Comparing Ontologies Using Multi-agent System and Knowledge Base

Anne Håkansson[1], Ronald Hartung[2], Esmiralda Moradian[1], and Dan Wu[1]

[1] Department of Communication Systems
Royal Institute of Technology, KTH
Electrum 418, SE-164 40 Kista, Sweden
annehak@kth.se, esmirald@dsv.su.se, danwu@dsv.su.se
[2] Department of Computer Science
Franklin University
Columbus, Ohio, USA
Hartung@franklin.edu

Abstract. This paper presents an approach for handling several ontologies in a domain by integrating a knowledge base in a multi-agent system. For some online facilities, like e-business, several ontologies in different languages are needed to match the users' request. Finding the ontologies is one of the tasks, comparing and combining these ontologies is the other. The accomplishment of these tasks depends on the content in the ontologies like tags and structure but, in some cases, also language and matching techniques. Matching the contents is difficult due to differences between ontologies, often resulting from the lack of explicit and exact standards and development guidelines. This complication increases with the ontologies diverged languages. These problems are tackled by applying a multi-agent system wherein the agents, i.e., software agents and meta-agents, use the users' request to search for ontologies and the knowledge base to compare and combine the contents of the ontologies to create an overall solution. The software agents search for ontologies; the meta-agents keep track of the software agents, ontologies and the knowledge base that reasons with the contents of the ontologies.

Keyword: Multi-agent systems, Software Agents, Meta-agents, Ontologies, Knowledge bases, ´Reasoning.

1 Introduction

Many ontologies are developed for e-business, supporting e-commerce and e-services [3; 5]. These ontologies assist in online commerce and services, like supporting customers' purchasing products, travel trips and making restaurant reservations, online, as well as, facilitating companies buying products for their own manufacturing. The benefit is that these facilities give competitive advantages to the enterprises by supplying additional and extra-ordinary services and offers. The facilities can also help the enterprise to expand the number of customers, nationally and internationally.

Ontologies can be seen as templates for collecting information from individuals in order to complete tasks. In addition to collect data, the ontologies apply structures and build relationships between concepts. The notion of ontologies is an explicit specification, which can be used to model a domain with objects, concepts, properties and relations [3]. However, the ontologies lack of explicit and exact standards for designing, structuring and implementing their contents make them difficult to compare. A comparison facility can make a combination of ontologies applicable for e-business, especially, if the combination and interaction of the ontologies benefit the customers in purchases and, thereby, benefit selling companies. Additionally, a system can produce significant business value by providing opportunities to the companies or travellers, which can be new or additional information about offers that otherwise might be unknown. For example, companies can order sets of parts from the same enterprise and bargain for quality, price and delivery dates; an e-tourist can make reservations for flights, hotels, trips within the destination country, and events that take place at the time for travelling.

To support e-business customer, with commerce and services providing additional and targeted information, we propose a multi-agent system (MAS) that utilises both ontologies and knowledge bases to compare and combine the ontologies. The MAS uses software agents to search for ontologies that match the user's request and meta-agents to handle the contents of the different ontologies. From the user's request and the ontologies, the meta-agents compare similar parts with knowledge from the knowledge bases. With this knowledge, the meta-agents can support reasoning with the ontologies and combine several different ontologies.

2 Related Work

Multi-agent systems have been used for e-business systems. These systems include agent approaches to e-business with negotiation and user preference [13; 7; 15]. The applications range from buying and selling products, including information products, to optimizing traveller's arrangements, where agents have all kinds of roles [16, 2, 1].

An earlier attempt using MAS for business applications in multilingual ontologies [11] is a system that receives user requests and searches for the ontologies in the language of the destination country [9]. The MAS locates the ontologies that correspond to the request and returns the links and paths to the ontologies. The system upholds communication with the users by keeping track of the information that needs to be supplied. Moreover, the system performs the mapping between ontologies and language translation and can be used to reason with the content [6, 9]. The research in this paper is an extension of this earlier work. It extends the MAS with a better tool to reason with the ontologies, i.e., a knowledge base and interpreter. By extension, the agents cannot only reason with the contents of each ontology and make sure that they correspond to the users' request, the knowledge base can also support comparing the different ontologies and support filling in missing information into web sites, linked from the ontologies.

Another similar work is ontology-based context modeling and reasoning using OWL [18]. The ontology is in the context of knowledge management, where the ontology is referred to as a shared understanding of some domains. The reasoning is a logic-based context-reasoning scheme that reasons with the environment based on the content of the ontology [18]. The ontology is used to reason with the context, whereas we use a knowledge base as a knowledge source when reasoning with the ontologies, which has the context association for the task. The context is not derived from the ontologies – instead the agents' tasks handle the context and matching.

Knowledge is generally used to guide the agent's autonomous local decision-making processes, supporting agents' problem solving by providing expertise or search in the agents' database [12]. This is an interesting approach, but we use the knowledge and an interpreter to reason about the contents of the ontologies and support the users supplying information. Our approach guarantees that the meta-agents and interpreter use the same knowledge. The meta-agents are autonomous and perform simple matching by using the knowledge base to compare the contents of the ontologies; the interpreter perform advanced matching by interpret the knowledge in the knowledge based against the contents in the ontologies.

3 The Ontologies

Ontologies can form a formal and explicit specification [3] of a set of concepts and relationships in a domain. This specification is critical to the systems providing automatic search of the ontologies and reasoning with their contents. However, often the specifications differ, since there are no agreements upon standards for designing, structuring and implementing the ontologies. Hence, most of this work i.e., search and reasoning, is performed manually by users providing intelligence to accomplish the tasks [4], but the amount work requires a tool to perform the work, automatically.

The ontologies can be seen as a knowledge representation of parts of the world, captured entities, ideas, and events, with their properties and relations. They are modelled to gain understanding of existing things and their relationships in a language close to natural language [10]. Among different usages, the ontologies are created for business processes to construct models of services. Often the business process definitions and related artefacts (e.g. business documents, business objects) are represented as knowledge based on ontologies [10].

When using ontology in business processes, there are two aspects. First is the classic view, where the ontology describes the object of commerce, or the objects of the process space. The objects of commerce are the things bought or sold, plus properties of those objects. We can extend the properties to include aspects like warranties, conditions of sale, payment terms, as well as, physical properties of the objects like color, size, and functionality. The second is the business process, which is a sequence of steps and may include conditionals, branching (alternatives) and loops. The steps must be concrete and well defined. The advantage of ontology, or other, formal representation is that a reasoning system can enact the process.

A more complex issue is being able to understand the process of a correspondent system, that is an external system to interact with for a business purpose. Here, the ontology, acting as a description of the process, lends itself to automating the transactions and even applying negotiation to the business process. These business processes comprise a known set of various purchase processes and other processes to support searching and presenting options for purchase services that are expressed in ontologies

[9]. If only one company is used for the e-business task, only a single ontology might be needed for handling the task, like combined facilities offered by travel agencies. However, if several ontologies are needed in conjunction, this becomes complex. A tool for handling several ontologies, automatically, can simplify the process of purchasing services from several distinct companies.

4 The Multi-agent System

The multi-agent system is developed to support users finding acceptable solutions to tasks in e-business. The tasks are sometimes complex and may require several ontologies to accomplish the assignments. These ontologies can be used as a collected set that enables the handling of a complex endeavour rather than one task at the time.

Working with several different external ontologies is difficult. These ontologies can have different structures, use a variety of names on the tags and require a variety of data and information since the design is due to the developers' structure choices and application of languages' standards. These differences must be taken care.

The purpose of the system is to analyse the content and combine the ontologies to create more complete solutions, as well as, support filling-in the missing information. Thus, the system must check the ontologies to find the web sites in which parts are to be filled in and communicate with the users to collect the information.

The MAS has connection with the web. The ontologies are fetched from the web, but the system compares the contents of the ontologies locally, by working with copies of the ontologies. The reason is that the web is expanding rapidly, which requires locally handling services on the computer rather than on the web. Nonetheless, the MAS needs to communicate with the original ontologies since these ontologies contain links to the web sites at which the missing information needs to be filled-in.

The system is a MAS with a knowledge base. The system is not purely an agent-based system since it uses a knowledge base to reason with the known knowledge components of the ontologies and also incorporate new parts. Neither is the system a knowledge-based system since the agents deal with the contents of the knowledge base, not the reasoning strategies. In the system, the reasoning strategies are not steering the process of providing the system with parts that it needs to draw conclusions. Neither are the users are providing the system with facts to operate on, the agents are. This works by splitting the matching and combining functions between the meta-agents and the knowledge base. For simple matching cases, the meta-agents can handle the match. The rules of a knowledge base provide a more powerful mechanism for matching and combining ontologies. The other significant feature of the rules is the ability to modify and add rule-based match reasoning. The modification and adding are performed as the system is being refined.

The multi-agent system contains software agents and meta-agents, a container with copies of the ontologies, a knowledge base with knowledge about the content of the ontologies and an interpreter that interpret the contents for architecture see Figure 1.

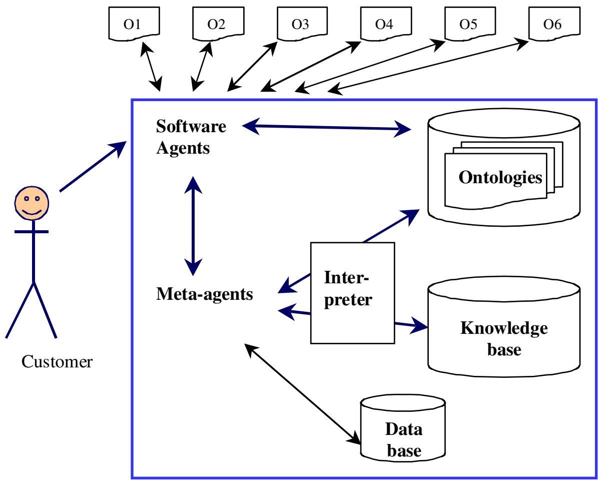

Fig. 1. The architecture of the multi-agent system

Briefly, the system procedure is: the customer launches the system with a user request. The software agents receive the request and look for the ontologies that match the request. The found ontologies, presented as O1-O6 up in the figure, are copied to an "ontologies-container", towards which the meta-agents work. The meta-agents use the knowledge base to compare the ontologies and the interpreter to interpret the knowledge in the knowledge based against the contents in the ontologies. Thus, the meta-agents make the simple matching by comparing different ontologies against the knowledge base. If there is a direct match the meta-agents can continue to work with the next part of the ontologies. However, not all the information, tags or names are represented in the knowledge base and, therefore, some interpretations of the rules are needed. The interpreter matches the content by inferring conclusions that can match the ontologies. If not, additional facts have to be provided by the users. The results of the interpretation and the user-given information are temporary stored in a database.

4.1 The Agents in the Multi-agent System

The multi-agent system uses software agents and meta-agents, to search for the ontologies and compare and combine the contents of the ontologies. The software agents are like search agents finding the ontologies that match the user requests. The meta-agents are managers that control the software agents and handle the ontologies. Although we use software agents, it is possible to use intelligent agents that learn from the environment. However the web is a changing environment and, for the current work in this paper, it is better to search for new information, every time the user poses a new request.

The system is launched by a user request. From this request, the software agents start searching for ontologies, independently of each other. The search is performed without any authorised control or user involvement, which is a benefit with agents [14]. Thus, the agents must be capable of autonomous actions, situated in different environments [17] where the agents independently search for the ontologies and return with locations for each ontology (the URL of the ontologies).

The MAS works internally on host computer and externally on the web, relieving the web by reducing computation. The internal part includes most of the MAS, handling the agents, communicating with the web, comparing the content, whereas searching is made externally. The combination of internal and external parts makes the agents' working environment interesting. The internal environment is fully observable, deterministic, episodic, discrete and static whereas the external environment is partially observable, stochastic, episodic, continuous, and dynamic.

In the internal system, the meta-agents can observe the environment and they know the next event, and are deterministic. Moreover, the task is divided into atoms, performed one atom at the time, i.e., episodic, and each agent has a finite number of states and actions it can take, i.e., discrete, in a static environment.

In the external environment, the agents, the software agents and the meta-agents, only partly know the environment. These agents know the closest nodes but not the nodes, at a distance. Although, the agents can learn the environment, it changes often enough to stay partly observable. The agent cannot predict the behavior of the environment since the states, which will result from performing an action, are unclear.

In the environment there will be stochastic elements, i.e., web sites and ontologies that randomly appear. The software agents task is searching ontologies but the contents and locations of the ontologies can vary with time as well as, finding ontologies, depending on when actions these are performed. Hence, it difficult to reuse all of results of early searches since the environment is highly unpredictable.

The tasks are episodic and the software agents work with one task at the time. Hence, the software agent has one assignment, which is following a connection to its node. If this node has several connections, there will be equally many software agents. When the agents reach a destination, node, which is not an end node, the agents will launch other agents [8], one for each new connection. To keep track of the software agents, we use meta-agents. These meta-agents are built on the software agents and can be used to influence the further work of the agents.

The environment is dynamic and changes on a daily bases. Even though the connections are not changed, the content of the web sites and ontologies can change. Moreover, the user requests change either with refinement of old requests or a totally new request. Hence, for each new request, the software agents need to restart the search. Only the closest nodes are reused and, further away from the starting point, new unexplored parts, most certainly, will be exercised.

The meta-agents become a chain of software agents, where the first agent is the first one launched from the user request and the last agent is the one that reached a web site, successfully. The chain is a path that successfully leads to destination of an ontology that corresponds to user request. These ontologies are tagged for further comparisons and combinations.

One problem with software agents is the continuous work on the web. There might be uncountable number of states and actions, which arise from the continuous time problem. The solution we use is execution suspension that controls the agents' performance to prevent them to continue the search, endlessly.

When the agents finish the search, the agents bring back a number of ontologies to the MAS. The meta-agents start analyzing these ontologies to compare their contents.

The comparing of the ontologies depends on the content in the ontologies where the structure is one of the major obstacles and the language of the ontologies is another. The first step is to use a matching technique that compares the ontologies with the user's request, which is more sophisticated than the software agents' search. The second step is to establish the equivalence between the ontologies.

The meta-agents need different functions: search meta-agent that contains the paths to the original site of the ontology and reasoning meta-agent that supports comparing the ontologies to the user's request, as well as, to the other ontologies and the knowledge in the knowledge base. These meta-agents are then used to combine the ontologies to give total solution of a request and support filling-in data in the ontologies. There are many parts to keep track of: the user request, the ontologies, the parts that have been analysed and are successfully compared to the knowledge base, the parts that are not found in the knowledge base but in the other ontology, and parts in the ontology that are found in the knowledge base, but not in the other ontology.

The knowledge can be stored centrally as a central unit or locally in the agents [12]. If the knowledge is central, it can be easily accessed be all the parts in the system. If the knowledge is local, the different agents need to be contacted to retrieve the knowledge. Since the accessibility is important, as well as, support comparison in parallel and frequently updating the knowledge, a central part is more effective than have agents carrying around knowledge, hence a separate knowledge base.

4.2 The Knowledge Base in the Multi-agent System

When the MAS performs complex tasks, it can benefit from a supporting tool, like a knowledge base. The knowledge base can provide necessary knowledge and guidance to the MAS, to perform more efficiently.

The agents search for the ontology by a search string in, e.g., the URL, annotations, definitions, and classes or properties. When the user requests ontologies, the request is transformed into a term, which is then specified in the search string, for example, if user request includes "travel", the term can be *url:travel* or *def:travel*. These are terms are implemented in the knowledge base, as well as, other well-known terms.

The knowledge base includes parts that can be found in the ontologies. Some commonly used fields, which is also used to constraint user requests, are basic metadata, document metadata and RDF metadata. The basic metadata includes *url:, desc:*, *def:*, *ref:*, *pop:*, and *ns:*, The document metadata consists of *hasEncoding:*, *hasLenght:*, *hasMd5sum:*, *hasFiletype:*, *hasDateLastmodified:*, and *hasDataCache*. The RDF metadata has *hasGrammar:*, *hasCntTriple:*, *hasOntoRatio:*, *hasCntSwtDef:* and *hasCntInstance*. For the description term, there are other terms associated to it. For example, *log:forSome*, *title*, *s:label*, and *s:subClassOf*. To the term "Class", there are terms like *s:comment*, *s:label*, *ont:UniqueProperty*, *s:domain*, and *s:range*. All these terms have to be represented as knowledge to be able to compare the content of the ontologies.

In the knowledge base, there are facts and different levels of complex rules. Some terms can be matched directly to the facts whereas other terms must use rules to reach a conclusion that matches the contents of the ontologies. Thus, these are matching terms in the rules of the knowledge base. The rules are either compound terms or can have different names leading to the same conclusion about the term. These rules are used to lead the same term in the ontologies to be compared.

When a term in the ontology is not directly represented in the knowledge base, it can be represented as a rule with several different alternatives. An example of several alternatives is synonyms, which can be represented as rules in the knowledge base.

The terms that are not covered by the knowledge base are stored temporarily. These are stored while finding out if they are significant and if there are any correspondences to other ontologies. If so, the new term can be permanent in the knowledge base, either as facts or as rules. It becomes a fact if it is direct correspondence; a rule if the other rules are used to reach conclusion about the term. Letting the knowledge base upgrade itself can be dangerous if it learns wrong terms but important for further communication. Therefore, expanding the knowledge base should be made under supervision of knowledgeable users.

4.3 The Interpretation

The MAS can either be ruled by the knowledge, or use the knowledge to perform tasks. Then, the interpretation mechanism can either be the engine that steers the system and its actions or the system can use the interpretation for partial solutions and use the results for further operation. We use the latter.

To reason with ontologies, and find correspondences between the ontologies, the system must inspect the ontologies and compare their contents to the knowledge base and to other ontologies. This is performed by the meta-agents and the interpretation.

The knowledge is accessible whenever the agents need it. By the knowledge in the knowledge base, it is possible to derive the different tags and match those to knowledge about the tags and text. If common ontology languages are used, tags are commonalities in the ontologies. This is not always the case and to get more accurate information about the tag equality, text is used. Even though the tags are not the target for the comparing the ontologies, they still need to be compared because they give additional information that can be decisive importance for the of the text comparison.

The interpreter is an inductive interpreter that works through the ontology with the knowledge from the knowledge base. It checks if the tags are found in the knowledge base and if these tags need to be explored further. The exploration occurs if the ontology links to web sites. All web sites must be examined to find out if information is missing. If the tags that are not present in the knowledge base is temporary stored in a database. These are used for the comparison of two ontologies.

All end tags and comment tags are left out since the knowledge base is not a compiler or an automatic documentation tool. Moreover, these tags do not add any additional information to the interpreter or the users. Nonetheless, if they would be significant, it is easy to include them in the knowledge base.

Then the text, associated with the tags is checked. Commonly the text is a string that starts with # , http:// or a word like Resource. These are important when it comes to comparing ontologies.

For the comparison, a second ontology is worked through. For each tag in the first ontology, the whole second ontology is checked until the tag is found. The tags are compared by interpret them with the knowledge in the knowledge base. This because it might not be a direct matching between the tags as when comparing the words presented as facts in the knowledge base. For example, the word professor can be the same as senior researcher, and senior lecture the same as teacher.

It can also be by using rules, where an *#Activity* can be *#BunjeeJumping*, *#Safari*, *#Sunbathing*, or *#Yoga*. This is then presented in a rule. To illustrate the comparison, we present real examples of the content in ontologies found on the web, in Figure 2.

```
<rdf:RDF
xml:base="http://www.mindswap.org/2004/multipleOnt/F
actoredOntologies/factoredTravel/FactoredActivity.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:about="#Adventure">
<rdfs:subClassOf>
<owl:Class rdf:about="#Activity"/>
</rdfs:subClassOf>
</owl:Class>
…
<owl:Class rdf:about="#Adventure">
<owl:disjointWith>
<owl:Class rdf:about="#Sightseeing">
  </owl:Class>
</owl:disjointWith>
</owl:Class>
…
<owl:ForeignClass
rdf:about="http://www.mindswap.org/2004/multipleOnt/F
actoredOntologies/factoredTravel/FactoredContact.owl#C
ontact">
<owl:foreignOntology
rdf:resource="http://www.mindswap.org/2004/multipleOn
t/FactoredOntologies/factoredTravel/FactoredContact.owl
"/>
</owl:ForeignClass>
<owl:LinkProperty rdf:about="#hasContact">
<owl:foreignOntology
rdf:resource="http://www.mindswap.org/2004/multipleOn
t/FactoredOntologies/factoredTravel/FactoredContact.owl
"/>
```

```
<rdf:RDF>
<owl:Ontology rdf:about="">
<rdfs:comment>Travel Itinerary</rdfs:comment>
<owl:versionInfo>
$Id: itinerary-ont.n3,v 1.4 2003/10/03 20:05:44 mdean
Exp $
</owl:versionInfo>
</owl:Ontology>
<Aircraft
rdf:about="http://www.daml.org/2001/06/itinerary/itin
erary-ont#A300">
   </Aircraft>
…
<owl:Class
rdf:about="http://www.daml.org/2001/06/itinerary/itin
erary-ont#Aircraft">
<owl:oneOf rdf:parseType="Resource">
<rdf:first
rdf:resource="http://www.daml.org/2001/06/itinerary/i
tinerary-ont#First"/>
<rdf:rest rdf:parseType="Resource">
…

<owl:oneOf rdf:parseType="Resource">
<rdf:first
rdf:resource="http://www.daml.org/2001/06/itinerary/i
tinerary-ont#First"/>
<rdf:rest rdf:parseType="Resource">
```

Fig. 2. Ontologies being compared and combined

In the figure, the first ontology is presented to the left and the second ontology to the right. The first is checked against the knowledge base, with tags, such as, *RDF*, *owl:Ontology*, *owl:Class*, *rdfs:subClassOf*, *owl:disjoint*, *owl:ForeignClass*. These are all found in the knowledge base. The interpreter turns to the second ontology and checks if the first tag in the first ontology also is present in the second, thus *<rdf:RDF>*, and then saves *xml:base=http://…*. Then the interpreter checks *owl:Ontology*, which is also present in the second ontology and moves to *owl:Class*. The first line in the second ontology is the comment, *versionInfo*, followed by end tags, which are all skipped until *owl:Class* is found. In the first ontology *#Adventure* is found and in the second a link to itinerary and aircraft. These are not compared but stored for further investigation later on. The adventure has some categories in the ontology that needs to be handled and a link to a web site, which is fetched and checked for parameters or requirements for user given information. When a statement is not known, we assume it is true until proven, hence, open world assumption.

For combining the ontologies, there must be parts that can be matched or related. Consider the ontologies in Figure 2, again, it is possible to combine those into a third ontology (O) or a document with ontological structure. The entities that are in the first ontology (A) and the second ontology (B) and matches the knowledge in the knowledge base, can be built into the third ontology, hence, A + B -> O. This means that every entity A1, of ontology A and entity of B1 of ontology B, is inserted in ontology O in such way that:

```
If the entity A1 (and B1) is found both A and B then insert
the entities in O and give the correspondence of equal,
subset partial or coverage.

If the entity A1 is found in either A or B, then insert the
entity A1 into O.

If the entity B1 is found in either A or B, then insert the
entity B1 into O.
```

The process proceeds until significant entities are found and matched and become a combined ontological structure with links to both ontologies A and B. As mentioned above, in some cases the users must be involved in this process.

5 Conclusion and Further Work

This paper presents an approach to handle ontologies using a multi-agent system with a knowledge base, which support online facilities, like e-tourism and e-business. The system searches for relevant ontologies, as well as, comparing and combining the ontologies by analyzing the structure of the ontologies. If good match is found, the multi-agent system can support filling in missing data in the ontologies.

In the multi-agent system, software agents and meta-agents are used to search for ontologies from the the users' request and work with the contents of the ontologies The meta-agents keep track of the software agents, ontologies and knowledge in the knowledge base and reason with contents of the ontologies.

The knowledge base is at a novel level handling RDF ontologies and, thus, needs to be expanded with more knowledge to handle many different kinds of ontologies with different structures. Also testing is needed to make sure that a high level of correspondences is met. A small number of discrepancies can become extremely costly. The combination is an interesting technique since, if tuned and thoroughly tested, it can be used to developed ontologies automatically. This can minimize problems with mistakes when using two or several different ontologies.

Acknowledgement

This work is supported by Sweden-Korea Research Cooperation Programme funded by STINT, The Swedish Foundation for International Cooperation in Research and Higher Education. http://www.stint.se/

References

1. Apelkrans, M., Håkansson, A.: Information coordination using Meta-Agents in Information Logistics Processes. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 788–798. Springer, Heidelberg (2008)
2. Bhavsari, V., Boley, H., Yang, L.: A Weighted-tree similarity algorithm for Multi-agent systems in E-Business environment (2004), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.8260&rep=rep1&type=pdf

3. De Bruijin, J., Fensel, D., Keller, U., Lara, R.: Using the Web Service Modeling Ontology to enable Semantic eBusiness. Commnications of the ACM 48(12) (2005)
4. Flett, A., Brown, M.: Enterprise-standard ontology environments for intelligent e-business (2001), `http://www.csd.abdn.ac.uk/~apreece/ebiweb/papers/flett.pdf`
5. Gordijn, J., Akkermans, H.: Designing and Evaluating E-Business Models. IEEE Intelligent Systems Archive 16(4), 11–17 (2001)
6. Hartung, R.L., Håkansson, A.: Using Meta-agents to Reason with Multiple Ontologies. In: Nguyen, N.T., Jo, G.-S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS (LNAI), vol. 4953, pp. 261–270. Springer, Heidelberg (2008)
7. Helal, S., Wang, M., Jagatheesan, A.: Service-centric brokering in dynamic e-business agent communities. JECR, Special Issue in Intelligent Agents in E-Commerce (2001)
8. Håkansson, A.: An Approach to Event-Driven Algorithm for Intelligent Agents in Multi-agent Systems. In: Nguyen, N.T., Jo, G.-S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS (LNAI), vol. 4953, pp. 411–420. Springer, Heidelberg (2008)
9. Håkansson, A., Hartung, R.L., Jung, J.J.: Using Multi-Agent System to handle information in Multilingual Ontologies. Accepted in Smart Innovation, Systems and Technologies 2010 (2010)
10. Jenz, D.: Business Process Ontologies: Speeding up Business Process Implementation (2003), `http://ip-68-178-224-58.ip.secureserver.net/publications/07-03%20WP%20BP%20Ontologies%20Jenz.pdf`
11. Jung, J., Håkansson, A., Hartung, R.L.: Indirect Alignment between Multilingual Ontologies: a Case Study of Korean and Swedish Ontologies. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS, vol. 5559, pp. 233–241. Springer, Heidelberg (2009)
12. Marik, V., Pechoucek, M., Stepankova, O.: Social Knowledge in Multi-agent Systems. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001 and EASSS 2001. LNCS (LNAI), vol. 2086, p. 211. Springer, Heidelberg (2001)
13. Purvis, M., Nowostawski, M., Oliveira, M., Cranefield, S.: Multi-agent interaction protocols of e-business. In: IAT IEEE/WIC, October 13-16, pp. 318–324 (2003)
14. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (2003), ISBN 0-13-790395-2
15. Shin, H., Bo-Yeon Shim, B.: e-Business Agent Oriented Component Based Development for Business Intelligence. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4252, pp. 803–811. Springer, Heidelberg (2006)
16. Wickramasinghe, L.K., Amarasiri, R., Alahakoon, L.D.: A Hybrid Intelligent Multiagent System for E-Business. Computational Intelligence 20(4), 603–623 (2004)
17. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons Ltd., Chichester (2002), ISBN 0-471-49691-X
18. Wang, X.H., Gu, T., Zhang, D.Q., Pung, H.K.: Ontology Based Context Modeling and Reasoning using OWL. In: IEEE International Conference on Pervasive Computing and Communication (PerCom 2004), Orlando Florida, March 14-17 (2004)