A Service-Oriented Architecture to Support Agent Reputation Models Interoperability

Luis G. Nardin[1], Anarosa A. F. Brandão[1], Jaime S. Sichman[1], and Laurent Vercouter[2]

[1] Laboratório de Técnicas Inteligentes - EP/USP
Av. Prof. Luciano Gualberto, 158 - trav. 3 - 05508-900 - São Paulo - SP - Brazil
{luis.nardin, anarosa.brandao, jaime.sichman}@poli.usp.br
[2] Ecole Nationale Supérieure des Mines de Saint-Etienne
158, cours Fauriel, 42023 Saint-Etienne Cedex 2, France
Laurent.Vercouter@emse.fr

Abstract. Agents are becoming a popular technology for the development of distributed, heterogeneous and always available systems. In those systems, interactions are essential, but semantic heterogeneity turns the establishment of interaction among agents into a problem. When considering reputation models in multi-agent systems, the lack of a consensus about the reputation definition could be a problem for interactions since they are essential to accelerate the convergence of the reputation evaluation. We propose in this paper a service-oriented architecture to deal with this semantic heterogeneity. This architecture supports concept mapping and translation among reputation model ontologies to a common ontology and vice-versa, thus allowing heterogeneous agents interoperability.

1 Introduction

The growth of the internet and associated technologies brought ”interaction” as a new metaphor for computation, turning computing into an activity that is inherently social, leading to new ways of conceiving, designing, developing and managing computational systems [10]. The multi-agent approach is an example of a technology that complies with such viewpoint, since agents have two important capabilities in multi-agent systems (MAS): to act autonomously at some extent and to engage in social activities needing cooperation, coordination and negotiation [19]. The engagement in any of these activities implies that the agent will exchange information. In open environments, where there is no control about the agents that enter or leave the system, agents that participate on those social activities are exposed to risks, e.g. when taking a decision based on inaccurate information received from a malevolent agent.

Some solutions to this problem are based on trust models, which serve as a decision criterion for an agent to engage and to participate in social activities. The concept of reputation from social sciences has been used by most of the researchers as the preferred mechanism to implement computational trust models. Moreover, reputation is a social property or a social process. It is a social property when considered as an agent's mental representation about other agents and a social process when considered as the result of the belief's transmission [5].

Considering reputation as a social process requires that the agents in the system interact

in order to increase the amount of information transmission, thus accelerating the reputation evaluation convergence. However, as there is no consensus on a single reputation definition, the several reputation models already proposed ([4], [5], [11], [14], [15], [20]) were developed using different approaches and different semantics attached to reputation and its associated concepts. In order to overcome this semantic heterogeneity of reputation models, the creation of an ontology that could subsume the different reputation models, called Functional Ontology of Reputation (FORe) was proposed in [3]. The underlying idea in this work was to use a so-called hybrid approach [18]: reputation interoperability would be obtained by translating a source model (expressed in ontological terms) to FORe, and then by translating the result obtained in FORe to a target model (also expressed in ontological terms). In another work [17], a general agent architecture was proposed to implement agents that could interoperate about reputation using FORe as a common ontology.

In this paper, we propose to use a service-oriented architecture (SOA) to support agent reputation interoperability. The use of a service-oriented approach is related to the idea proposed by [7] of providing all the ontology-related functions through Web Services [2]. The paper is organized as follows. In section 2, we briefly present the general agent architecture proposed on [17], and on which we are basing our work. Section 3 presents an overview of the service-oriented architecture proposed to support agent reputation models interoperation. A detailed description of the architecture components (the Ontology Mapping Service and the Translator module) is given in sections 4 and 5. A scenario illustrating the architecture usage is shown in section 6 and a case study is presented in section 7. Finally, conclusions and comments about future work are presented in section 8.

2 A General Architecture for Reputation Interaction

A general architecture for reputation interaction was proposed in [17] (Fig. 1). In this architecture, it is considered that different agents may have heterogeneous reputation models and the agent reputation model is implemented in a module called Reputation Reasoner Module (RRM). A component called Reputation Mapping Module (RMM) was integrated to the agent architecture in order to provide the mapping and translation ontology functions [9] that would allow agents with different reputation models to interact.

Two ontologies are used: the FORe and the reputation model ontology, this latter consisting of the representation of the internal agent reputation model in ontological terms. Hence, when two agents need to exchange information about reputation, the RRM of the source agent activates the RMM to translate its query from its inner reputation model to FORe and then the Interaction Module (IM) sends the message to the target agent. When this latter receives the message, a similar process is done: the query is translated from FORe to the target reputation model by the RMM, which activates the RRM in order to process the query according to its inner reputation model.

Although this architecture gives a general idea of providing reputation interoperability, it does not represent, however, a full detailed architecture [17]. We propose in the sequence a possible solution, by using a SOA approach.

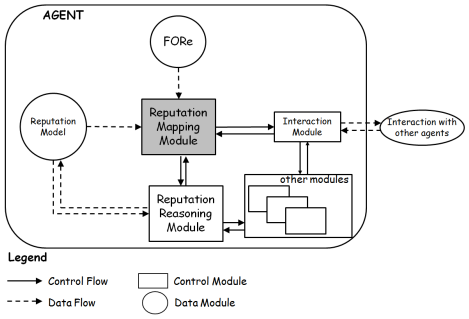

Fig. 1. Agent architecture for reputation interaction

3 The Service-Oriented Architecture for Reputation Interaction

The main underlying idea in the proposed service-oriented architecture for reputation interaction is that the mapping between different reputation models represented by ontologies may be realized off-line and be available on-line as a service. In this architecture, the hybrid approach proposed by Visser et al. [18] is used to allow the interoperation of agents with different reputation models based on a common vocabulary, which requires the agent to perform two distinct but interrelated ontology functions: mapping and translation. Mapping is a collection of functions assigning the concepts and relations in one ontology to the concepts and relations in another ontology. Translation is the application of the mapping functions to translate sentences from a source to a target ontology [9].

The advantage of using a service-oriented architecture, from a design/programming perspective, is that the agents become simpler since they do not need to perform the mapping function internally. While the advantage of using the hybrid approach comes from the fact that the agents do not know the other agents internal reputation model avoiding cheating.

Hence, the architecture extends the previous general agent architecture in two ways: (i) it subdivides the RMM in two distinct and specialized modules: the Ontology Mapping Service (OMS), which performs the ontology mapping function, and the Translator, which performs the ontology translation function. (ii) It performs the ontology mapping function as a service outside the agent architecture. Fig. 2 shows this extended architecture. By defining such extension, we intend to alleviate the agent dynamic workload, since it will not need to perform the mapping function on-line. Moreover, the results of such mapping will be stored in the service and it may be reused by new agents that enter the system and have an internal reputation model that was already mapped and stored in the service.

The **Ontology Mapping Service** module is a service and resides outside the agent. It implements the ontology mapping function and it has two main functionalities: (i) to map concepts from the target's reputation model ontology to the concepts of the common ontology. This mapping can be directly inferred by simply classifying the resulting ontology from the integration and alignment of a given reputation model and the com-

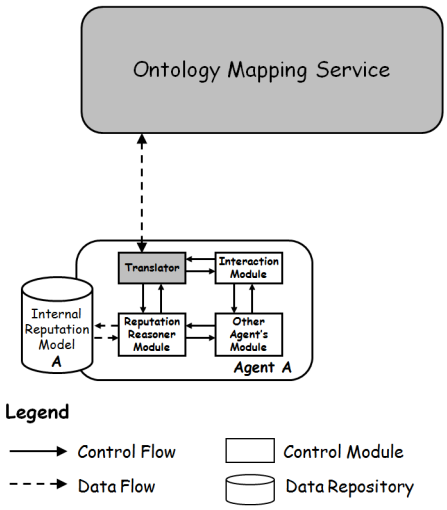

Fig. 2. The proposed service-oriented architecture

mon ontology [12], and (ii) to answer concept translation requests from the **Translator** module.

The **Translator** module resides inside the agent and it implements the ontology translation function. It has four main activities: (i) to translate the reputation messages from the common ontology to the target agent's reputation model ontology whenever the message comes from the IM; (ii) to translate the reputation messages from the agent's reputation model ontology to the common ontology whenever the message is sent to IM; (iii) to trigger some function in the RRM based on the interpretation of messages written using the reputation model ontology; and (iv) to create a message using the reputation model ontology whenever requested by RRM. The RRM is dependent of the agent internal reputation model and its implementation is out of the scope of this paper.

4 The Ontology Mapping Service

The Ontology Mapping Service (Fig. 3) is considered as the architecture's core component since it performs part of the ontology mappings and provides it through its Web Services interface for use by the Translator module. Its existence is independent of the agent since it is provided as a service.

A description of its components follows:

- *Ontology Repository* stores the reputation models ontology described in terms of a common ontology.
- *Translation Repository* stores the mapping of reputation models ontology concepts to the common ontology concepts.

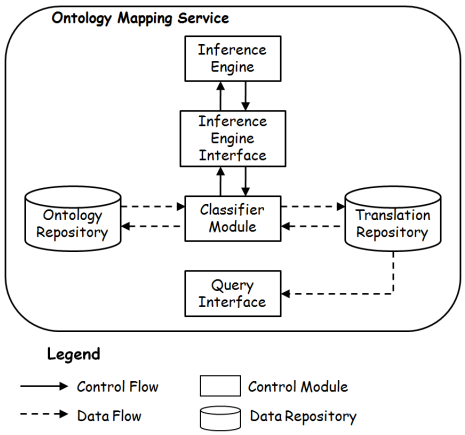

Fig. 3. The Ontology Mapping Service components

– *Classifier Module* reads the ontologies stored in the Ontology Repository; classifies it using the Inference Engine and stores the result in the Translation Repository. Its pseudo-code follows:

```
read ontology
classifyOntology(ontology)
allAsserted = getAllAssertedConcepts(ontology)
for each asserted of allAsserted do
  allInferred = getAllInferedConcepts(asserted)
  for each inferred of allInferred do
    write TranslationRepository[asserted,
                                inferred]
  end for
end for
```

Algorithm 1. Classifier Module algorithm

– *Inference Engine Interface* is a communication module between the Classifier Module and the Inference Engine.
– *Inference Engine* is an ontology reasoner that infers the relations between the reputation model ontology and the common ontology concepts.
– *Query Interface* answers the Translator module requests for translation of concepts.

The Ontology Mapping Service is implemented as a Web Service [2] and it is fully implemented in Java programming language using the Protégé-OWL Plugin [8]. The inference engine chosen was Pellet [16] since it (1) is completely developed in Java and (2) has a method call integration to the Protégé-OWL Plugin.

5 The Translator Module

The Translator module (Fig. 4) utilizes the mappings stored in the OMS in order to translate sentences from the common ontology to the agent's reputation model ontology and vice-versa. This module is tightly integrated to the other agent's modules, since

it resides inside the agent. The Translator module interacts with two agent's modules: the IM and the RRM. Among the functionalities performed by this module, there is the treatment of translation inconsistencies.

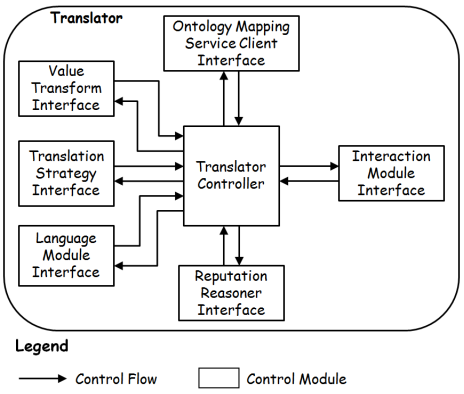

Fig. 4. The Translator Module components

Translation inconsistency is considered any translation of a source ontology concept to a target ontology concept which mapping is not one to one, for instance, when the source ontology concept maps to more than one target ontology concept or when the source ontology concept is not mapped to any target ontology concept.
Flexibility and configurability were identified as the two main required characteristics for this module design, since not all the agents implement the same IM and RRM, and each agent may have different translation strategies. Flexibility allows the Translator module to be adapted to interact with different kinds of IM and RRM, while configurability allows the selection of specific translation strategies and value transformations to fulfill the agent needs.
In order to satisfy the flexibility characteristic, the Translator module is designed with one central component, the Translator Controller (TC) and six other components that perform specific functions. All the six components are actually interface specifications, which have predetermined function in the architecture according to the following:

- *Interaction Module Interface* (IMI) receives/sends messages from/to the agent's IM.
- *Ontology Mapping Service Client Interface* (OMSCI) queries the OMS in order to obtain all possible translations from the source ontology to the target ontology.
- *Value Transform Interface* (VTI) is executed to transform, if necessary, the values associated with a concept when that concept is translated from an ontology to another.
- *Translation Strategy Interface* (TSI) determines the strategy to use when the concept does not have a possible translation or when there are multiple possible translations. In the former, some of the possible inconsistency treatments are: (i) to remove

the concept from the translated message, (ii) do not translate the concept and keep the original one, or (iii) to use some kind of heuristic to determine a non-direct translation. In the latter, the possible strategies are similar: (i) to remove the concept from the translated message, (ii) to translate the source concept to the multiple target concepts, or (iii) to use some kind of heuristics to choose one of the possible choices.

Despite the possibility of creating strategies to prevent translation inconsistencies, they may not solve the inconsistencies for all possible situations and they may generate other inconsistencies. Some examples are: (i) the strategy of removing the concept from the translated message may invalidate the message content causing some misunderstanding between the agents; (ii) the strategy of keeping the original concept when there is no possible translation may cause the impossibility of the message interpretation by the target agent since it does not know the meaning of the non-translated concept.

In order to prevent new inconsistencies when using heuristic strategies, those strategies must be logically equivalent (A $\Leftrightarrow$ B), meaning that if concept A translates to concept B, when the inverse translation is requested the concept B must translate to concept A.

- *Language Module Interface* (LMI) performs the parsing of the internal language used in the message to identify concepts for translation and to rebuild the message after the concepts translation.
- *Reputation Reasoner Interface* (RRI) interprets the messages described using the Translator module internal language and, based on that, performs calls to the RRM. It receives calls from the RRM and builds messages in the internal language.

The configurability characteristic is satisfied by the possibility of configuring the TC to use the desired implementation of each one of the components specified as an interface.

6 Usage Example

This section presents a scenario showing the level of interoperability achieved as a result of the agents' interaction about reputation when using the architecture previously described. Considering the use of the service-oriented architecture proposed, at least one OMS must exist in the system to support the mapping and translation of reputation model ontology concepts to a common reputation ontology concepts and vice-versa. The following two steps are required prior to use OMS [12]:

1. to design the reputation model ontologies of the reputation models, if the reputation models are not described in ontological terms, since the OMS only maps ontologies.
2. to align the reputation models ontology to the common reputation ontology, since the OMS processes only ontologies that are already described in terms of a common reputation ontology.

Designing the Reputation Model Ontologies
We built the reputation model ontologies manually by using Protégé as the editor and

OWL [1] as the ontology language, which is the most recent standard ontology language from the World Wide Web Consortium (W3C[1]). In the sequence, the terminologies identified as concepts in two reputation models, L.I.A.R. [11] and Repage [14], are described. Those concepts compose the design of the reputation model ontologies.

L.I.A.R. (Liar Identification for Agent Reputation) is a model for the implementation of social control of agent interaction. The idea is to provide tools that allow agents to (1) reason about other agent's interaction; (2) detect any interaction rules violation; and (3) maintain a reputation model of other agents. Its reputation model distinguishes reputation in five different types, which are based on seven roles involved in the reputation-related processes. Each of the seven roles is defined by the source and kind of information used to calculate the reputation value. The seven roles are: *Target* role, *Participant* role, *Observer* role, *Evaluator* role, *Punisher* role, *Beneficiary* role and *Propagator* role. The five different types of reputation are: *Direct Interaction-based Reputation* (DIbRp), *Indirect Interaction-based Reputation* (IIbRp), *Observation Recommendation-based Reputation* (ObsRcbRp), *Evaluation Recommendation-base Reputation* (EvRcbRp) and *Reputation Recommendation-based Reputation* (RpRcbRp). Each reputation is associated to a facet, which is the subject the evaluation is about.

Repage. The Repage (Reputation and ImAGE) system is a computational module based on a reputation model proposed by [5]. *Image* and *reputation* are the two main concepts in this model and they represent social evaluations. *Image* is an evaluative belief, which is formed using information, acquired by agent experience or propagated third-party images. *Reputation* is a meta-belief, which is formed, based on anonymous reputation value transmitted on the social network about the target agent. The social evaluations are context-based which means that the agent may hold different social evaluation for the same target (*AgentImage* and *AgentReputation*). The model distinguishes the types of agents involved in the image or reputation formation in four different types: *Target* agents, *Evaluator* agents, *Propagator* agents and *Beneficiary* agents.

Aligning the Reputation Model Ontologies to FORe

Alignment is the establishment of binary relations between the concepts of two ontologies [9]. The binary relations used to perform this operation in our case are defined in FORe.

We manually defined each of the reputation model concepts identified previously in terms of FORe. For example, the L.I.A.R. instances of the *Direct Interaction-based Reputation* concept have at least one association through the *hasInformationSource* relation to instances of the *DirectExperience*, formally defined:

$$\exists hasInformationSource(DirectExperience)$$

In addition, the Repage instances of the Image concept have at least one association through the *hasInformationSource* relation to instances of the *DirectExperience* or *Observation* or *SecondHandInformation*, formally defined:

$$\exists hasInformationSource(DirectExperience\ or\ Observation\ or\ SecondHandInformation)$$

Having the alignment, it is manually stored in the Ontology Repository. Therefore, the Ontology Mapping Service detects this new ontology and executes the Classifier

[1] http://www.w3c.org

Module (Algorithm 1) to process it. As a result, the output of such process is stored in the Translation Repository. This is made for each reputation model ontology. The mapping results generated by the OMS can be seen in [12].
In the sequence, we present an example of reputation interaction between two agents that implement different reputation models, L.I.A.R. and Repage.

7 Case Study

Let us consider an agent mediated service marketplace scenario, composed of service provider agents and consumer agents. Providers publish and provide services, while consumers search and contract services. In order to mitigate the risks imposed by an open and dynamic marketplace environment, each agent incorporates a reputation mechanism to help in the agent's decision of contracting or not services from a determined provider. In this scenario, when a consumer agent wants to contract a service, it first searches for the providers that have that kind of service available. Secondly, it exchanges reputation information with other consumer agents to evaluate the reputation of each provider and finally, it decides which provider to contract the service from, based on the reputation values received.
Suppose a system where there are two consumer agents, called Alice and Bob, implementing the proposed service-oriented architecture and using different reputation models. Agent Alice uses the Repage reputation model and agent Bob uses the L.I.A.R. reputation model. Thus, considering that we have opted for using FORe as the common reputation ontology, the OMS needs to be set up to support the translation from Repage and L.I.A.R. to FORe, and vice-versa.
Now, assume that agent Alice wants to buy a ticket to fly from Rome to Paris. In order to buy a cheap ticket, Alice searches in a list of airline companies the ones that fly those cities. Since she has never flown any of the airlines in the list, she is afraid of choosing a distrustful one. However, she knows Bob, which travels a lot in Europe. She decides to ask him about the reputation of each one of the airline companies in the list to help in her decision.
Agent Alice, with the objective of obtaining the airline companies' reputation value, activates her RRM, requesting it to query agent Bob for the information required. The RRM, through the RMI, elaborates a SPARQL [13] message using the Repage concepts. Repage concepts are used since it is the Alice inner reputation model ontology.

```
SELECT ?AgentName ?Reputation ?Value
WHERE {?x reputation:AgentName ?AgentName .
       ?x reputation:Value ?Value .
       ?x reputation:Reputation ?Reputation .
       ?x reputation:AgentReputation ?RepNature .
       FILTER ((?Reputation = true) && REGEX(?RepNature,Airline))
      }
```

The message is sent to TC that activates the LMI. This module identifies the concepts *Value*, *Reputation* and *AgentReputation* for translation. Using the OMSCI, the TC queries the OMS in order to obtain the possible translations for those concepts from the agent's reputation model ontology to the common reputation ontology, the FORe. Table 1 presents the possible translations for the concepts identified by the LMI (the

Table 1. Mapping of Repage concepts to FORe concepts

Repage Concept	FORe Concept
Value	ReputationEvaluationValue ReputationFinalValue
Reputation	SecondaryReputation
AgentReputation	ReputationNature

complete mapping is found in [12]).
Having this table, the TSI is activated to solve all the possible translation inconsistencies it may exist. In this case, the only inconsistency that exists is the translation of the *Value* Repage concept to FORe concept, since there are two possible target concepts. Selecting the third translation strategy described in section 5 and defining the heuristics to select the first option from the list, the concept *Value* translates to *ReputationEvaluationValue*. Since there is no numerical value for translation in the message, the TC does not use the VTI and the resulting translated message is:

```
SELECT ?AgentName ?SecondaryReputation ?ReputationEvaluationValue
WHERE {?x reputation:AgentName ?AgentName .
       ?x reputation:ReputationEvaluationValue ?ReputationEvaluationValue .
       ?x reputation:SecondaryReputation ?SecondaryReputation .
       ?x reputation:ReputationNature ?RepNature .
       FILTER ((?SecondaryReputation = true) && (REGEX(?RepNature,Airline)))
      }
```

Afterwards, this message is redirected to the IMI that activates the agent's IM, which encapsulates the message into a communication protocol (FIPA[2]) before sending it to agent Bob.
The agent Bob receives the message from agent Alice through its IM and extracts the information from the message. It identifies that the message is related to reputation, thus it redirect the message content to the IMI. The TC receives the message content from the IMI and sends it to the LMI to retrieve the concepts for translation from the FORe to the L.I.A.R. ontology. The LMI recognizes the concepts *ReputationEvaluationValue*, *SecondaryReputation* and *ReputationNature* from the FORe. The TC obtains the possible translations for those concepts using the OMSCI, which are shown on Table 2 (the complete mapping is found in [12]).

Table 2. Mapping of FORe concepts to L.I.A.R. concepts

FORe Concept	L.I.A.R. Concept
ReputationEvaluationValue	ReputationValue
SecondaryReputation	
ReputationNature	Facet

The TC activates the TSI in order to have all the possible inconsistencies resolved. There is only one inconsistency, related to the *SecondaryReputation* concept. Actually, there is no L.I.A.R. concept that directly relates to this FORe concept. In order to solve that inconsistency there are two possible strategies: (i) to remove the concept from the translated message or (ii) to use some kind of heuristic to determine a non-direct translation. Since the former is too simple, the strategy would be to use

[2] http://www.fipa.org

the classified ontology to identify all the L.I.A.R. concepts that are classified under the FORe concept *SecondaryReputation*, i.e, *EvaluationRecommendationBasedReputation*, *ObservationRecommendation-BasedReputation* and *ReputationRecommendation-BasedReputation*).

```
SELECT ?AgentName ?EvRcbRp ?ObsRcbRp ?RpRcbRp ?ReputationValue
WHERE {?x reputation:AgentName ?AgentName .
        ?x reputation:ReputationValue ?ReputationValue .
        ?x reputation:EvaluationRecommendationBasedReputation ?EvRcbRp .
        ?x reputation:ObservationRecommendationBasedReputation ?ObsRcbRp .
        ?x reputation:ReputationRecommendationBasedReputation ?RpRcbRp .
        ?x reputation:Facet ?RepNature .
        FILTER((?EvRcbRp = true || ?ObsRcbRp = true || ?RpRcbRp = true) &&
               (REGEX(?RepNature,Airline)))
      }
```

After the message is translated, the TC sends the message to RRI. The RRI interprets the SPARQL query and it activates the agent's RRM, which processes the request and returns a response to RRI. An example of such a response can be seen bellow.

```
((AgentName=A; ReputationValue=0.7; EvRcbRp=false; ObsRcbRp=false; RpRcbRp=true; Facet=Airline);
 (AgentName=A; ReputationValue=0.4; EvRcbRp=true; ObsRcbRp=false; RpRcbRp=false; Facet=Airline);
 (AgentName=B; ReputationValue=0.9; EvRcbRp=false; ObsRcbRp=false; RpRcbRp=true; Facet=Airline);
 (AgentName=C; ReputationValue=0.1; EvRcbRp=false; ObsRcbRp=true; RpRcbRp=false; Facet=Airline))
```

This response is sent back to agent Alice as an inform message and the translation process is executed all again, but at this time backwards.

8 Conclusions and Future Work

This paper presented a service-oriented architecture for agent interaction about reputation. It extended the general architecture presented in [17] by dividing one of its modules into two: (i) an external service for ontology mapping; and (ii) a translation module that remains inside the agent. A detailed description of those new modules and their components was presented.

In order to describe the entire interaction about reputation, an example of the service-oriented architecture use was shown. In this example, the way new modules and their components communicate was described as well as some inconsistencies translation strategies.

As future work, we intend to improve the treatment of some inconsistencies that were found during the development of the translation strategies. We also intend to evaluate the possible application of the general approach, i.e. providing agent interoperability using a service-oriented architecture based on ontology translation in different aspects of multi-agent systems other than reputation, for instance, organizational models [6].

9 Acknowledgements

This work is partially supported by CNPq and FAPESP, Brazil, as well as by USP/ COFECUB ORGMAS project.

References

1. S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. L. McGuiness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C recommendation, W3C, February 2004.
2. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. W3C recommendation, W3C, February 2004.
3. S. Casare and J. S. Sichman. Using a functional ontology of reputation to interoperate different agent reputation models. *Journal of the Brazilian Computer Society*, 11(2):79–94, 2005.
4. C. Castelfranchi and R. Falcone. Principles of trust in mas: Cognitive anatomy, social importance and quantification. *ICMAS'98*, pages 72–79, 1998.
5. R. Conte and M. Paolucci. *Reputation in Artificial Societies. Social Beliefs for Social Order*. Boston: Kluwer, 2002.
6. L. Coutinho, A. A. F. Brandão, J. S. Sichman, and O. Boissier. Model-driven integration of organizational models. *Proceedings of the 9th International Workshop on Agent Oriented Software Engineering*, 2008.
7. O. Dameron, N. F. Noy, H. Knublauch, and M. A. Musen. Accessing and manipulating ontologies using web services. *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.
8. M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building owl ontologies using the protégé-owl plugin and coode tools edition 1.0. Technical report, The University Of Manchester Stanford University, 2004.
9. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18:1–31, 2003.
10. M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.
11. G. Müller and L. Vercouter. L.i.a.r. achieving social control in open and decentralised multiagent systems. Technical report, Ecole Nationale Suprieure des Mines de Saint-Etienne, 2008.
12. L. G. Nardin, A. A. F. Brandão, J. S. Sichman, and L. Vercouter. An ontology mapping service to support agent reputation models interoperability. (to appear). *Proceedings of Workshop on Trust in Agent Societies*, 2008.
13. E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf (working draft). Technical report, W3C, March 2007.
14. J. Sabater, M. Paolucci, and R. Conte. Repage: Reputation and image among limited autonomous partners. *Journal of Artificial Societies and Social Simulation*, 9(2), 2006.
15. J. Sabater and C. Sierra. Social regret, a reputation model based on social relations. *Journal SIGecom Exch.*, 2(1):44–56, 2001.
16. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 2006.
17. L. Vercouter, A. A. F. Brandão, S. Casare, and J. S. Sichman. An experience on reputation models interoperability based on a functional ontology. *Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2007.
18. U. Visser, H. Stuckenschmidt, H. Wache, and T. Vögele. Enabling technologies for interoperability. *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*, 2000.
19. M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, June 2002.
20. G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Journal of Applied Artificial Intelligence*, 14(2):881–907, 2000.