users in a proprietary format—need to be assembled in an interoperable system. The use of distributed-computing standards, such as CORBA,[14] has

[14] http://www.corba.org/

proven to be a successful means for encapsulating problem-solving components and combining them into platform-independent, operational systems [8]. Indeed our initial IBROW broker prototype used a CORBA-based architecture to operate heterogeneous components [2] and we have also implemented a generic CORBA-based interface to the IRS, that enables library providers to make their components available to the IRS.

5. CONCLUSION

Our IRS approach strongly builds on proven knowledge-engineering techniques. The key asset that the IRS adds to these techniques is a principled methodology and user-centered tool support for prototyping knowledge systems by configuring reusable components from structured libraries. Although some symbol-level and knowledge-level interoperability issues remain to be addressed, we believe that the IRS approach will foster a wider diffusion of knowledge-engineering paradigms, by lowering the current conceptual and engineering barriers to applying them.

The Web vehicle will allow us to perform a large evaluation experiment of our IRS approach, that promises to simplify system construction by component reuse. We plan to evaluate aspects such as speed up in system construction and exploitation, efficacy and utility of built systems and straightforwardness of the application-building process for domain users. At the same time, the heterogeneous, distributed and versatile nature of the Web will challenge us to incorporate more automated support in the IRS, with simplified procedures and customized interactions, so that less-experienced users also can benefit from advanced distributed problem-solving services.

Current users of the IRS are reasonably skilled developers, who wish to prototype a KBS application, or simply explore KBS technology. Over time, we aim to broaden the target audience of the IRS, by making it possible for less-experienced developers to create applications with the IRS. Finally, although the current focus of our research is on facilitating user access to sophisticated reasoning services, which involve heterogeneous ontologies, we expect to harness the technologies produced in this research in support of more 'mundane' levels of service provision on the Web. In these cases we can take advantage of scenarios in which ontology mapping and service interoperability issues are greatly simplified.

ACKNOWLEDGEMENTS

This work is part of the IBROW Project, funded by the IST program of the European Community. We thank our IBROW collaborators for the many

stimulating discussions on the topics discussed in this paper. Finally, we thank our two anonymous reviewers for their helpful comments.

REFERENCES

1. Benjamins, V.R. Problem Solving Methods for Diagnosis, University of Amsterdam, 1993.
2. Benjamins, V.R., Plaza, E., Motta, E., Fensel, D., Studer, R., Wielinga, R., Schreiber, G., Zdrahal, Z. and Decker, S., An intelligent brokering service for knowledge-component reuse on the World-Wide Web. in *KAW'1998*, (1998).
3. Breuker, J.A., and van de Velde, W. (ed.), *The CommonKADS Library for Expertise Modeling*. IOS Press, Amsterdam, 1994.
4. Chandrasekaran, B. Generic tasks for knowledge-based reasoning: High-level building blocks for expert system design. *IEEE Expert, 1* (3). 23-30.
5. Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P., OKBC: A programmatic foundation for knowledge base interoperability. in *AAAI'1998*, (Madison, Wisconsin, 1998), AAAI Press/The MIT Press, 600-607.
6. Eriksson, H., Shahar, Y., Tu, S.W., Puerta, A.R. and Musen, M.A. Task modeling with reusable problem-solving methods. *Artificial Intelligence, 79*. 293-326.
7. Fensel, D. and Motta, E. Structured Development of Problem Solving Methods. *IEEE Transactions on Knowledge and Data Engineering, 13* (6). 913-932.
8. Gennari, J.H., Cheng, H., Altman, R. B., Musen, M.A. Reuse, CORBA, and Knowledge-Based Systems. *International Journal of Human-Computer Studies, 49* (4). 523-546.
9. Linster, M. Problem Statement for Sisyphus: Models of Problem Solving. *International Journal of Human-Computer Studies, 40* (2). 187-192.
10. McIlraith, S.A., Son, T.C. and Zeng, H. Semantic Web Services. *IEEE Intelligent Systems, 16* (2). 46-53.
11. Motta, E. *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. IOS Press, Amsterdam, 1999.
12. Motta, E. and Lu, W., A Library of Components for Classification Problem Solving. in *PKAW'2000*, (Sydney, Australia, 2000).
13. Musen, M.A., Modern Architectures for Intelligent Systems: Reusable Ontologies and Problem-Solving Methods. in *AMIA Annual Symposium*, (Orlando, FL, 1998), 46-52.
14. Musen, M.A., Fergerson, R.W., Grosso, W.E., Noy, N.F., Crubézy, M. and Gennari, J.H., Component-Based Support for Building Knowledge-Acquisition Systems. in *Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000)*, (Beijing, China, 2000).
15. Omelayenko, B., Crubézy, M., Fensel, D., Benjamins, V.R., Wielinga, B.J., Motta, E., Musen, M.A. and Ding, Y. UPML: The Language and Tool Support for Making the Semantic Web Alive. in Fensel, D., Hendler, J., Liebermann, H. and Wahlster, W. eds. *To appear in: Creating the Semantic Web*, MIT Press, In press.
16. Park, J.Y., Gennari, J.H. and Musen, M.A., Mappings for Reuse in Knowledge-Based Systems. in *Eleventh Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, (Banff, Alberta., 1998).
17. Schreiber, A.T., Akkermans, J.M., Anjewierden, A.A., de Hoog, R., Shadbolt, N.R., van de Velde, W., and Wielinga, B.J. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, Cambridge, 2000.
18. Sycara, K., Lu, J., Klusch, M. and Widoff, S., Matchmaking among Heterogeneous Agents on the Internet. in *AAAI Spring Symposium on Intelligent Agents in Cyberspace*, (Stanford, CA, 1999).