

Agent-based Ontology Integration for Ontology-based Applications

Li Li

Baolin Wu

Yun Yang

Faculty of Information and Communication Technologies
Swinburne University of Technology,
PO Box 218, Hawthorn, Melbourne, Australia 3122,
Email {lli,bwu,yyang}@it.swin.edu.au

Abstract

In this paper, a novel agent-based ontology integration framework is developed for agents which consume ontologies in ontology-based applications as well as engage in tasks of ontology integration. The corresponding ontology integration mechanism is discussed. Derived ontologies can be reused in the system. A prototype is built by using the JADE agent platform for evaluation.

Keywords: Agent, ontology integration, consistency checking.

1 Introduction and Motivation

An ontology is defined as an explicit specification of a conceptualisation (Gruber 1993). It is a formal description of a domain of discourse, intended for sharing among different applications, and expressed in a language that can be used for reasoning (Noy 2003). Ontologies facilitate the interoperability between heterogeneous systems involved in commonly interested domain applications by providing a shared understanding of domain problems and a formalisation that makes ontologies machine-processable. Furthermore, ontologies are seen as key enablers for the emerging Semantic Web. It is known that any information system uses its own ontology, either implicitly or explicitly. Proliferation of Internet technology and globalisation of business environments has given rise to the advent of a variety of ontologies which are far beyond expectations. Thus it is unlikely that everyone conforms to a single ontology because of technical and non-technical reasons. On the other hand, it is a trend that different systems combine to achieve a goal by taking advantage of existing resources or integrating available systems to avoid error-prone and costly reinvention. All these may take place at unpredictable times, for unpredictable reasons, between unpredictable organisations. Agents in multi-agent systems (MAS) operate flexibly and rationally in environments which are dynamic and heterogeneous, given that agents have abilities to perceive changes of environments and respond promptly. A MAS perspective is thus suitable for tackling ontology integration within and across the boundaries of organisations. The aim of this paper is to develop a novel agent-based framework to conduct ontology integration based on a certain business scenario and prior work (Li, Yang & Wu 2005b).

Two major architectures for ontology integration have been investigated (Noy 2003). The first one is a general upper ontology agreed upon by users of different applications, while the second one is a method comprising heuristics-based or machine learning techniques that use various characteristics of ontologies. These two approaches focus on ontology management and have been taken by many researchers in information integration. However, ontology integration, as part of the ontology development process (Pinto & Martins 2004), is far more complex than expected. Considerable effort is needed in ontology reuse (Uschold, Healy, Williamson, Clark & Woods 1998). We refer the reader to an excellent and thorough review (Kalfoglou & Schorlemmer 2003) for a detailed discussion in this field. In terms of contributions from database community, we refer the reader to a survey (Rahm & Bernstein 2001). Also other researchers (Calvanese, Giacomo & Lenzerini 2002, Klein & Noy 2003, Noy 2004, Wache, Vögele, Visser, Stuckenschmidt, Schuster & Neumann 2001) provide overviews of ontology integration. Some of the specific challenges in ontology integration that must be addressed in the near future are listed in (Noy 2003). Although the problem of specifying the architecture is the core of ontology integration, it has not been thoroughly investigated yet. It is a great challenge to perform ontology integration as flexibly as possible as required on the Web. In terms of flexibility, agent technology fits well in developing applications in a dynamic and distributed environment which requires substantial support for change. A MAS approach is thus ideally suited in handling ontology integration in an open environment such as Web.

To this end, an agent-based framework is developed. By following the framework, agents' behaviours in interaction diagrams are presented. A key feature of our framework is its flexibility and extendibility in ontology integration by allowing ontology reuse in the system.

This paper is organised as follows. Section 2 introduces the terminology of ontologies and presents an ontology integration scenario. Section 3 presents an agent-based integration framework and mechanisms. Section 4 describes the prototype and evaluation. Section 5 contains the related work and discussion, and finally, Section 6 addresses conclusions and future work.

2 Ontology Definition and Scenario

Agents act on different ontologies from different sources. These agents in a certain business scenario might contact each other to work together to solve the problems that are beyond the individual capabilities or knowledge. It is not surprising that there are some terms used inconsistently when people talk about ontology integration and even ontology itself. We

introduce ontology definition and terminology used throughout this paper next. Under the ontology specification, example ontologies will be presented.

2.1 Ontology Definition and Terminology

It is most likely that different organisations opt to choose terminology according to their understandings and requirements. Hence, terms such as mapping and integration may have been used in differing ways. Because a consensus about the meanings of the terms in this field is unlikely, we need to specify terms that we use throughout this paper. The following definition and terminology will specify terms used in this paper.

In this paper, we follow Gruber’s best known ontology definition (Gruber 1993). Under this definition, we define an ontology \mathcal{O} with a specific domain model, \mathcal{T} . Thus a conceptualisation Σ is a pair of $\langle \mathcal{C}, \mathcal{R} \rangle$, where \mathcal{C} represents a set of concepts, and \mathcal{R} stands for a set of relations over these concepts. A specification is a pair of $\langle \Sigma, \Psi \rangle$ to describe that Σ satisfies the axioms Ψ derived from the domain model. In the following, notation $\mathcal{C}(\mathcal{O})$ is used to annotate concepts \mathcal{C} of the ontology \mathcal{O} .

There are many representations and languages available for encoding an ontology, however to establish the notation of ontology used in a MAS internally for the task of ontology integration, an **Entity-Relation (E-R)** data model is considered to encode an ontology, where concepts are regarded as classes. A typical concept class will have an identifier that distinguishes from others, and a set of attributes that describes the properties of the concept class. Then it is feasible to compare two concepts by looking at the identifier as well as the attributes. Below are three kinds of mutually exclusive semantic relations between existing concept classes from two different ontologies. We assume that \mathcal{O}_i and \mathcal{O}_j are in the same domain ($i, j \in \mathbb{N}$, where \mathbb{N} : natural numbers). c_i where $c_i \in \mathcal{C}_i(\mathcal{O}_i)$ and c_j where $c_j \in \mathcal{C}_j(\mathcal{O}_j)$ are two different concepts.

Definition (Equivalent): Two concepts are semantically equivalent, if $\exists c_i, c_j$, s.t. $c_i \sim c_j$. Namely, these two concepts: (1) have the same denotation names (e.g. labels); (2) are synonyms; or (3) their attributes are the same.

Definition (Inclusive): Two concepts are semantically inclusive, if $\exists c_i, c_j$, s.t. $c_i \leq c_j$ (e.g. c_i is a kind of c_j) or $c_i \geq c_j$ (e.g. c_j is a kind of c_i). Namely, the attributes of one concept are also the attributes of the other.

Definition (Disjoint): Two concepts are disjoint, if $\exists c_i, c_j$, s.t. $c_i \cap c_j = \Phi$. Namely, there is no common attribute between them.

For the purpose of ontology integration, we need to consider the consistency issue of an integrated ontology. It is obvious that the E-R data model of an ontology and the defined semantic relations between concepts allow us to check the consistency of a newly derived ontology. The ontology consistency is defined as follows.

Definition (Consistent): An ontology is consistent, if $\forall c_i^k, c_i^n, c_i^m$ ($c_i^k, c_i^n, c_i^m \in \mathcal{C}_i(\mathcal{O}_i)$, and $c_i^k \neq c_i^n \neq c_i^m$ ($k, n, m \in \mathbb{N}$), $c_i^k \leq c_i^n$ and $c_i^n \cap c_i^m = \Phi$, s.t. $c_i^k \not\leq c_i^m$). Namely no sub-concepts of a particular concept is also a sub-concept of another concept where these two concepts are disjoint.

Definition (Ontology Mapping): An mapping \mathfrak{R} between two ontologies \mathcal{O}_i and \mathcal{O}_j exists, if $\exists c_i, c_j$, s.t. $\mathfrak{R}(c_i, c_j) \in \{\sim, \leq, \geq\}$. In terms of integration, we will use the following definition throughout this paper.

Definition (Ontology Integration): Reusing available source ontologies within a range to build a new ontology which serves at a higher level in the application than that of various ontologies in ontology libraries. It is associated with semantic integration. Different levels of integration can be distinguished.

2.2 Scenario

The running example ontologies come from the domain of *beer* and concern the types of beer. One is from the DAML ontology library (<http://www.daml.org/ontologies/66>). The second one is built on the definition of the term “beer” from the WordNet (<http://wordnet.princeton.edu/>). The third one is based on basic types of beer provided by the website http://www.dma.be/p/bier/1_2_uk.htm#. The fourth one is Australian beer types ontology based on information from the following websites: (1) http://www.australianbeers.com/beers/beer_types/beer_types.htm; and (2) http://www.fosters.com.au/beer/about/beertypes/beer_types.asp.

Our main interest is on term “beer” and the corresponding hyponym relationship. The scenario such as different wine retailers or even brewers using different beer ontologies is not unusual. In a business, achieving some goals frequently requires more than individual capabilities and knowledge. A general view of a variety of existing ontologies at an abstract level is necessary for achieving such goals.

The above four ontologies are about types of “beer”, but from different points of view. We will come back to them in Section 4.

3 Agent-based Ontology Integration

Agent technology presents an exciting prospect in a field where high dynamics are requested. It has the potential to significantly extend the range of applications that can be feasibly tackled (Jennings & Wooldridge 2001). We have discussed the rationale of an agent-based perspective in (Li, Yang & Wu 2005b) and behaviours of **mapping agent (MA)** in (Li, Yang & Wu 2005a). Next, we first present an agent-based integration architecture. We incorporate ontology reuse in the integration. Then we describe the behaviours of **integration agent (InA)** and its interactions within the integration process. After that, we discuss the integration process. And finally, we detail the integration mechanism.

3.1 Incorporate Ontology Reuse in Integration

Some ontologies in the same area exist with different aspects and overlapping information. Ontologies independently created by individual organisations may need to be integrated later on. Moreover, integration serves ontology reuse in applications. So we should take it into consideration in ontology integration design. Ontology reuse in this paper has two meanings. On one hand, existing ontologies can be used to generate new ontologies (e.g. extending or combining); on the other hand, newly generated ontologies are ready for reuse in the system whenever needed. Ontology integration embedded with ontology reuse is shown in Figure 1.

Figure 1 points out this paper’s focus. Under the proposed architecture, some of the important aspects are highlighted: (1) integration can be done gradually; and (2) derived ontology can be reused instantly by taking advantage of the presence of **ontology agent (OA)**. In other words, whenever a new ontology is derived (based on existing ontologies), an *OA*

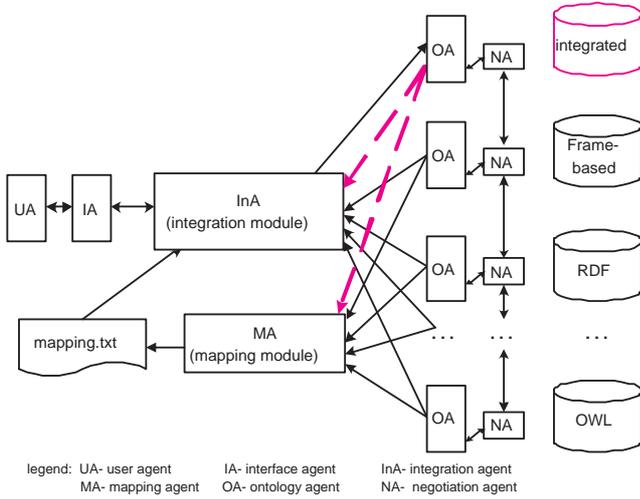


Figure 1: Ontology integration architecture

will be created on the fly to be responsible for it. Ontology integration (module) is based on the mapping results discussed in (Li, Yang & Wu 2005a). Integrated ontology is depicted at the most right top in Figure 1. It can be reused in the system (associated with the integration module in dotted line). Refer to (Li, Yang & Wu 2005b) for details of definitions of other agents in Figure 1.

3.2 Agent Interaction Diagram

At an abstract level, agents achieve a goal by working with other agents to solve problems that are beyond individual capabilities and knowledge. Agents work together based on interactions. An interaction process (e.g. diagram) is needed as interactions between agents may take place concurrently. After investigating the existing tools, we have chosen AUML (<http://www.auml.org/>) to describe the artifacts of agent interactions.

Figure 2 displays interactions in the integration module. *InA* consults *OAs* involved in integration process and counts the number of occurrences. Then it records concepts that meet the requirements. Certainly, a visualisation module of the *user agent* (*UA*) can present a graphic view of the result at the end.

3.3 Integration Process

In this paper, the integration module starts from the root of the specified ontology and then traverses all sub-concepts of the ontologies. Briefly speaking, *InA* counts the appearance of each concept of existing ontologies, and then filters unexpected concepts with a given threshold. By saying this, we do not mean that we attempt to change the conceptual modelling of the ontology. Instead, ontology integration is based on a specified ontology. The process is as follows:

- (1) obtain ontology related information (e.g. mapping results) via *OAs* ;
- (2) keep the numbers of occurrences of each concept in the specified ontology;
Three cases may take place according to the mapping results. They are:

- Case 1 semantic equivalence for the current two concepts (for example, “beer” is the same as “suds”):

In this case, increase the number of occurrences of the concept by 1 for each equivalence;

- Case 2 inclusive relation for the current two concepts (for example, “stout” is a kind of “ale”):

In this case, insert sub-concepts of the counterpart into the specified ontology structure but keep the original relations;

- Case 3 no semantic equivalence for the current two concepts but their corresponding direct ancestors are semantically equivalent:

In this case, insert the counterpart into the specified ontology but without conflict with existing sub-concepts of the same ancestor;

- (3) filter unexpected concepts by a given threshold.

Following the above approach, we derive a newly integrated ontology based on the mapping results. Moreover, as the *OA* is used in the proposed framework to be in charge of ontology related tasks, the integrated ontology can be reused.

3.4 Integration Mechanism

Integration module operates over available mapping results. The module uses the following functions or data structures to execute relevant operations. The pseudocode for the integration algorithm is shown in Table 1.

- **initialise**: initialise the integration process;
- **next-c**: request a particular *OA* the next concept of a specified ontology and returns the concept if it exists;
- **search**: search for relations in mapping results and returns existing relations if it exists or Null otherwise;
- **insert-sup**: insert a specified concept as a super-node of a given concept in a particular ontology structure;
- **insert-sub**: insert a specified concept as a sub-node of a given concept in a particular ontology structure;
- **get-threshold**: contact the *UA* via *IA* to obtain a threshold. It returns the threshold;
- **filter**: filter unexpected items by given threshold from a given ontology and returns a filtered ontology.

4 Prototyping and Evaluation

We have developed an agent-based ontology integration prototype using the JADE platform (<http://jade.tilab.com/>). The application background of the prototype is ontology integration in a certain scenario where agents involved work together to achieve a goal. In that case, an abstract conformance view is needed. After that, the evaluation of the framework and the work we have done is presented.

4.1 Prototyping

Following the analysis of the proposed architecture, we worked out the details of the prototype. It consists of the following agents: one *user agent* (*UA*), one *interface agent* (*IA*), one *mapping agent* (*MA*), one *integration agent* (*InA*), one

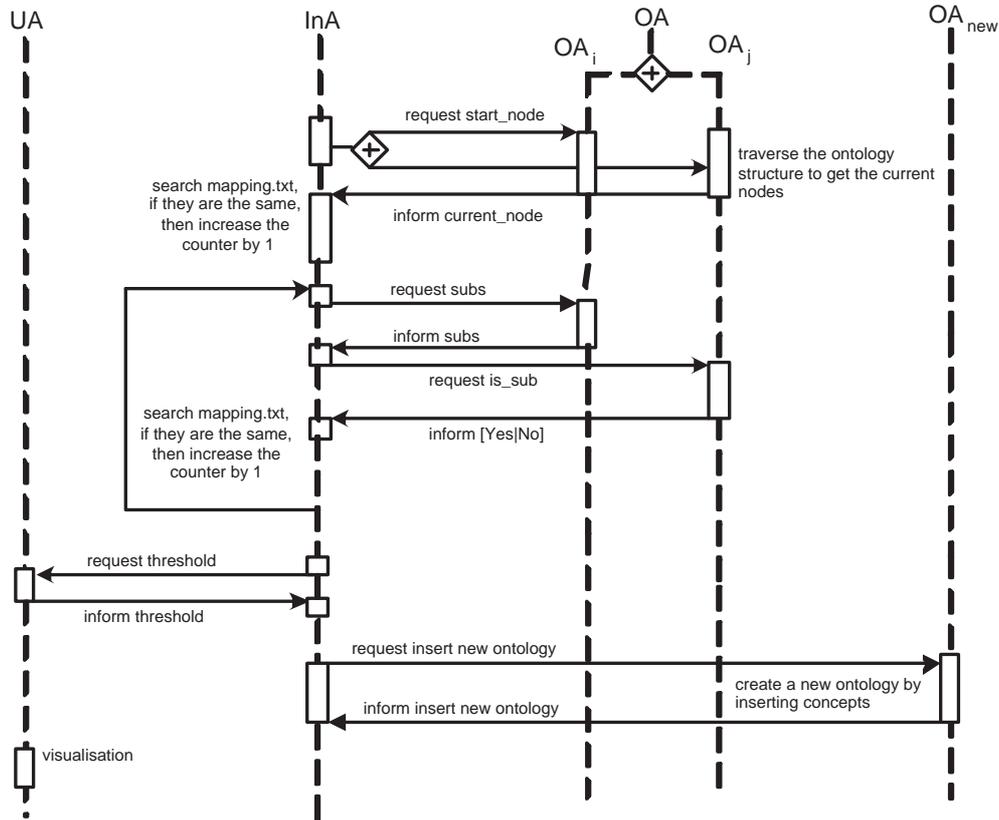


Figure 2: Interactions between agents in *integration* module

consistency checking agent (*CA*), four ontology agents (*OAs*), four negotiation agents (*NAs*).

The prototype runs as follows:

- (1) *Import* existing ontologies;
- (2) *Develop* corresponding *OAs* for each available ontology;

(3) *Execute* mapping module;

(4) *Execute* integration module;

The process may take the following steps if required.

(5) *Visualise* the integrated ontology;

(6) *Export* the integrated ontology in a specified format (e.g. RDF);

(7) *Check* the consistency of the integrated ontology.

Suppose four similar organisations are seeking to achieve a goal that is beyond their individual capabilities and knowledge. They have their own ontologies as presented in Section 2.2. In order to have a general view of the variety of ontologies, a higher abstract ontology is necessary. In the example, we assume that existing four ontologies are integrated based on the WordNet “beer” definition.

Figure 3 is a screen shot of the ontology integration results (see Section 2.2 for existing running examples). The upper part is the overall prototype, the lower right part is the integrated ontology in a hierarchical structure.

Consistency checking is conducted by applying a certain description logic (DL) based reasoning tool. DL-based inference engines, which use a tableau based algorithms (Baader & Sattler 2001), are decidable and support complete consistency checking. In this paper, ontology consistency checking is done with RACER (<http://www.racer-systems.com/>) in Protégé (<http://protege.stanford.edu>).

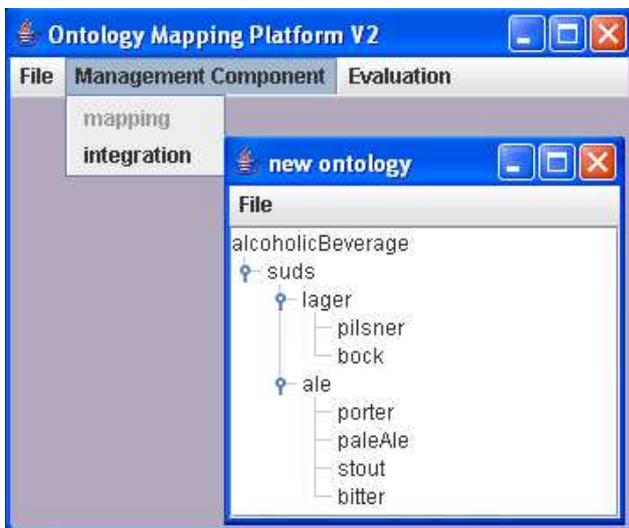


Figure 3: Screen shot of ontology integration

4.2 Evaluation

Ontologies and ontology-based applications perform in the environment of dynamics, distribution and heterogeneity. The agent-based framework proposed in this paper is suitable for tasks such as ontology integration in a certain business scenario. By adopting a MAS’s perspective, interactions among multiple agents, which work together to achieve the goal beyond individual capabilities and knowledge, are highlighted. In other words, MASs are able to take a variety of environmental circumstances into consideration rather than treating the environment mainly as being static. The evaluation work takes the following characteristics into account:

- **Flexibility:** In the framework, the already set up agent communication channel facilitates mes-

Table 1: Integration algorithm

```

/* assume the mapping module starting from a given start point;
 $O_s, O_t$ : two different ontologies;
 $O_d$ : the derived ontology;
 $c_s, c_t$ : concepts from ontologies  $O_s$  and  $O_t$ , respectively;
 $n_{c_s}, n_{c_t}$ : the numbers of occurrences of concepts  $c_s$  and  $c_t$ , respectively;
 $m$ : the number of available ontologies;
 $relation$ : relations between two given concepts from different ontologies;
 $threshold$ : the threshold given by the user to filter unexpected items from the generated
ontology.
*/
Function integration {
  initialise;
  for( $i = 1; i < m; i++$ ) {
    while ( $(next-c(O_s) \neq Null) \ \&\& \ (next-c(O_t) \neq Null)$ ) {
       $c_s = next-c(O_s)$ ;
       $c_t = next-c(O_t)$ ;
       $relation = search(c_s, c_t)$ ;
      switch ( $relation$ ) {
        case "=":  $n_{c_s}++$ ; break;
        case " $\leq$ ": insert-sup( $c_t, c_s$ );  $n_{c_t} = 1$ ; break;
        case " $\geq$ ": insert-sub( $c_t, c_s$ );  $n_{c_t} = 1$ ; break;
        default:  $n_{c_s} = 1$ ;
      } //end switch
    } //end while
  } //end for
   $threshold = get-threshold$ ;
  filter( $O_d, threshold$ );
} //end function

```

sage delivery. Moreover, the presence of the *OAs* allows flexible system organisation. The system allows freely adding/deleting *OAs* and all defined agents for a particular tasks to/from the system;

- **Interactivity:** Agents are highly interactive in the framework. Interactions take place not only between *OAs*, but also between other agents if a particular task needs to deploy the functionalities of others;
- **Interoperability:** The framework enables interoperability between agents of different agent platforms. In terms of syntactic and semantic heterogeneity of ontologies, a meta-ontology (Li, Yang & Wu 2005b) is developed to resolve semantic heterogeneities;
- **Scalability:** In the framework, different classes are developed. They include *concept class*, *ontology class* and *agent class*. Moreover, all ontology related operations are encapsulated and isolated from other agents's view (e.g. only being visible to *OAs*). By extending corresponding classes, the agents can be created easily.
- **Reusability:** In the framework, whenever a new ontology is generated based on existing ontologies, an *OA* is developed correspondingly. It enables the general view of a particular application domain to be reused in the system.
- **Reliability:** It depends on agents performing rationally in the framework. As every agent of the system is required to register and advertise its capabilities to the *IA*, any other agents are able to reach all available capabilities in the system whenever needed. Moreover, the upper bound on the number of iterations (the integration algorithms) required to reach a fixed point is the number of concepts in an ontology. It is known that the number of concepts in an ontology is finite.

Agents generated for the running examples work properly at this stage. They all exhibit their behaviours correctly as specified.

To sum up, the proposed framework provides a flexible and effective modelling approach to tackle ontology integration over a variety of ontologies.

5 Related Work and Discussion

Independently developed ontologies may need to be integrated or composited later on if required. It is natural to treat it from a process perspective. Pinto et al. (Pinto & Martins 2001) define activities of the process. A methodology is presented to support and guide the process. Even though each stage of the integration process still needs to be addressed in the future, we can find some existing tools which are claimed to support it in some way. Some of these systems are: PROMPT (Noy & Musen 2003), Chimaera (<http://www.ksl.stanford.edu/software/chimaera>), SHOE (<http://www.cs.umd.edu/projects/plus/SHOE>), and OntoEdit (<http://www.ontoknowledge.org/tools/ontoedit.shtml>). The PROMPT suite consists of a set of tools to assist merging, alignment and versioning of ontologies. These tools support some of the tasks in the context of multiple ontology management. Chimaera (McGuinness, Fikes, Rice & Wilder 2000) is an environment for merging and testing large ontologies. SHOE (Heflin 2001) allows Web page authors to annotate their web documents with machine-readable knowledge. OntoEdit (Sure, Erdmann, Angele, Staab, Studer & Wenke 2002) is an ontology editor that integrates numerous aspects of ontology engineering. Moreover, there are some other tools listed in (Duineveld, Stoter, Weiden, Kenepa & Benjamins 2000) that support ontology integration to some extent.

Besides those tools, some researchers, such as Grüninger et al. (Grüninger & Kopena 2003), Kalfoglou et al. (Kalfoglou & Schorlemmer 2003),

Gómez-Pérez et al. (Gómez-Pérez & and Richard Benjamins 1999), have investigated ontology integration from different technical aspects. Other related works, such as GLUE (Doan, Madhavan, Domingos & Halevy 2003) and Hovy (Hovy 1998) are more concerned with mapping techniques than the overall architecture.

Another closely related work is Process Specification Language (PSL) (<http://www.mel.nist.gov/psl/ontology.html>). Its aim is to create a process interchange language that is common to all manufacturing applications, generic enough to be decoupled from any given applications and robust enough to be able to represent the necessary process information for any given applications. It is more appropriate to refer to it as an ontology or a data model than a language.

Our work is inspired by the approaches in information integration and PSL as well. We argue that different techniques and methodologies are complementary and thus must be used in combination rather than exclusively. Bearing these in mind, we adopt a MAS perspective to model the variety of ontologies in ontology-based applications. We believe that the agent technology is ideally suited in a dynamic and distributed environment such as ontology integration on the Web.

An agent-based approach aims to provide a flexible and robust way to automate the process of ontology integration as much as possible to alleviate the heavy burden of building ontologies from scratch, which is believed to be error-prone and costly. Moreover, little prior knowledge is needed to start the system because agents, which are autonomous and engaged in flexible interactions, can perceive changes in the environment and adopt corresponding actions to achieve the goals in a timely fashion. Clearly, communications between agents play very important roles in agent interactions, which are based on some kinds of ontologies.

6 Conclusions and Future Work

Ontologies are becoming more and more important in the context of the emerging Semantic Web. Because of this trend, the need for integration and reuse of ontologies increases as well. In this paper, we have presented an novel agent-based framework to achieve ontology integration in the environment of similar ontologies existing in a distributed and heterogeneous way. With the presence of *ontology agents*, newly generated ontologies can be reused, which is in accordance with the intuition of reuse of existing ontologies no matter whether they are original or newly built.

Although the approach proposed in this paper is promising for ontology integration, some issues need to be further addressed. We attempt to use a process algebra to support agent interactions at a high level of abstraction. We hope that this perspective will allow the proposed framework to deal with ontology integration in an abstract but more flexible way.

Acknowledgements

Work reported here is partly supported by Swinburne VC's Strategic Research Initiative Grant 2002-2004 for project "Internet-based e-business ventures". The authors are grateful for Shane Grund's prototyping work.

References

- Baader, F., and Sattler, U., (2001), An overview of tableau algorithms for description logics, *Studia Logica*, **69**, 5-40.
- Calvanese, D., Giacomo, D. G., and Lenzerini, M., A framework for ontology integration, (2002), *Proc. of the 1st Semantic Web Working Symposium at the Emerging Semantic Web*, pp. 201-214.
- Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A., (2003), Learning to match ontologies on the Semantic Web, *VLDB Journal, Special Issue on the Semantic Web*, **12** (4), 303-319.
- Duineveld, J. A., Stoter, R., Weiden, R. M., Kenepa, B., and Benjamins, R. V., (2000), Wonder-Tools? A comparative study of ontological engineering tools, *International Journal of Human-Computer Studies*, **52**(6), 1111-1133.
- Gómez-Pérez, A., and Richard Benjamins, V., (1999), Applications of ontologies and problem-solving methods, *AI Magazine*, **20**(1), 119-122.
- Gruber, T. R., (1993), Toward principles for the design of ontologies used for knowledge sharing, KSL-93-04, Knowledge Systems Laboratory, Stanford University, <http://ksl-web.stanford.edu/>.
- Grüninger, M., and Kopena, B. J., (2003), Semantic integration, position statement, In: *Proc. of the Workshop on Semantic Integration*, jointly held with the *2nd International Semantic Web Conference*, Sanibal Island, Florida, USA.
- Heflin, J., (2001), Towards the Semantic Web: knowledge representation in a dynamic distributed environment, Ph.D. Thesis, University of Maryland, College Park.
- Hovy, E., (1998), Combining and standardising large-scale, practical ontologies for machine translation and other uses, In: *Proc. of the 1st International Conference on Language Resources and Evaluation (LREC)*, pp. 535-542. Granada, Spain,
- Jennings, N., and Wooldridge, M., (2001), Agent-oriented software engineering, J. Bradshaw (Eds.), *Handbook of Agent Technology*, AAAI/MIT Press.
- Kalfoglou, Y., and Schorlemmer, M., (2003), Ontology mapping: the state of the art, *Knowledge Engineering Review*, **18**(1), 1-31.
- Kalfoglou, Y., and Schorlemmer, M., (2003), IF-map: an ontology-mapping method based on information-flow theory, *Journal of Data Semantics*, **1**(1), 98-127.
- Klein, M., and Noy, F. N., (2003), A component-based framework for ontology evolution, In: *Proc. of the IJCAI'03 Workshop: Ontologies and Distributed Systems*, Acapulco, Mexico.
- Li, L., Yang, Y., and Wu, B., (2005a), Agent-based ontology mapping towards ontology interoperability, In: *Proc. of the 18th Australian Joint Conference on Artificial Intelligence (AI'05)*, LNAI 3809, Springer-Verlag, pp. 843-846, Sydney, Australia.

- Li, L., Yang, Y., and Wu, B., (2005b), Implementation of agent-based ontology mapping and integration, Technical Report, Swinburne University, <http://www.it.swin.edu.au/personal/yayang/papers/2005TR-Li-1.pdf>.
- McGuinness, D., Fikes, R., Rice, J., and Wilder, S., (2003), An environment for merging and testing large ontologies, In: *Proc. of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pp. 483-493, Breckenridge, Colorado, USA.
- Noy, F. N., (2003), What do we need for ontology integration on the semantic web, position statement, In: *Proc. of the Workshop on Semantic Integration*, jointed held with the *2nd International Semantic Web Conference*, Sanibal Island, Florida, USA.
- Noy, F. N., and Musen, M. A., (2003), The PROMPT suite: interactive tools for ontology merging and mapping, *International Journal of Human-Computer Studies*, **59**(6), 983-1024.
- Noy, F. N., (2004), Semantic integration: a survey of ontology-based approaches, *SIGMOD Record, Special Issue on Semantic Integration*, **33** (4), 65-70.
- Rahm, E., and Bernstein, P., (2004), A survey of approaches to automatic schema matching, *The VLDB Journal*, **10**, 334-350.
- Pinto, H. S., and Martins, P. J., (2001), A methodology for ontology integration, In: *Proc. of the International Conference on Knowledge Capture*, pp. 131-138, Victoria, British Columbia, Canada.
- Pinto, H. S., and Martins, P. J., (2004), Ontologies: How can they be built?, *Knowledge and Information Systems*, **6** (4), 441-464.
- Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D., (2002), OntoEdit: collaborative ontology engineering for the Semantic Web, In: *Proc. of the International Semantic Web Conference (ISWC 2002)*, LNCS 2342, Springer-Verlag, pp. 221-235, Sardinia, Italy.
- Uschold, M., Healy, M., Williamson, K., Clark, P., and Woods, S., (1998), Ontology reuse and application, In: *Proc. of Formal Ontology in Information Systems (FOIS'98)*, Treno, Italy.
- Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S., (2001), Ontology-based integration of information - A survey of existing approaches, In: *Proc. of the IJCAI'01 Workshop: Ontologies and Information Sharing*, Seattle, Washington, USA.