Ontology Management: a Case Study and Research Plans

Jeroen Hoppenbrouwers, Manfred A. Jeusfeld, Hans Weigand, Willem-Jan van den Heuvel

{hoppie,jeusfeld,weigand,wjheuvel}@uvt.nl
CRISM/Infolab, Tilburg University,
P.O.Box 90153, 5000 LE Tilburg, The Netherlands
http://infolab.uvt.nl/

Abstract. We foresee that ontologies for interoperability are likely organized by a loosely linked network. In the case of digital libraries, the network is realized by a federation of ontologies where cross-links are made on demand. For the future, we plan to work on two extensions. The first one is to develop a new paradigm for synchronization of ontologies, called imitation. The second one is about publishing existing information sources by an ontology that is scalable in its size. Local parts of the ontology can be extended by global parts, which provide more meaning to the concepts. Finally, we are continuing our research on reverse engineering for integrating legacy components.

1. Federated ontology management: a case study

In 2000, the Conference of European National Librarians (CENL) [2] initiated a project to investigate the possibilities of loosely linked subject heading languages (SHLs) [3]. Four national libraries (Bibliothèque Nationale de France, The British Library, Die Deutsche Bibliothek and the Swiss National Library) teamed up with Index Data ApS and Tilburg University to create a demonstration prototype of a multilingual search engine, a SHL link maintenance system, and an organisational workflow to get all partners into routine maintenance of the link database. The project is called MACS, for Multilingual ACess to Subject headings. It has evoluated into a full production system an will be routinely used by many organisations [4].

A Subject Heading Language (SHL) is a specialised, artificial minilanguage consisting of a vocabulary and syntax rules to express categories in a library system [5]. These elaborate controlled vocabularies are a key resource for the ongoing indexing of media collections. They are maintained by specific, often national authorities and usually restricted to a specific natural language. Because these SHLs have their own identity, they vary considerably in extent, vocabulary, and composition rules, and because huge amounts of material are indexed with the subject headings of a specific SHL, these SHLs resist any significant change. Efforts have been started in the late 1990s to cross-link a few major SHLs using manual methods [6]. The MACS project was initially provided with the result of this work and therefore started out with a database of approximately 1000 links between the French, English and German major SHLs (RAMEAU, LCSH, and SWD) in the areas of Sports and Theatre. Since then, large amounts of cross-links have been added to the database, either manually or by bulk loading from external data sets. Currently the database contains more than 30 000 links.

With SHLs being very different in nature, cross-linking them is not a formal or even straightforward translation. The primary target of the translation of a SHL expression in another SHL is to yield the same results when the translated expression is used to query the same collection of materials indexed in the other SHL [3,7]. Perfect matching is nearly impossible, but a good approximation can be obtained. With the identity and absolute independency of the participating SHL authorities as a primary goal, the only feasible organisational model of the link database was a federation. The participating authorities link expressions in their own SHL to expressions in other SHLs on a voluntary basis. There is no central authority, as this would imply a central SHL. MACS partners can propose links to each other's SHL as they see fit, but retain absolute sovereignity over the links between their own SHL expressions and the central records. This means that the central records function as an artificial semi-union of all participating SHLs. We organised the MACS federation to be scalable to a reasonable number of participating SHLs (our aim was about 50, the current number of European National Libraries) with 100 000 subject headings per SHL. This meant that we had no technical scalability problem in terms of database size, but instead an organisational problem. Adding one new SHL should not lead to extra work for all the currently federated SHLs (e.g., to review the new link and certify its validity to each individual SHL). By creating a strict system of automated ToDo and ToKnow lists, a strict technical link validation procedure, and direct access to integrated annotation and resolution mechanisms, we created a technical support infrastructure to keep the process running when maintenance activity levels rise [7].

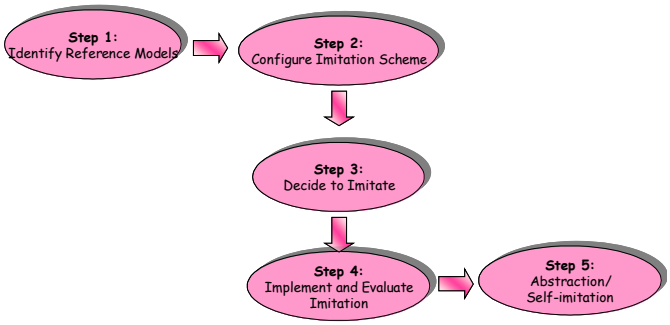
With any federation, mutual understanding and a cooperative attitude towards the common goal are paramount. In MACS, we initially faced four partners who were committed to the project, but the project target is to extend the group with, eventually, dozens of SHL authorities world-wide. This requires a firm organisational and technical community infrastructure, not only for the semi-formal link management, but also to support the requirements of a growing federation of significantly different organisations. Using standard ICT "community plumbing" tools, which are by now readily available as well-maintained OpenSource resources, we set up sufficient infrastructure to support the complete MACS community as such. This support infrastructure itself is part of the ongoing community development. [8]

2. Ontology maintenance by imitation

Several studies show that the maintenance of systems, already make up about 70-80 % of all costs during an enterprise application's lifecycle. In order to cut down costs of rather tedious and labor-intensive routine maintenance tasks, and establish more effective mechanisms to deal with change, autonomic computing is touted in industry as an effective distributed computing solution. In a nutshell, autonomic computing provides a contemporary paradigm for managing computing resources, eliminating the need for human interference. This is achieved by giving enterprise systems not only full awareness about their internal model, but, also the ability to adapt themselves.

Our current research scrutinizes the application of various concepts from the domain of Complex Adaptive Systems and Autonomic Systems [9,10], and combines them with the notion of imitation. Our claim is that self-adaptation based on imitation can also be applied effectively to ontology maintenance.

Imitation is a powerful concept and an inherently complex one. In the figure besides, a framework for imitation is presented. In a fully autonomic system, the system could perform all the steps itself, but we assume that for the time being, human intervention will be needed to set up the imitation process. In the following, we describe briefly the first 2 steps.



Step-1: identify reference model(s)
If you want to imitate, you must have a reference model to imitate, so imitation always starts with the identification of models. The models can be hard-coded by the user or be determined dynamically. An example of a hard-coded model is when a local database is supposed to replicate a particular central database. Another example of the use of a hard-coded model is a local business ontology that regularly consults a standardization web site and executes any update available. Dynamic identification of models can be done, e.g., by using the transitivity property of imitation. That is, use the models that are used by the peers.

Step-2: Configure imitation scheme
Once a model is determined, the information flow may be designed. The imitating system can ask (actively), or observe (passively). In the case of passive imitation, the model system should somehow advertise meta-data about its behavior. For example, it should send a notification when it has done something interesting, together with an URL where the action can be seen. Or it can just broadcast all its actions to whoever is interested by publish/subscribe. It can be done instantly or at regular time intervals. In general, it seems better not to assign the initiative to the imitating system, as it cannot know whether something interesting has happened, but if the imitating system is triggered by some urgent need, and wants to learn something, it should have the capability to take initiative. For example, suppose an XML-based message handler is confronted with a data type that it doesn't know. Then it could go to a reference model (a peer, or a standardization group web site), and ask whether it has a definition, and could decide then to copy it. Apart from the initiative and the timing, important to determine is also *what* to imitate. Imitation is always imitation of some relationship: e.g. between a subject and an object (possession) between a word and its referent (meaning), or between a goal and an instrument, or between a situation and a goal (desire). By imitating meanings, as in the case of ontologies, systems can extend their knowledge of the world both the conceptual level and on the instance level. As there are so many relationships that one could imitate, it is necessary that the imitation scheme describes the object of modeling in a normalized way. As a first attempt, we propose to distinguish the following attributes:

- *relationship* - an action or static relationship that is to be imitated;
- *reference* - the fixed term (joint attention), an entity or system component that exists in the context of the reference model and in the context of the imitating agent;
- *object* - the thing to be copied, indicated by a variable and a type;
- *conditions* - a declarative specification of restrictions on the objects of imitation;
- *goal* - the goal to which the imitated action or relationship is supposed to contribute.

For example, suppose we want to imitate the way products are identified, then the relationship is "has-key", the referent is "product" (a concept), and the object is "attribute X". The goal is "identify object uniquely". This means that when the reference model changes the way products are identified (candidate key), then the agent will also consider this attribute as a candidate key. To apply the imitation mechanism effectively to ontology maintenance, a goal analysis of ontologies is needed first. This analysis should make clear what are the elements of an ontology and what are their goals. For example, the goal of a key is a unique reference. This goal analysis is part of our short-term research plan. Once the goal structure of ontologies is known, an agent can replace a certain means by another means (found by imitation) that has the same goal.

3. Substituting schemas by ontologies for information access

Ontologies have been named as a tool for bridging the gap between heterogeneous systems. Information, in particular in the form of databases, is only understandable in the context in which it has been created and used. This context is partly reflected in the schema (=syntax of the data) and the history of the data (=semantics of the schema). The reflection is however not complete as some of the semantics is buried into the code of the application programs, or even only exists as tacit knowledge. Our approach for supporting seemless information access is to completely hide the schema of an information source and substitute it by ontologies that either

1. contain no domain knowledge at all but only ontological concepts of the data model (like entity, attribute, relation, key), or
2. are equivalent to the original schema (lossless reconstruction, see [1]), or
3. are containing arbitraly much knowledge about the application domain.

The very idea of our approach is to decompose any object (here: any tuple of a relation) into its atoms, being the object's *membership to a set* (here: membership of a tuple identifier to a relation), and the object's *relations to other objects* (here: the attributes describing an object in a tuple). Set membership corresponds to the semantics of an ontological concept like 'Car'. The object's relations (or attributes) are reified (i.e. they are objects themselves) and seen as members of attribute sets (such as denoted by 'size', 'color', 'predecessor'). In the case of absence of domain knowledge, the ontological concepts into which we map a data source are those of the data model of the data source, here: tuple, attribute of a tuple, key attribute etc. The mapping of any data source with a given schema to such a simple ontology is done by derivation rules. While this variant of schema substitution is very simple, it still allows formulating queries to any data source regardless of their schema. For example, one can ask for are objects that are related to a given object. In the second case, the equivalent substitution, any concept of the schema is associated to a counterpart in the ontology. The original relations can be reconstructed by just referring to set membership and attribute relations in terms of the application-specific ontology. This case is not adding extra value but only a uniform query facility.

The interesting case is when the ontology is containing more concepts than the original schema, in aprticular more concepts for describing attributes of objects. Since the original schema is substituted by an ontology, this ontology can be extended to provide specializations, generalizations, and aggregations of the original schema concepts. From the semantics viewpoint, the substitution of the schema commits the ontology concepts to the instances. From an application development viewpoint, the ontology replaces the schema for query definition.

4. Morphing legacy systems into new business applications

A critical challenge in designing robust business applications is allowing to *selectively* identify reusable and modifiable portions of a legacy system, and, to combine them with modern enterprise components in a gradual and consistent manner [11]. A contemporary vision for tackling this challenge, entails OMG's Model Driven Architecture (MDA) [12], placing the notion of models at the

core of virtually all phases of application (re-)development. In particular, MDA claims that conceptual, computational independent enterprise models (CIMs) may be transformed in Platform Independent Models (PIMs), that may in turn be converted into Platform Specific Models (PSMs), e.g., for J2EE or the .NET platform. Transformations thus constitute an essential mechanism to realize the MDA philosophy, allowing the automatic generation of one or more target models from one or more source model, given a transformation definition (using imperative or declarative languages) [13]. Unfortunately, the MDA predominantly addresses forward engineering, emphasizing forward transformations, of new enterprise applications into code. However, we firmly believe that MDA provides an interesting foundation, including some (ECLIPSE-based) tools and techniques, to also deal with even more challenging reverse engineering aspects. Our current research efforts are bundled in INTEROP's Model Morphism (MoMo) workpackage, involving the INFOLAB (University of Tilburg) and the ATLAS-group (University of Nantes). Within the context of this workpackage, we aim at amortizing the MDA with more advanced transformation operators. In particular, our research efforts are directed at investigating new and improved ways for model transformations so that they can not only handle with forward engineering of enterprise models (into applications), but also with reverse engineering of legacy systems into conceptual models. This research will ideally result in a range of transformations that extend and refine MDA's current rather mechanical, fully automatic, and one-shot transformations (both imperative and declarative).

In fact, this new breed of model transformations will not any longer be restricted to *vertical* transformations for generating code from models and vise versa, but also, and more importantly, will involve a broad range of semi-automatic, heuristic-based, *horizontal* transformations that assist the designer of a new enterprise application in matching its design to a set of available legacy system models, and, adapting legacy models to overcome syntactic and semantic model mismatches, both at the level of semantics and syntax. This work will build forward on research as reported in [11] and [14]. We plan to embed these transformations into a model-driven legacy integration methodology, encompassing phases for forward engineering of new enterprise applications, which reuse reverse engineered fragments of legacy systems, based on a solid and concise matching and adaptation process. Lastly, the methodology will be equipped a phase that offers model morphism-based techniques for enabling pro-active change management of application models.

5. References

[1] M.A. Jeusfeld: Integrating product catalogs via multiple-language ontologies. In Proc. EAI 2004 Workshop Enterprise Application, Oldenburg, February 12-13, 2004, http://CEUR-WS.org/Vol-93, ISSN 1603-0073.

[2] CENL http://www.bl.uk/gabriel/about_cenl/

[3] P. Landry et.al.: MACS project prototype final report. http://laborix.uvt.nl/prj/macs/pub/MACSreport3.pdf, 2002.

[4] P. Landry: MACS Update: Moving toward a Link Management Production Database. In: ELAG 2003 proceedings. http://www.elag2003.ch/papers/MACS-ELAG-article.pdf

[5] Subject Indexing: Principles and Practices in the 90's, ed. by Robert P. Holley, et al. UBCIM Publications - New Series, v.15. München: K.G. Saur, 1995.

[6] G. Clavel et.al. (1999): CoBRA+ Working Group on Multilingual Subject Access, Final Report. http://www.ddb.de/gabriel/projects/pages/cobra/finrap3.html

[7] J. Hoppenbrouwer: Architecture of the MACS system. http://laborix.uvt.nl/ prj/macs/pub/architecture.pdf, 2001

[8] MACS main web site: https://macs.cenl.org/

[9] A.G. Ganek and T.A. Corbi, The Dawning of the Autonomic Computer Area, *IBM Systems Journal*, 42 (1): 5-18, 2003.

[10]. J.O. Kephart, D.M. Chess The Vision of Autonomic Computing *Computer* 36(1):41-50, IEEE,2003.

[11] W.J. vanden Heuvel: Integrating Modern Business Applications with Legacy Systems: A Web-Component Perspective. MIT-Press, To be published in 2005.

[12] OMG: MDA guide 1.01. http://www.omg.org/docs/omg/03-06-01.pdf, Visited: June 2004.

[13] A. Kleppe, J. Warmer, and W. Bast: MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

[14] E. Rahm and P.A. Bernstein: A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases}, 10(4):334--350, 2001.