MDSS: A framework for the integration of ontology mapping tools

Gabriele Marcelli
IASI - CNR
Viale Manzoni, 30
00185 Roma, Italy
Email: marcelli@ing.univaq.it

Tania Di Mascio
University of L'Aquila
Monteluco di Roio
67040 L'Aquila, Italy
Email: tania@ing.univaq.it

Fulvio D'Antonio
IASI - CNR
Viale Manzoni, 30
00185 Roma, Italy
Email: dantonio@iasi.rm.cnr.it

***Abstract*— In the last period there has been a proliferation of tools that support users in discovering mappings among given ontologies in an automatic or semi-automatic way. However the way such tools are developed is unstandardized: they usually differ in input/output formats, type of user interaction and external resources exploited, hindering the possibility of making them interoperate (for example composing different mapping algorithms). In this paper we propose MDSS (Mapping Discovery Support System) a framework for the integration of mapping tools, aiming at solving interoperability issues among different mapping discovery tools and at supporting users in choosing the best suited mapping discovery algorithm for their needs.**

***Keywords*— ontology alignment, ontology mapping, mapping discovery, service-oriented architecture.**

I. Introduction

The Semantic Web proposes a common framework that allows data to be shared and reused across different applications, enterprises and community boundaries. In the current web, resources (such as documents, web pages, etc.) contain data expressed in a machine-readable, but not machine-understandable form. For the Web to scale, programs must be able to share and process data even when these programs have been designed totally independently. The Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people. To this end ontologies play a crucial role that allow the formalization of semantics of information and enable interesting reasoning operations such as verification of consistency, equivalence and so on.

However, as it is plausible that a single universal ontology can not be built, we must expect an explosion in the number of ontologies, even when considering the same domain. For these reasons, a key challenge in building the Semantic Web is to enable interoperability among different ontologies. Ontologies can interoperate only if correspondences between their elements have been identified and established. At the moment, if a problem of this type is encountered, the mapping is mainly achieved by hand. But this task is tedious, error-prone, and time consuming; therefore the manual solution of the ontology interoperability problem is likely to become a bottleneck in building a network of cooperating information management systems. Hence, introduction of new methodologies and user-friendly tools that support users (knowledge engineers) in discovering semantic correspondences is crucial to the success of the Semantic Web. For instance, in order to discover semantic correspondences, knowledge engineers generally must possess a high degree of technical skill, and a deep knowledge of algorithms employed.

Our system aims at solving the issue of interoperability among different mapping discovery tools and it also aims at supporting users in choosing optimal mapping discovery algorithms.

The paper is structured as follows: in Section II we present the problem of ontology mapping discovery and the various issues connected with it; in Section III we propose our abstract framework highlighting the architecture and its information flow; in Section IV we give an implementation of the framework using Web Service technology; a brief state of the art is described in Section V.

II. The Issues of Mapping discovery

In this work we deal with the mapping discovery process, that is the process of finding mappings between two ontologies; it is, in other words, the identification of

relations holding among the entities in the two ontologies (concepts and properties equivalence/subsumption, similarity degrees etc.). The process is also known in literature as ontology matchmaking ([3]), ontology alignment ([6]), or ontology mapping ([16]), where the key terms *mapping* and *alignment* have been defined in [6] as follows: mapping is "*a formal expression that states the semantic relation between two entities belonging to different ontologies*", and alignment is a set of correspondences between two ontologies expressed as mappings.

However, as stated before, performing mapping discovery is a complex process, and as long as it will require a deep human intervention it will be highly expensive and hardly scalable.

Therefore, the hope is to automatize as much as possible the process (see for instance in [16] and [18]) and that is the main reason for proliferation of tools that support users in discovering mappings among given ontologies; they usually make possible computation of alignments in an automatic or semi-automatic way - see [12] and [16].

These tools may be characterized by the following function (see [6]):

$$A' = f(O_1, O_2, A, P, R)$$

where A' is the result alignment, O_1, O_2 are the source ontologies, A is some previously known alignment between O_1 and O_2, P is a set of parameters, and R is a set of external resources used in the process.

We have chosen to adopt these definitions in order to abstract from implementation details of mapping discovery tools and to have a unifying view about their features; moreover, this characterization of mapping discovery tools is a first step toward definition of a "mapping discovery algebra" (e.g. formally representing the iteration and composition of mapping discovery tools).

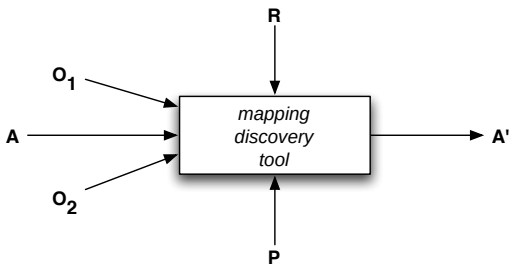

Fig. 1. The mapping discovery tools schema

Figure 1 describes the Input/Output schema of mapping discovery tools, but it does not highlight that alignments (A, A') represented by tools and parameters/resources (P, R) of tools are not standard; therefore composition of mapping algorithms from different tools is hard, when not impossible at all. Moreover the choice of mapping discovery strategies is not supported by tools, and user interaction strongly differs among them; in other words, the focus of mapping discovery tools (as clarified for instance in [16]) is on the performance of mapping discovery, it is not on users (the knowledge engineers).

We conclude highlighting two open issues of mapping discovery process we deal with:

- from the system point of view: the lack of a common platform to compose mapping algorithms from different tools;
- from the user point of view: the lack of a strategy to support users in performing mapping discovery process.

III. Our proposal: Mapping Discovery Support System (MDSS)

We propose a framework called MDSS (*Mapping Discovery Support System*) to support users in the mapping discovery process using algorithms supported from different tools; this framework is an intermediate layer between mapping tools and knowledge engineers; MDSS cooperates with existing tools to compute alignments (it *is not* another mapping discovery tool).

MDSS aims at solving the two open issues mentioned in Section II, providing the following main features:

- it handles low-level details regarding input and output of each tool;
- it supports users in the choice of mapping discovery algorithms.

Low-level handling of inputs and outputs makes easier composition of algorithms from different tools; moreover, this also simplifies access to supported mapping tools, providing a single and uniform interaction modality.

The choice of mapping algorithms requires a high level of technical knowledge; in order to simplify this choice, we propose a new approach: the mapping algorithm chosen is obtained thanks to a set of qualitative parameters. More details of this approach are described in Section III-B.

A global picture of our system is sketched in Figure 2.

Let us explain the main concepts exposed in the diagram:

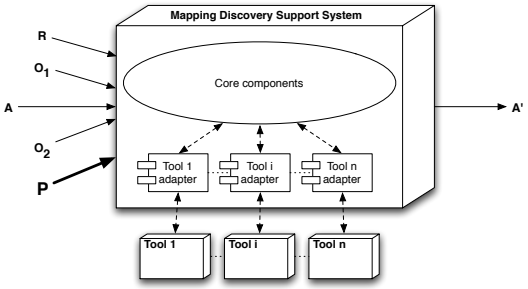

Fig. 2. A system to support the mapping discovery process

- MDSS cooperates with existing tools to compute alignments; access to external tools is mediated through the *adapter pattern*[13] (one adapter component for each tool),
- inputs and outputs depicted in Figure 2 have the same meaning of inputs and outputs of Figure 1 except P, that here represents not only tool parameters, but also parameters of the MDSS system.

MDSS parameters are used to determine how our system works; for instance, they are used to choose mapping discovery algorithms, or to select particular formats for alignment representation.

A. MDSS Architecture

We have designed MDSS architecture according to **modularity** and **extensibility**. Modularity means that each component not only is independent, but it also has definite boundaries and specific goals. Extensibility means that it is easy to extend the set of supported mapping discovery tools; in fact, it is enough to create a plug-in component as described below.

Main components of MDSS architecture, shown in Figure 3, are: the *Control Subsystem*, the *Tools and Algorithms Registries*, the *Algorithm Chooser Module*, the *Alignment Formats Handling Subsystem*, and the *Mapping Discovery Tools Adaptation Layer*.

The **Control Subsystem** is the core of MDSS. It manages operations, components and resources, and it plays the role of *front controller* to manage external requests to the system. It contains the main application controller that interacts with other components of the system.

The **Tools and Algorithms Registries** are two repositories; the first is the *Tools Registry* that contains all the knowledge that MDSS has about integrated tools and the second, the *Algorithms Registry*, contains all the knowledge that MDSS has about algorithms of each tool. In particular, the *Tools Registry* contains the identifier of tools in the system, the set of algorithms that it provides, and informations about supported alignment formats; the *Algorithms Registry* contains the unique identifier of an algorithm, its description, metadata about its features, and metadata about parameters and resources managed. The main purpose of the *Tools Registry* is to keep track of tools integrated into MDSS, and it is used to interact with them. The main purpose of *Algorithms Registry* is to store metadata informations about algorithms used by the *Algorithm Chooser Module* and the *Control Subsystem*.

The **Algorithm Chooser Module** implements the logic supporting mapping algorithms choice; this choice is made according to users requests: the *Algorithm Chooser Module* extracts knowledge about algorithms from the *Algorithms Registry*, where it is expressed as sets of qualitative parameters; then these parameters are compared to the set of preferences expressed by users. Finally, according to users selections, the module chooses the algorithm.

Actually, algorithms are classified on the basis of techniques employed, in a way approximately similar to the classification made in [12]. Algorithms choice is then made comparing users preferences with features of each algorithm, in a tabular fashion. We would like to remark that our main concern in this work has been defining our framework, while development of optimal implementations of its components will be addressed in future work; in particular, aspects relative to mapping algorithms classification and selection are one of our main lines of future research, as remarked later in Section VI.

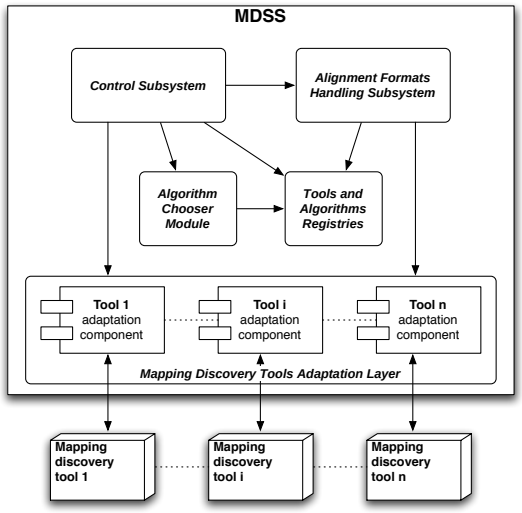

Fig. 3. Architecture of MDSS

The **Alignment Formats Handling Subsystem** manages alignment representations. This functionality is essential because no affirmed standard exists (some proposals are in [11]). This subsystem offers both automatic and on-demand management of alignment representations, performed exploiting the *Mapping Discovery Adaptation Layer*.

The **Mapping Discovery Adaptation Layer** is an abstraction layer from mapping discovery tools; in other words MDSS components interact only with this layer, they never interact directly with tools. The *Mapping Discovery Adaptation Layer* is composed of several modules, one for each tool supported. Each module is called *Pluggable Mapping Discovery Module* (or PMDM), and each module is responsible of encapsulating characteristics of a single mapping discovery tool. Unlike components described above, PMDMs are not static entities of the system; they are dynamically discovered and loaded into MDSS at startup time. They are plug-ins, thus making MDSS easily extensible in supporting mapping tools; in other words it is enough writing a PMDM to add support for a specific tool. A **Pluggable Mapping Discovery Module** (PMDM) encapsulates distinctive characteristics and behaviour of a specific tool. In particular, each module represents both *semantics* and *functionalities* of a particular tool.

At semantic level, a PMDM provides:

- information about tools - entries in *Tools Registry*,
- metadata about algorithms provided by tools - entries in *Algorithms Registry*,
- information about alignment format of tools - used by *Aligment Formats Handling Subsystem*.

At functional level, a PMDM provides:

- a standard, tool-independent, communication interface for higher level components,
- low-level interaction with mapping tools,
- low-level handling of tools alignment formats.

Summarizing, a PMDM is the basic component needed to integrate tools into the MDSS architecture. To add a new mapping discovery tool, it is enough developing a new PMDM module and plugging it into MDSS. In order to do that, the new tool has to support automatic alignment computation.

B. Information flow

We will now describe the information flow among MDSS components. We may divide this flow into two main phases:

1) boot-up phase;
2) ready phase.

BOOT-UP PHASE

First of all, the *Control Subsystem* looks for available PMDMs. For each PMDM found, the *Control Subsystem* extracts metadata about integrated tool from the PMDM, and it adds them to the *Tools Registry*; then, it builds the list of algorithms provided by the tool. For each algorithm in this list, metadata is extracted and added to the *Algorithms Registry*. Now, MDSS is ready to accept requests.

READY PHASE

When MDSS is ready, the *Control Subsystem* waits for requests. This request may contain a specific mapping algorithm name, or it contains a set of preferences that MDSS has to use in order to choose this algorithm.

In order to determine it, the *Control Subsystem* sends the set of preferences to the *Algorithm Chooser Module*. This component examines the preferences, and it selects in the *Algorithm Registry* the algorithm that better matches them. The algorithm found is then sent to the *Control Subsystem*.

Once MDSS knows which algorithm to use, the mapping process starts. First of all, the *Control Subsystem* asks to the *Algorithms Registry* which tool implements the chosen algorithm. Then, it queries the *Tools Registry* to gather informations about the tool.

Using these information, the *Alignment Formats Handling Subsystem* converts the input alignment to the tool specific alignment format. The request is then dispatched to the PMDM of the tool. At first, the PMDM filters out inputs not supported by the tool; at second, the PMDM invokes the tool that actually computes the alignment. The alignment is then converted by the *Alignment Formats Handling Subsystem* to the desired alignment format and the *Control Subsystem* returns it.

IV. MDSS IN A SOA CONTEXT

The design of MDSS described above is quite general; we did not specify any implementation detail, and in particular how such an architecture could be instantiated in a specific context. In fact, we have designed MDSS structure to be independent of its actual realization, be it a stand-alone application, or a web application, etc.

An interesting possibility is instantiating it in the context of *Service-Oriented Architecture* (SOA) using Web Service technology. In such an architecture, software functionalities are represented as discoverable services on the network; every function is defined as

an independent service, with a well-defined invokable interface. Main benefits of SOA are independence from development technologies and platforms, high degree of interoperability, easy integration of different services, on-demand composition of simpler services to perform complex tasks.

In this context, our goal is using the MDSS architecture to provide a mapping discovery and ontology alignment service.

Since its beginnings the semantic web has been thought explicitly as "*an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users*" ([2]). But such an environment requires software agents to someway understand each other, even without being expressly designed to work together. When semantics is expressed using different ontologies, some kind of reconciliation service is needed, in order to enable agent interoperability without human intervention. To summarize: *semantic web needs semantic coordination*.

A mapping discovery service could provide some of this coordination; when two semantically enabled agents use different vocabularies - that is, different ontologies - they can ask the service to obtain correspondences between them, enabling cooperation without human interaction.

As an example, let's suppose an user entrusting his personal agent to make weekly reports on books about themes of his interest. The user would express his request through a specific ontology; the agent would in turn query online books resellers - like for instance Amazon.com or Barnes&Noble - whose books would be described, in general, with different ontologies. Exploiting a mapping discovery service, the agent could translate the request of the user into the vocabulary of the reseller, obtaining the desired result.

Of course, to realize a such a service one might simply expose as web service some existing mapping tool; but this would lead to a different mapping discovery service for each mapping discovery software, with poor interoperability among them.

Instead, using the MDSS architecture as base, we have a single service able to use mapping discovery capabilities of a variety of tools. This is not restrictive, since MDSS is designed to interact with remote mapping tools, so it can even take advantage of other similar services.

The scenario is represented in Figure 4. The core MDSS architecture can be easily exposed as web service, realising the mapping discovery service we described above. Interaction with the system is so extended to other software, and not only to human users. Remote mapping discovery tools are also supported, since as depicted only the tool's PMDM is local to MDSS.

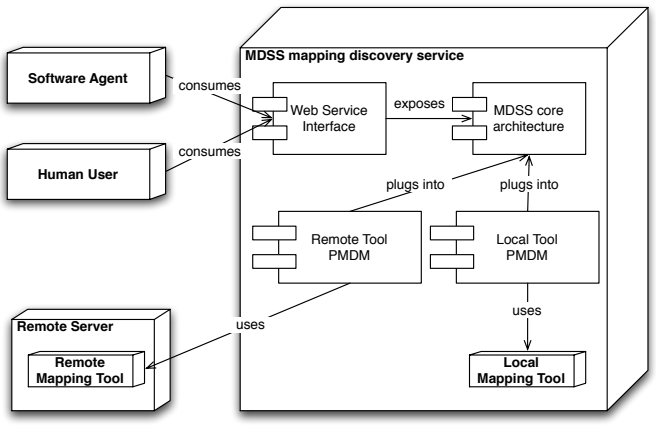

Fig. 4. MDSS as a service

V. Related work

As described in Section II, mapping tools solve the mapping discovery process issue, but it is worth noting that there are few systems in literature explicitly designed to solve the main issue we deal with: mapping discovery tools interoperability.

About this issue, the INRIA Alignment API[11] is maybe the most remarkable proposed solution. INRIA proposes an RDF-based alignment representation format and a Java API to handle it. Main goals of INRIA Alignment API are:

- composition of alignment algorithms,
- iterative mapping process, through reuse of previously computed alignments,
- mapping discovery algorithms modularization.

The main focus of the INRIA system is to define an alignment representation format and to provide an API to work with it; in our system, we used this format due to its flexibility; in fact it is easy (using the provided API) to convert the INRIA format to a large number of other formats.

Moreover, differently from INRIA, our system explicitly supports integration of "external" algorithms as described in Section III-A. In the INRIA tool, in fact, in order to use an external algorithm, it has to be implemented using the provided API.

MDSS aims at playing the same role in the mapping discovery field as the GATE system [7],[4],[5] plays in

the Natural Language Processing (NLP) field; as GATE represents a framework for NLP, that provides a framework of reusable Language Engineering components, MDSS represents a framework for mapping discovery.

It is worth also referring to several mapping discovery tools studied, in view of the fact that each of them possess ideas (e.g., iterative mapping discovery) supported by our MDSS. They are FOAM [9],[10], CROSI Mapping System (CMS) [15],[14], and COMA++ [8],[1].

FOAM is a mapping discovery tool developed by University of Karlsruhe. Similarly to our system (as described in Section III-B), FOAM aims at supporting the choice of mapping strategies using some parameters called "scenario" and "strategy".

CMS is a mapping discovery system developed by University of Southhampton and HP Laboratories. Similarly to our system, CMS supports existing mapping tools integration; in fact it integrates the INRIA Alignment API and FOAM implementing an ad-hoc wrapper for each of them. MDSS, instead, supports existing mapping tools integration using PMDMs. This approach (described in Section III-A) allows supporting tools with automatic alignment computation.

COMA++, developed by University of Leipzig, is a mapping tool proposing a composite approach to the mapping discovery issue: different algorithms are combined, using computed alignments as input to next iterations of mapping process. MDSS supports the same mapping process approach, naming it iterative mapping discovery.

VI. Conclusion and future work

In this paper we have presented MDSS (Mapping Discovery Support System), a framework for mapping tools integration, aiming at solving interoperability issues among different mapping discovery tools and at supporting users in choosing the best suited mapping discovery algorithm for their needs. These issues were only partially addressed by existing solutions in literature. We have presented MDSS as an abstract architecture, and proposed a possible implementation based on Web Service technology. In the future our research about MDSS will concentrate along two lines:

- **creating new MDSS integration patterns**, in order to easily integrate an increasing number of Ontology mapping tools
- **enhancing the mechanism for supporting the user choice of an optimal mapping algorithm** (with respect to users mapping problems); this requires development of a more sophisticated Decision Support System (DSS) being able to match users problem descriptions with MDSS integrated tools.

References

[1] D. Amueller, H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with coma++," in *SIGMOD*, June 14-16 2005.

[2] T. Berners-Lee, J. Handler, and O. Lassila, "The semantic web," *Scientific American*, pp. 28–37, May 2001.

[3] D. Bianchini, S. Castano, F. D'Antonio, V. D. Antonellis, M. Harzallah, M. Missikoff, and S. Montanelli, "Digital resource discovery: Semantic annotation and matchmaking techniques," in *Proc. of the Interoperability for Enterprise Software and Applications Conference (I-ESA 2006)*, Bordeaux, France, March 2006.

[4] K. Bontcheva, D. Maynard, V. Tablan, and H. Cunningham, "Gate: A unicode-based infrastructure supporting multilingual information extraction," in *Proc. Workshop on Information Extraction for Slavonic and other Central and Eastern European Languages, Borovets, Bulgaria*, 2003.

[5] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, "Evolving gate to meet new challenges in language engineering," *Natural Language Engineering*, 2004.

[6] P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Kroetzsch, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris, "Knowledgeweb deliverable 2.2.1 - specification of a common framework for characterizing alignment," Knowledge Web Consortium, Tech. Rep., 2005.

[7] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "Gate: an architecture for development of robust hlt applications," in *Proc. of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2001.

[8] H. Do and E. Rahm, "Coma - a system for flexible combination of schema matching approaches," in *Proc. of the 28th VLDB Conference, 2002*, 2002.

[9] M. Ehrig and S. Staab, "Qom: Quick ontology mapping," in *Proc. of the International Semantic Web Conference (ISWC)*, 2004, pp. 683–697.

[10] M. Ehrig and Y. Sure, "Ontology mapping - an integrated approach," in *Proc. of the European Semantic Web Symposium (ESWS)*, 2004, pp. 76–91.

[11] J. Euzenat, "An api for ontology alignment," in *Proc. 3rd international semantic web conference*, Hiroshima(JP), 2004.

[12] J. Euzenat, J. Barrasa, P. Bouquet, J. D. Bo, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. V. Acker, and I. Zaihrayeu, "Knowledgeweb deliverable 2.2.3 - state of the art on ontology alignment," Knowledge Web Consortium, Tech. Rep., 2004.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[14] Y. Kalfoglou and B. Hu, "Issues with evaluating and using publicly available ontologies," in *Proc. of 4th International EON Workshop, Evaluating Ontologies for the Web, Edinburgh, UK*, May 2006.

[15] Y. Kalfoglou, B. Hu, D. Reynolds, and N. Shadbolt, "Crosi project, final report," School of Electronics and Computer Science, University of Southhampton, Tech. Rep., 2005. [Online]. Available: http://eprints.ecs.soton.ac.uk/11717/

[16] Y. Kalfoglou and M. Schorlemmer, “Ontology mapping: the state of the art,” *The Knowledge Engineering Review*, 2003.
[17] J. Madhavan, E. Rahm, and P. A. Bernstein, “Generic schema matching with cupid,” in *Proc. 27th VLDB Conference*, Roma, 2001.
[18] N. Noy, “Semantic integration: a survey of ontology-based approaches,” *ACM SIGMOD Record*, 2004.
[19] P. Shvaiko and J. Euzenat, “Ontology matching initiative.” [Online]. Available: http://www.ontologymatching.org
[20] ——, “A survey of schema-based matching approaches,” *Journal on Data Semantics*, 2005.