Web Intelligence 15 (2017) 291–302
DOI 10.3233/WEB-170366
IOS Press

Applying ontologies to the development and execution of Multi-Agent Systems

Artur Freitas *, Alison R. Panisson, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira and Rafael H. Bordini

School of Informatics (FACIN), Pontifical Catholic University of Rio Grande do Sul (PUCRS), Postal code 90619-900, Ipiranga Avenue 6681, Porto Alegre, RS, Brazil
E-mails: artur.freitas@acad.pucrs.br, alison.panisson@acad.pucrs.br, lucas.hilgert@pucrs.br, felipe.meneguzzi@pucrs.br, renata.vieira@pucrs.br, rafael.bordini@pucrs.br

Abstract. Several advantages can be obtained by allowing multi-agent systems to easily access ontologies, for example, in scenarios where agents make their decisions based on knowledge provided by ontologies. Thus, this paper presents an infrastructure to allow the use of web ontologies in different agent-oriented platforms. The agents use this infrastructure layer as a tool for storing, accessing and querying domain-specific OWL ontologies. As a result, this layer allows an integration of agent platforms with semantic web data and ontologies. We exemplify in practice how agents, coded in one such platform, can use the proposed access layer to ontological reasoning engines, as well as which features can be obtained from it. We evaluated and compared performance and memory consumption of this semantic infrastructure against usual knowledge representation in agent programming.

Keywords: Ontology, Multi-Agent System, Agent-Oriented Software Engineering, CArtAgO

1. Introduction

Ontologies empower the execution of semantic reasoners [26], which provide functionalities such as *consistency checking*, *concept satisfiability*, *classification* and *realisation*. Ontologies also allow sharing a common understanding of the structure of information among people and software agents and the reuse of domain knowledge [11,22]. The integration of such semantic technologies into Multi-Agent Systems (MAS) enhances the knowledge representation features and reasoning capabilities of applications developed under these paradigms [12]. Using ontologies in MAS results in the possibility of creating logic rules that can be applied by semantic reasoners to infer new knowledge. Thus, the logic is moved from the agent code to the ontology, and the knowledge may be reused by different applications. Moreover, each agent is allowed to include these ontologies and specialise them with more specific and domain-dependent knowledge.

Our approach enables the use of ontologies within MAS, by enabling agents to reason about and query elements encoded in ontologies, such as instances, concepts and properties. Agents in such systems interact with ontologies by means of an infrastructure layer coded in a CArtAgO [24] artifact. CArtAgO offers computational abstractions and provides services that agents can exploit to support their activities. The information obtained from operations over this infrastructure may be used in agent plans to achieve goals, such as in argumentation-based negotiation/dialogue scenarios, whereupon more information can benefit the agents engaged in such process [21]. Agents can use the operations of our artifact to access and manipulate information in ontologies, as we show in further sections, using the Jason [5] agent platform to access ontologies in OWL [3].

*Corresponding author. E-mail: artur.freitas@acad.pucrs.br; Tel.: +55(51)3320-3500.

This paper makes the following contributions: (i) developing an infrastructure layer (artifact) coded in CArtAgO to enable ontology reasoning and querying features in different agent-oriented platforms; (ii) describing and implementing scenarios in Jason agent platform using the operations provided by such infrastructure; (iii) evaluating and comparing performance and computational resource consumption of this new knowledge representation approach (of accessing ontologies by the infrastructure) against representations that use the belief base of agents; (iv) discussing advantages, limitations and trends of enabling agents to access the knowledge from ontology to support their decision making.

This paper is structured as follows. Section 2 explains a theoretical background on MAS and ontologies. Then, Section 3 proposes an architecture, based on a CArtAgO artifact, working as an infrastructure layer to provide ontology manipulation capabilities in agent platforms. Section 4 uses this artifact to access an OWL ontology in the context of Jason agents. We explain the ontology used, reasoning examples and how it can support the decision making of agents. Next, in Section 5, experiments are used to compare performance and memory consumption of our approach against an agent reasoning that uses only the regular agent's belief base. Finally, we discuss related work in Section 6 and expose our final remarks and outline research directions in Section 7.

2. Ontologies and Multi-Agent Systems

Ontology is defined as an *explicit specification of a conceptualisation* [11], where a *conceptualisation* is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualisation, explicitly or implicitly [11]. Some essential properties of ontologies are [12]: (i) ontologies describe a specific domain; (ii) ontology users agree to use the terms consistently; (iii) ontology concepts and relations are unambiguously defined in a formal language by axioms and definitions; (iv) relationships between ontology concepts determine the ontology structure; and (v) ontologies can be understood by computers. More importantly, ontologies empower the execution of semantic reasoners which provide functionalities such as *consistency checking*, *concept satisfiability*, *classification* and *realisation*.

Ontologies are knowledge representation structures, usually based on Description Logics, composed of concepts, properties, individuals, relationships and axioms [2]. A *concept* (or class) is a collection of objects that share specific restrictions, similarities or common properties. A *property* expresses relationships between concepts. An *individual* (instance, object, or fact) represents an element of a concept. A *relationship* instantiates a property to relate two individuals. And an *axiom* (or rule) imposes constraints on values of concepts or individuals normally using logic languages (that can be used to check ontological consistency or infer new knowledge). Nowadays, the most prominent ontology language is OWL (Web Ontology Language), which is a language for processing web information and semantic web standard formalism to explicitly represent the meaning and relationships between terms [3].

We argue that those properties of ontologies have a role to play in MAS. Agents are reactive systems that can independently determine how to best achieve their goals and perform their tasks while having properties such as autonomy, reactivity, pro-activeness, and social ability [5]. Ontologies are investigated in MAS for improving communication [1], sharing knowledge among agents [25], enabling personal assistant agents [23], and so on. In fact, ontologies are useful in several other related areas, such as information gathering [27], knowledge management [13], information security [29], and decision support [17]. In this context it is easy to observe how promising can be the investigation on the use of ontologies embedded in agent systems.

Although the advantages of ontologies for agents are clear, few MAS platforms currently integrate ontology techniques. Limited ontological support is provided by agent-oriented software engineering approaches since they do not incorporate ontologies throughout the entire systems development life cycle nor consider ways in which ontologies can be used to account for interoperability and verification during design [28]. Considering such context, this work investigates the interconnected use of ontologies in MAS.

3. Engineering ontology-based agents

There are many agent-oriented programming platforms, such as [4] Jason, Jadex, Jack, AgentFactory, 2APL, GOAL, Golog, and MetateM. Those languages differ in the agent architecture used, in their form of communication/interaction, and also on the programming paradigms that inspired or underlie each lan-

guage. Our proposal to interact with ontologies can be used in any agent platform that supports CArtAgO [24]. In this work we used Jason [5] to demonstrate the applicability of such approach. Jason is one of the best-known languages inspired by the BDI (*Beliefs-Desires-Intentions*) architecture. It is an open source interpreter that offers several features such as speech-act based agent communication, plans annotation, architecture customisation, distributed execution and extensibility through internal actions.

As previously explained, ontology is defined in computer science as an *explicit specification of a conceptualisation*. In other words, it means an abstract model of some world aspect that specifies properties of important concepts and relationships. The use of ontology in agents is motivated by the needs of improving knowledge representation and enabling the execution of semantic reasoning. For example, in OWL, a given class `C` can be declared with certain conditions (i.e., every instance of `C` has to satisfy these restrictions, and/or every instance that satisfies these restrictions can be inferred as belonging to `C`). OWL class restrictions [3] can be defined by elements such as cardinality and logic restrictions (e.g., union, intersection, complement, the universal and the existential quantifier). These restrictions allow to make inferences by using semantic reasoners over the ontology, which are important features to provide to agent when building complex artificial intelligence systems.

A comparison about the integration of ontologies within MAS is discussed with more detail in Section 6, which focuses on related work. In short, AgentSpeak-DL [20] is a language which appears in a paper that does not implement it in any agent platform; JASDL [16] is an AgentSpeak-DL implementation directly in Jason; and CooL-AgentSpeak [19] is implemented in a way that each agent ontology is private. Our approach differs in the sense that ontologies can be shared among more than one agent and the ontologies can be used in several agent platforms. These features are obtained based on the architecture we designed that is implemented in CArtAgO [24]. CArtAgO is a platform to support the artifact notion in MAS. Artifacts are function-oriented computational abstractions which provide services that agents can exploit to support their activities. As design and implementation decision, each instance of our artifact can load and encapsulate exactly one OWL ontology. However, each workspace can have any number of instances of this artifact, where each instance makes reference to an ontology, and the agents can observe and manipulate any artifacts whenever they are both located in a same workspace. Thus, MAS using our approach can handle multiple ontologies.

The approach proposed in this research is an alternative to agents in which the knowledge is represented and manipulated by means of ontologies, instead of using a platform-specific mechanism (such as a belief base). However, an agent may still use its regular knowledge representation approach simultaneous with the new approach proposed here (or completely replace the old approach). As this paper demonstrates, the use of our mechanism provides advantages in terms of expressiveness, interoperability, and performance. Our infrastructure layer implemented in CArtAgO provides ontology features to agents by using the OWL API [14], which allows to create, manipulate and serialise OWL ontologies. An artifact makes its functionalities available and exploitable by agents through a set of operations and a set of observable properties [24]. Operations represent computational processes executed inside artifacts, that may be triggered by agents or other artifacts.

An architectural illustration of our approach can be seen in Fig. 1, where we have 3 workspaces with different configurations. As described, each workspace can have any number of instances of CArtAgO artifacts, and each artifact loads and encapsulates an OWL ontology. The agents can observe and manipulate the correspondent ontologies depending on the artifacts available to them in their workspaces. Further, the agents can still use their regular knowledge representation approach (e.g., belief base) simultaneous with the new approach proposed here, e.g., Workspace 1 in Fig. 1, or completely replace the old approach, e.g., Workspace 2 in Fig. 1. The usual approach, without using our proposed artifact to interact with ontologies, is shown in the Workspace 3 of Fig. 1.

Figure 1 clearly demonstrates that our approach allows agents to share the same ontology, including agents from different workspaces (e.g., the agents on Workspace 1 and 2 are sharing the Ontology 2), as well as it allows the agents to use information from specific ontologies based on their role in the MAS. Figure 1 is showing just one possible configuration of MAS, however, we emphasise that different configurations are possible depending on the resources provided by the multi-agent platform in use. In other words, our approach requires a platform that supports CArtAgO artifacts. Our artifact provides the following operations:

– **addInstance**(instance): adds the new instance in the ontology (without defining any concept to it);

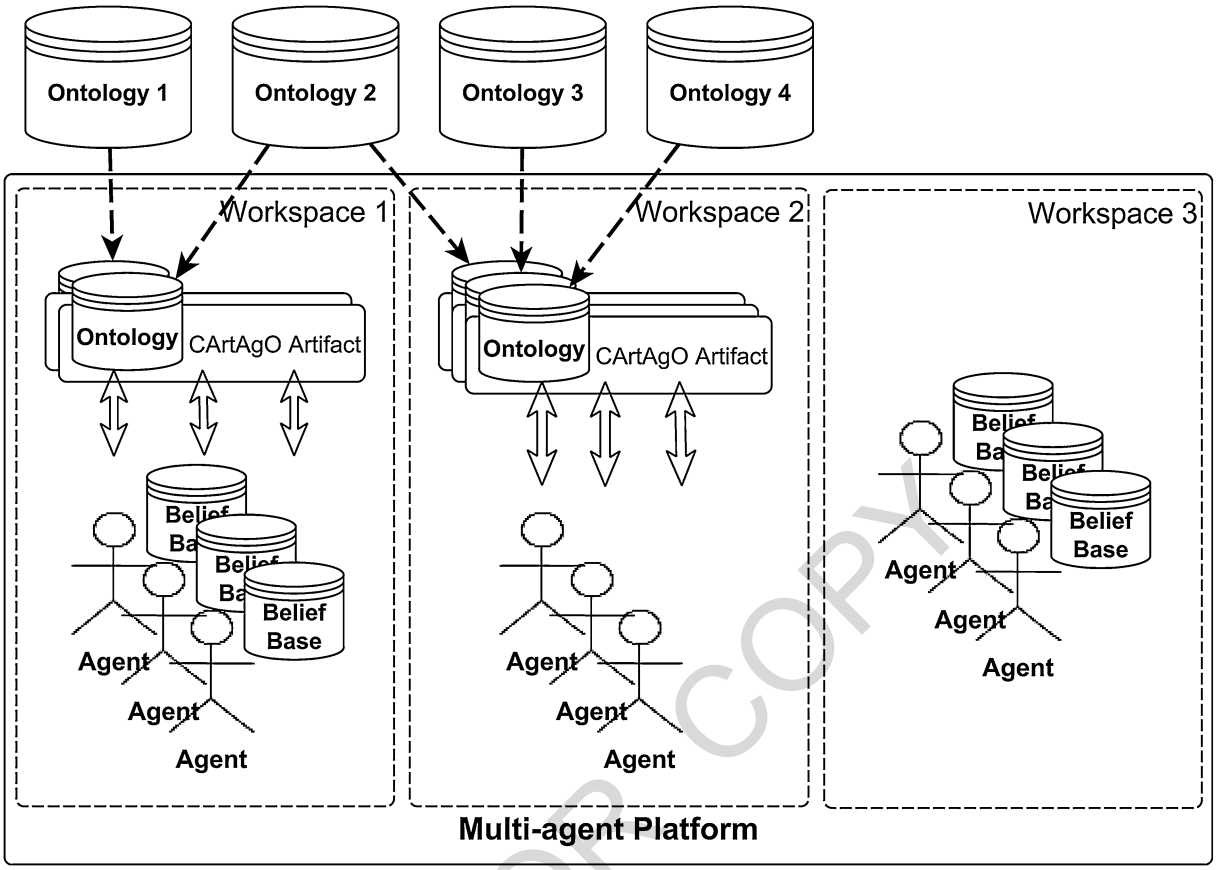

Fig. 1. Example of agents using the proposed approach (Workspace 1 and 2) versus usual multi-agent systems (Workspace 3).

- **addInstance**(instance, concept): adds the new instance and sets its type as the given concept;
- **isInstanceOf**(instance, concept): verifies if the instance belongs to the given concept, returning a boolean value;
- **getInstances**(concept): retrieves a set of instances classified in a specific concept, returning a Set<OWLNamedIndividual>;
- **addProperty**(domain, property, range): adds a relationship among the specified instances;
- **isRelated**(domain, property, range): verifies if there is a specific kind of property among the given instances, returning a boolean value;
- **getInstances**(domain, property): retrieves the instances that are targeted by the given domain and property, returning a Set<OWLNamedIndividual>;
- **addConcept**(concept): adds the new concept in the ontology;
- **isSubConceptOf**(subConcept, superConcept): verifies if the first concept is subclass of the second one, returning a boolean value; and
- **getConcepts**(instance): retrieves the set of concepts (asserted and inferred) for the given instance, returning a Set<OWLClass>.

4. Usage examples of the ontology artifact

We explain the use of our approach with a scenario commonly used in the agent literature: suppose a MAS which represents a soccer team and that each role is represented by concepts in an ontology. For example, a soccer team has players who can be right midfielders, which specialises the concept of midfield, which is a subclass of player, and so on. In certain moments the coach agent of a team needs to choose a player to replace other. To make its decision the coach agent just needs to look for the corresponding ontology concept and choose a player among the individuals of that concept. This Section explains how agent decision making with ontological knowledge can be coded in Jason

using our proposed approach. Our examples are illustrated based on the commented soccer scenario.

4.1. Agent decision making using ontology information

Decision making is a process where an agent looks for the information available to it in order to decide which course of action to follow [15]. This information generally comes from its environment perceptions, its initial beliefs, or from the communication with other agents (i.e., beliefs from different sources). This work proposes an infrastructure layer in the form of a CArtAgO artifact to access domain specific knowledge provided by ontologies in a way that agents can use such information to make their decisions. Decision making is one example of how our approach may be employed, however it can be used in other domains where information and reasoning provided by ontologies is necessary or useful. In our example, the coach agent uses an ontology describing the team members and the roles of each agent/player in several situations (e.g., to retrieve information, reason and make its decisions).

Since Jason is used in our examples, it is important for the reader to be familiar with its syntax and semantics. Plans in Jason have the following format:[1]

```
triggering_event: context <- body
```

The `triggering_event` represents an event related with an agent goal (or belief) that will cause the plan to be put in practice, and it has a format such as `!goal(Parameter)`. The `context` establishes the required preconditions to perform that plan. Lastly, the `body` defines the sequence of actions and subgoals to fulfil that plan.

According to the soccer team scenario, the ontology concepts model soccer roles, such as Player, Midfield and Right Midfield (represented as concepts such as `C1`, `C2` and `C3`). The instances may represent Players, e.g., `i1` can be a player whose role is Right Midfield (concept `C3`). The instance `i1` can be related with `i2` through `r1` (e.g., `r1` can be defined as "is a player less defensive than"). Suppose an agent that needs to make a decision about which course of action to follow considering its context that is represented in an ontology. This decision can be guided, for example, by checking if a particular individual belongs to a particular concept. Using operations provided in the CArtAgO artifact presented in this paper to access the ontology, the agent can obtain the required information by executing the operation `isInstanceOf`, as shown in the Jason plan below. This operation returns a *boolean*, which is *true* if the individual queried, `i1`, belongs to a given concept `C3`, or *false* in the other case. The return unifies with the last parameter of the operation (`R`), which the agent uses to decide between executing `action_01` or `action_02`. Suppose a coach that needs to choose a player in some position, which is done by querying player agents that belongs to the desired role encoded as concepts in the ontology. For example, if in a given moment a player is injured, then the coach agent needs to scale another player in that position. To make this, the coach checks if a player agent belongs to the right midfield role. In this scenario, the coach has perceived that the injured player plays in front, but it does not remember its exact role (right midfield or left midfield). After checking this information, which is encoded as concepts in the ontology, the coach can make the decision of scaling a new player.

```
+!goal1: context
    <- isInstanceOf("i1","C3",R)
       if(R){ action_01 }
       else{ action_02 }
```

Now, suppose that an agent needs to recover all individuals who participate in a particular relationship. The agent can use this information to make a decision about the existence of an individual in the returned set, or to select one of these individuals for a particular need. In this case, the operation `getInstances` can be used, as presented in the following Jason plan. The return of this operation is a set of individuals which have that relationship (`r1`) with the given instance (`i2`). Then, this plan tests if the set returned is empty, which leads to the execution of `action_03` if *true*, or in the other case the agent will pursue a goal involving a new decision making which uses the set of individuals returned (`goal3`). In our example, suppose that the coach wants to scale more defensive players to replace a particular player (`i2`) using the relation *defensive_substitution* (`r1`) which returns the list of defensive substitutions available for that player. If the list is empty, the coach may decide to reposition the players to have a more defensive team (`action_03`). In other case, where there is at least one player more defensive to substitute `i2`, the coach may choose one player of this set to be scaled (`!goal3(Set)`).

[1] We refer the reader to [5] for more details about the syntax and semantics of the language.

```
+!goal2: context
    <- getInstance("i2","r1",Set)
       if(.empty(Set)){ action_03 }
       else{ !goal3(Set) }
```

5. Comparing ontology and agent approaches

An agent can represent its knowledge within its internal structures (e.g., its belief base), or in external structures (e.g., an ontology). This paper shows an artifact for agents to work with knowledge represented in ontologies, and such approach offers advantages in terms of expressiveness and reusability. More expressiveness is obtained by the execution of semantic reasoners over the ontology; and more reusability comes from the possibility of different platforms updating and querying the same repository and formalism. Despite these improvements, when engineers have to consider the development of distributed applications, such as MAS, there is a need to consider if there will be any loss in applications using the proposed ontological infrastructure. Therefore, we evaluated our approach considering two important issues to put it in practice: performance in terms of execution time and computational resources consumption in terms of allocated memory.

First, we wanted to measure how fast each approach can be, in terms of their execution time to perform similar operations. In this case, we compared two environments with exactly the same inputs and outputs. Among these environments, the only change lies in the computational approach, since one considers the use of ontology to conduct operations. Our tests aim at measuring any performance gain (or loss) in the response time when the ontological approach is compared against the traditional approach. This evaluation is very important for the development of distributed applications, where we are interested in applications that respond in a timely manner. Therefore, if the infrastructure causes considerable performance loss in response time, we should be aware of and careful about its use in practice within commercial applications. Secondly, our experiments measure the computational resources consumption of using each of the two approaches (the traditional and our new one). This evaluation is interesting when considering the development of distributed applications, where servers have limited computational resources, and multi-agents applications instantiate a big number of agents. Therefore, if a given infrastructure consumes less computational resources, then it presents an advantage over the others. Our experiments are considering the impact on memory consumption of different infrastructures for such systems.

Table 1
Statements in ontologies and in Jason code

Statement	Ontology	Jason
x is instance of A	x : A	A(x)
x has property P targeting y	(x,y) : P	P(x,y)
B is subclass of A	$B \sqsubseteq A$	A(x) :- B(x)
If B then A (B implies A)	$B \Rightarrow A$	A :- B

Table 2
Some examples of operations in ontologies and in Jason code

Operation	Ontology artifact	Jason
x is instance of A?	isInstanceOf(x,A)	?A(x)
P relates x with y?	isRelated(x,P,y)	?P(x,y)
Add: x is instance of A	addIntance(x,A)	+A(x)
Add: P holds in (x,y)	addProperty(x,P,y)	+P(x,y)

To compare the ontology reasoning with the reasoning executed only in the agents, we defined ways to convert ontology statements to agent code, as depicted in Table 1. These equivalences allow us to execute both approaches (which will return the same result) to compare their performance (i.e., the performance of reasoning with the ontology against simulating the same reasoning inside agents). Thus, the proposed artifact offers a new way to represent knowledge and new operations compared when using Jason alone and the proposed CArtAgO artifact to integrate agents with ontologies. For simplicity reason, Table 1 shows only the main statements which were used to test our approach of reasoning with the ontology in order to compare it with simulating the same reasoning only inside agents. Table 2 shows some examples of operations using our artifact to interact with ontologies and the equivalent operations using purely Jason code.

5.1. Experiments description

Our experiments compare the performance and the amount of resource consumption of executing agent plans that follow one of these two approaches for handling knowledge (internal or external structures). One approach uses our CArtAgO artifact to query information from ontologies; and the other approach queries the knowledge stored in the agent belief base. When using ontologies the number of individuals is increased, and when using the belief base the ontolo-

gies were converted to beliefs and rules in Jason. In both approaches we measured and compared the execution time or the amount of resource consumption for an agent to retrieve its information (from queries in an ontology or from its belief base).

The ontology used has 3 concepts (e.g., C1, C2 and C3) defined such as C3 is subclass of C2, and C2 is subclass of C1. The number of individuals ranges from 100 to 100.000, which are asserted to the most specific concept, in this case C3. The executed queries verify if an individual is an instance of the most specific (C3) and the most generic concept (C1). These queries were performed and compared both in ontology reasoning, and in Jason, and these queries return true, since an instance of C3 is inferred as C1 and the queried individuals were asserted as C3. All tests were executed in the same computer, which is a Mac Pro Server (OS X 10.9.4) with two 6-core Intel Xeon (2.4 GHz) CPU, 32 GB of RAM (DDR3 1333 MHz) and 2 TB of disk storage. Regarding software, we used Java SDK 1.7 (build 1.7.0_65-b17), Jason 1.3.9, OWL API 2 version 3.50 and HermiT reasoner version 1.3.8.

The Sections 5.2 and 5.3 present, respectively, specific details on our experiments considering execution time and memory allocation. In all cases, we simply prepared input files to represent each desired scenario to be executed. As illustrated in our graphs, only one variable was selected to be changed between each simulation, such as the amount of information, or the total number of agents. Then, the experiments were conducted as follows. In the execution time tests, the time was measured for different operations, and the measurement of time begins immediately before executing the desired operation and ends exactly with the return of that operation. In the memory allocation tests, the values were measured before putting the system in execution and after loading everything in order to calculate the amount of consumed memory. This metric was calculated by subtracting the value obtained when the system is up and running, from the value measured before initialising the infrastructure. More details about our experiments are given in the following subsections.

5.2. Experiments in execution time

The results demonstrate that Jason performance can be improved by using our new approach instead of querying and reasoning only with the regular belief base. The execution time was measured to retrieve the same information, however in one case it is represented and retrieved from the ontology using our artifact, and in the other case it is stored and queried in the regular Jason's belief base. When using ontologies, we tested two alternatives: with or without the execution of a semantic reasoner (respectively, Hermit and Structural). We queried for asserted, inferred and nonexistent knowledge. An example of query on asserted knowledge that returns true is to verify if an instance has a type that is explicitly asserted to it (in our tests we have several instances asserted as C3). An example of query on inferred knowledge that returns true is to verify if an instance has a type which is not explicitly defined but can be inferred (e.g., C3 is an indirect subclass of C1, therefore all instances of type C3 are instances of C1 too). Nonexistent knowledge queries will return false since the knowledge is not explicitly asserted neither can be inferred in any way.

Our experiments demonstrate that the proposed approach enhances Jason's performance, and also offers advantages of reuse and expressiveness. In scenarios with a low number of instances (Fig. 2) we see a minor loss in performance, however if we consider instance rich ontologies then the proposed approach shows improvement (Fig. 3). We would like to highlight that these two approaches are not mutually exclusive, which means that the agent programmer can choose to use them together, or just one if desired.

The experiments consider different sizes of ontologies, for example, Fig. 2 shows our results with instances ranging from 100 to 1.000 instances. The results using ontologies with more instances (until 100.000 instances) are depicted in Fig. 3. All performance tests (so far) demonstrate that the best performance is obtained when using our artifact with the Structural approach. When considering a large number of instances, the worst performance obtained comes from using Jason's regular belief base. When retrieving inferred information (i.e., it is not explicit as-

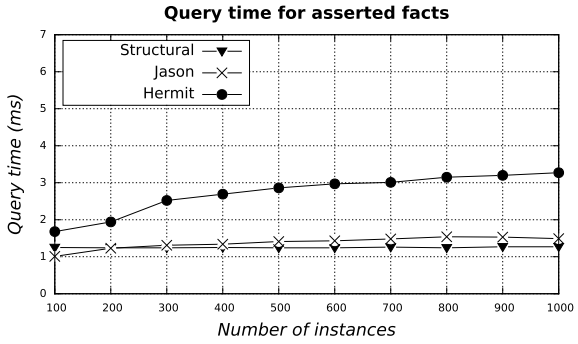

Fig. 2. Performance to retrieve asserted knowledge from a small number of individuals (instance of C3 axiom).

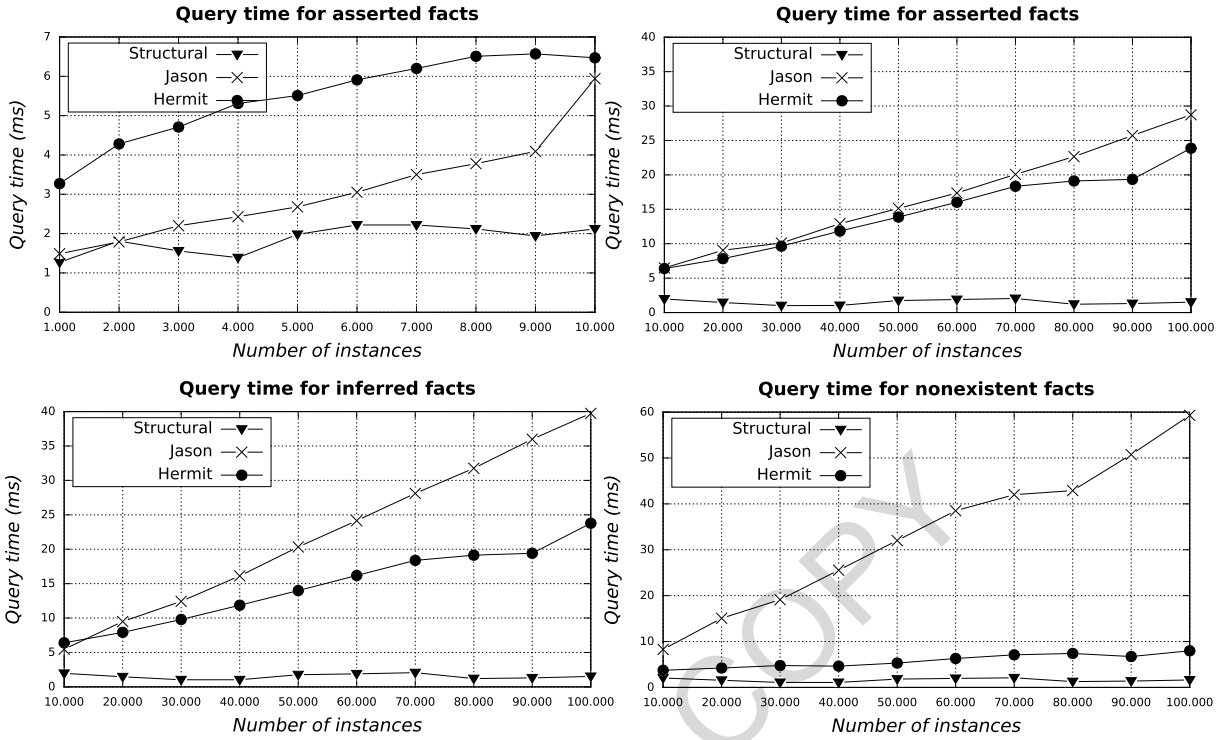

Fig. 3. Performance to retrieve asserted, inferred and nonexistent knowledge using ontology versus agent approaches.

serted), the performance of ontological approaches is similar for asserted facts. However, the regular belief base of Jason takes more time to apply the rules and return the result. When retrieving nonexistent information (which is not explicit asserted and cannot be inferred), the performance of ontologies is similar to previous ones. However, the regular belief base of Jason takes even more time than the previous cases.

The experiments measure the execution time of a Jason plan which uses an operation of our CArtAgO artifact (e.g., *isInstanceOf*). The time was measured for a hundred operations, and the sum of these values was divided by one hundred to obtain the average time of a single operation. We used this approach to obtain more accurate results by calculating an average that avoids spikes (too low or high values). This process was repeated ten times, and the final result is an average of these ten executions, each one executing a hundred of operations. The instances queried are selected based in the calculation of an interval ($interval = \frac{NumberOfInstances}{NumberOfQueries}$), where it is ensured that it will be selected members of all set, and not only members at the start or at the end of the set (uniformity). We performed ten executions of each test, and we obtained very similar results among the tests. Thus, we have concluded that the differences in our measurements are statistically insignificant and there would be no need to report error bars graphs to show this.

5.3. Experiments in memory consumption

In regards the consumption of computational resources, we evaluated memory allocation, which is an important factor of consideration when we are interested in developing multi-agent applications with many instances of agents. Some of our results are presented in Figs 4 and 5. For an application with just one agent, the traditional approach consumes less memory, as observed in the left side of Fig. 4, considering that the infrastructure has a small weight to be loaded. This cost is overcome once that a few agents start to share their knowledge instead of replicating the information privately, as shown in the right side of Fig. 4. As can be observed in Fig. 5, the consumption of memory in the ontological approach is very inferior compared to the traditional approach (agents using belief bases) for 10 or more agents. The explanation for such results is because the knowledge is not replicated for all agents in the system, but, instead, they have access to such information by means of our proposed infrastruc-

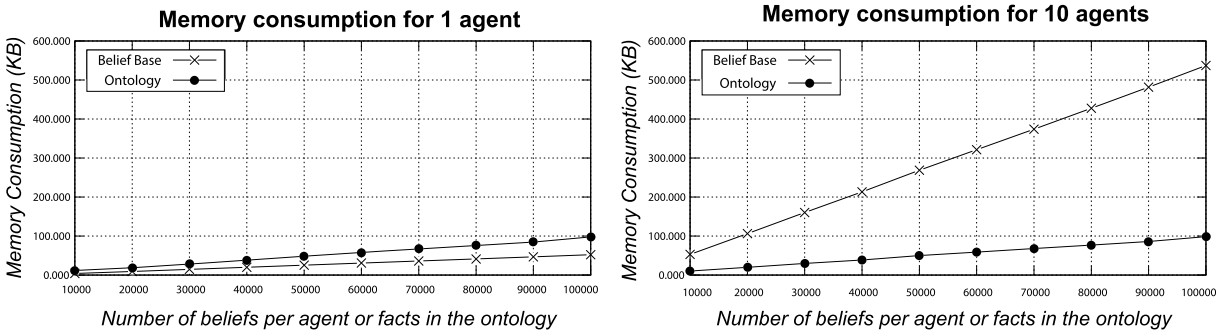

Fig. 4. Memory consumption of approaches based on ontology versus regular belief base of agents in Jason (considering a fixed number of agents and varying the number of facts in a shared ontology versus the number of beliefs per agent).

ture. Of course, a centralised knowledge base brings the problem of a single point of failure. To deal with this problem, which is common in distributed systems, we could have some replicated knowledge bases.

Some details about how our experiments were implemented are given next. For measuring memory consumption, we read the value of memory allocation in two moments: before putting the system in execution and after loading everything. The amount of consumed memory for a given approach is obtained from subtracting the metric obtained when the system is up and running, from the metric measured before initialising the infrastructure. We implemented in Java an internal action to collect the amount of allocated memory, so we could obtain such metric inside agent code. The first thing done in our method to prepare the measurement in memory is invoking the Java garbage collector with the command `Runtime.getRuntime().gc()`. The collector forces the Java Virtual Machine (JVM) to expend effort toward recycling unused objects in order to make the memory currently occupied by them available for quick reuse. When control returns from the method call, the JVM has made its best effort to recycle all discarded objects. After that, the consumed memory is obtained by subtracting the free memory from the total memory. The `Runtime.getRuntime().totalMemory()` returns the total amount of memory currently available for current and future objects in the JVM. The `Runtime.getRuntime().freeMemory()` returns the amount of free memory in the JVM, which means the total amount of memory currently available for future allocated objects. These values are obtained in bytes, and can be converted to other scales if needed.

The Jason internal action `create_agent` is used for creating the desired number of agents for our tests.

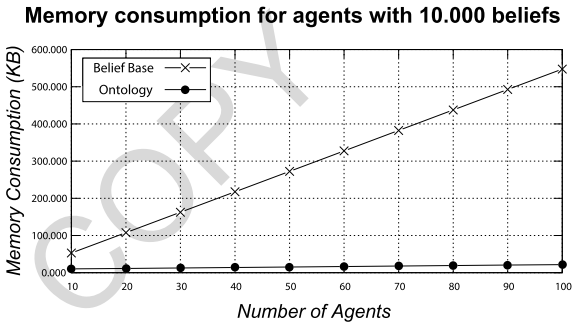

Fig. 5. Memory consumption of approaches based on ontology versus regular belief base (considering a fixed number of beliefs and varying the number of agents in Jason).

In the infrastructures that use our artifact to interact with ontologies, a single artifact is created through the action `makeArtifact`, regardless of the number of agents that will be created in the test. If there is an artifact in use, each created agent performs the following actions from CArtAgO: `joinWorkspace`, `lookupArtifact`, and `focus`. When working without the artifact, the `include` instruction is used to add the required information in the belief base.

Some differences regarding our experiments are worth to highlight. In our performance tests, the number of agents was fixed in one, and the metric being manipulated is the amount of information. In one approach the knowledge is encoded purely in the belief base of agents, and in the other it is stored in an ontology. Our memory allocation tests considered two scenarios. The scenario depicted in Fig. 5 fixes the amount of knowledge, but varies the number of agents that are storing and able to access it. The other scenario is depicted in Fig. 4, where the number of agents is fixed (it is just 1 or 10), and the knowledge in the system is being changed.

6. Related work: Agents & ontologies

AgentSpeak-DL [20] is an agent-oriented programming language that extends agents' belief bases with Description Logic. The advantages of integrating ontologies with agents are: (i) more expressive queries in their belief bases, since results can be inferred from ontologies and thus are not limited to explicit knowledge; (ii) refined belief update given that ontological consistency of a belief addition can be checked; (iii) the search for a plan to deal with an event is more flexible (not limited to unification), i.e., subsumption relationships between concepts can be considered; and (iv) agents can share knowledge using ontology languages, such as OWL. AgentSpeak-DL extends agents' belief base with Description Logic in order to include: (i) one immutable TBox (terminological box) that characterises the domain concepts and properties; and (ii) one ABox (assertion box) with dynamic factual knowledge that changes according with the results of environment perception, plan execution and agent communication. Summarising, AgentSpeak-DL enriches agents' belief bases with the definition of complex concepts that can go beyond factual knowledge [20].

JASDL [16] implements AgentSpeak-DL in Jason to merge agents' belief bases with ontological reasoning. It provides ontology manipulation capabilities to agents, i.e., it is a practical approach for using ontologies and semantic reasoning in Jason agents. Agent programmers benefit from features such as plan trigger generalisation based on ontologies and the use of such knowledge in belief base querying. Jason modules were altered to implement JASDL, such as the belief base (that was extended to partly resides within an ontology and a DL reasoner), the plan library and the agent architecture. JASDL provides reuse of ontological knowledge, new inferences that an agent can make based on its beliefs, knowledge consistency, enhanced plan searching; and improved message processing with semantically-enriched inter-agent communication.

CooL-AgentSpeak [19] extends AgentSpeak-DL with plan exchange and ontology services. It implements a CArtAgO artifact functioning as ontology repository which stores a possibly dynamic set of ontologies and offers ontology matching/alignment features. It searches for ontologically relevant plans not only in agent's local plan library, but in other agents' libraries too, according to a cooperation strategy (that is not based solely on unification and on the subsumption relation between concepts, but also on ontology matching). In short, CooL-AgentSpeak [19] performs cross ontological unification for agents that do not disclose their ontologies to each other (that cooperate while preserving their privacy).

Our approach differs in some points. First, we implement an infrastructure layer which works as an interface between ontologies and MAS using a CArtAgO artifact that can be reused in several MAS platforms. On the other hand, AgentSpeak-DL [20] targets only AgentSpeak, and JASDL [16] addresses only Jason. CooL-AgentSpeak [19] also uses CArtAgO as a mean to integrate ontologies and agents, but our work differs from this one since we assume that agents may share their ontologies, while in CooL-AgentSpeak the agents do not share their ontologies. A comparison among such related work is depicted in Table 3.

There are other papers on using ontologies in MAS to provide a common vocabulary for communication. Although our work focuses on a more general architecture, for completeness we describe some of them below. One work [22] proposes the design of an intelligent speech interface for Personal Assistants applied to knowledge management. The authors argue that an effective personal agent should have two main sources of knowledge available. The first contains shared knowl-

Table 3
Comparing related work in multi-agent systems with ontologies

Research	Overview of the work	Ontologies included	MAS platforms used
AgentSpeak-DL [20]	An approach for using ontologies during agent reasoning to extend agents' belief base with DL	It is a way for agents to represent knowledge and interact with ontologies	AgentSpeak
JASDL [16]	An implementation of AgentSpeak-DL in the Jason platform	Jason agents can represent knowledge and interact with ontologies	Jason
CooL-AgentSpeak [19]	An extension of AgentSpeak-DL with plan exchange and ontology services	Each agent has access only to its private ontologies	Jason
Our approach	A CArtAgO infrastructure to integrate multi-agent platforms with ontologies	Agents can access and manipulate shared ontologies using our artifact	Any platform supporting CArtAgO artifacts (e.g., Jason, JaCaMo)

edge (ontology) in order to provide a structured representation of domain knowledge to applications. The second source are memories, which are responsible for describing previous records of users' behaviour. In [22], ontologies play the role of providing knowledge for agents to process the user's statement and reasoning (e.g., to provide a vocabulary for the application, including the meaning of terms). Different from [22], we propose a general architecture for using ontologies for both developing and executing MAS.

In [18], the authors also focus on communication in MAS. That paper claims that, given different terminology and incompleteness of information pieces among ontologies used by different agents, understanding through communication is rather complex to achieve. Therefore, the authors propose a distributed version of description logic to model the mappings between ontologies.

In [30], the authors propose a MAS with collective reasoning, which results from methods for cooperation and coordination. In that work, a key point is the definition of a central administrator agent that is responsible for forming a knowledge pool about the domain, providing domain knowledge for agents.

Furthermore, we have done previous research based on which we first introduced our infrastructure for agent platforms to interact with ontologies [8]. The current paper is an extended version of the work presented in [8]. Also, we developed applications of such proposed approach to using ontologies as source of information by agent systems in scenarios where agents apply ontological information about tasks to achieve, for example, their goals of task management, recognition, negotiation, and reallocation [21,25]. In other papers, we have considered different research directions towards combining ontology and multi-agent technologies, in which we provided tools for engineering MAS using an ontology as a meta-model [7]. That work extends our ideas towards models of MAS represented as abstractions in ontologies [6,9]. We also investigated approaches for using an ontology to represent planning domains in HTN (Hierarchical Task Network), which can be used to specify and convert plans between ontological and Jason formats [10].

7. Final remarks

The integration of agent platforms with ontologies enables agents the ability to operate in a Semantic Web context. This work investigates how to enable current agent-oriented development platforms to transparently merge with such semantic technologies. As result, developers obtain new features for developing complex software systems with a semantic infrastructure that applies software and knowledge engineering principles. The development of applications that integrate semantic and agent technologies is still an open challenge. To address this issue, we pointed out that ontology languages offering semantic querying and reasoning should be suitably integrated into agent development frameworks.

Our implementation to integrate ontologies within agents uses an artifact implemented in CArtAgO [24] that provides agents the ability to reason and manipulate ontologies. Our infrastructure is applicable to several agent-oriented platforms to engineer ontology-based artificial intelligence and distributed applications, and we demonstrate how to use it in Jason [5] to access ontologies in OWL [3]. We measured performance and computational resource consumption of our approach and compared with an alternative one which stores all the knowledge inside the agent. Our experiments demonstrated that the proposed technology enables the development of new and more powerful applications. However, these approaches can be used together, in other words, an agent can represent part of its knowledge in its own belief base and part in ontologies to be accessed using our described artifact. As future work, we plan to carry out experiments to compare the use of our approach together with and against other agent platforms.

Acknowledgements

Part of the results presented in this paper were obtained through research on a project titled "Semantic and Multi-Agent Technologies for Group Interaction", sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91. We are grateful for the partial support given by CNPq and CAPES.

References

[1] M. Afsharchi, J. Denzinger and B.H. Far, Enhancing communication with groups of agents using learned non-unanimous ontology concepts, *Web Intelligence and Agent Systems* **7**(1) (2009), 107–121. doi:10.3233/WIA-2009-0157.

[2] F. Baader, I. Horrocks and U. Sattler, Description logics, in: *Handbook on Ontologies*, S. Staab and R. Studer, eds, Springer, 2009, pp. 3–28. doi:10.1007/978-3-540-24750-0_1.

[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider and L.A. Stein, OWL web ontology language reference, Technical report, W3C, 2004. Available at: http://www.w3.org/TR/owl-ref/.

[4] R.H. Bordini, M. Dastani, J. Dix and A.E.F. Seghrouchni, *Multi-Agent Programming: Languages, Tools and Applications*, Springer, 2009. doi:10.1007/978-0-387-89299-3.

[5] R.H. Bordini, J.F. Hübner and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*, John Wiley & Sons, 2007. doi:10.1002/9780470061848.

[6] A. Freitas, R.H. Bordini, F. Meneguzzi and R. Vieira, Towards integrating ontologies in multi-agent programming platforms, in: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 3, 2015, pp. 225–226. doi:10.1109/WI-IAT.2015.207.

[7] A. Freitas, L. Hilgert, S. Marczak, F. Meneguzzi, R.H. Bordini and R. Vieira, A multi-agent systems engineering tool based on ontologies, in: *34th International Conference on Conceptual Modeling*, Lecture Notes in Computer Science, Springer, 2015.

[8] A. Freitas, A.R. Panisson, L. Hilgert, F. Meneguzzi, R. Vieira and R.H. Bordini, Integrating ontologies with multi-agent systems through CArtAgO artifacts, in: *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2015, pp. 143–150. doi:10.1109/WI-IAT.2015.116.

[9] A. Freitas, D. Schmidt, A. Panisson, F. Meneguzzi, R. Vieira and R.H. Bordini, Applying ontologies and agent technologies to generate ambient intelligence applications, in: *Joint Proceedings Collaborative Agents – Research & Development, CARE for Intelligent Mobile Services & Agents, Virtual Societies and Analytics*, 2014, pp. 22–33. doi:10.1007/978-3-662-46241-6_3.

[10] A. Freitas, D. Schmidt, A. Panisson, F. Meneguzzi, R. Vieira and R.H. Bordini, Semantic representations of agent plans and planning problem domains, in: *Engineering Multi-Agent Systems*, F. Dalpiaz, J. Dix and M.B. van Riemsdijk, eds, Lecture Notes in Computer Science, Vol. 8758, Springer, 2014, pp. 351–366. doi:10.1007/978-3-319-14484-9_18.

[11] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* **5**(2) (1993), 199–220. doi:10.1006/knac.1993.1008.

[12] M. Hadzic, P. Wongthongtham, T. Dillon and E. Chang, *Ontology-Based Multi-Agent Systems*, Studies in Computational Intelligence, Springer, 2009. doi:10.1007/978-3-642-01904-3.

[13] C.W. Holsapple and K.D. Joshi, A formal knowledge management ontology: Conduct, activities, resources, and influences, *Journal of the American Society for Information Science and Technology* **55**(7) (2004), 593–612. doi:10.1002/asi.20007.

[14] M. Horridge and S. Bechhofer, The OWL API: A Java API for OWL ontologies, *Semantic Web* **2**(1) (2011), 11–21. doi:10.3233/SW-2011-0025.

[15] M. Kaufman, Local decision-making in multi-agent systems, PhD thesis, Oxford University, 2010.

[16] T. Klapiscak and R.H. Bordini, JASDL: A practical programming approach combining agent and semantic web technologies, in: *6th International Workshop on Declarative Agent Languages and Technologies*, Vol. 5397, 2008, pp. 91–110. doi:10.1007/978-3-540-93920-7_7.

[17] C.-S. Lee, M.-H. Wang and J.-J. Chen, Ontology-based intelligent decision support agent for CMMI project monitoring and control, *International Journal of Approximate Reasoning* **48**(1) (2008), 62–76. doi:10.1016/j.ijar.2007.06.007.

[18] Y. Ma, Y. Feng, B. Jin and J. Wei, A default extension to distributed description logics, *Web Intelligence and Agent Systems: An International Journal* **4**(4) (2006), 371–383.

[19] V. Mascardi, D. Ancona, M. Barbieri, R.H. Bordini and A. Ricci, CooL-AgentSpeak: Endowing AgentSpeak-DL agents with plan exchange and ontology services, *Web Intelligence and Agent Systems* **12**(1) (2014), 83–107. doi:10.3233/WIA-140287.

[20] Á.F. Moreira, R. Vieira, R.H. Bordini and J.F. Hübner, Agent-oriented programming with underlying ontological reasoning, in: *3rd International Workshop on Declarative Agent Languages and Technologies*, Vol. 3904, 2005, pp. 155–170. doi:10.1007/11691792_10.

[21] A.R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira and R.H. Bordini, Arguing about task reallocation using ontological information in multi-agent systems, in: *12th International Workshop on Argumentation in Multiagent Systems*, 2015.

[22] E.C. Paraiso and J.-P.A. Barthès, An intelligent speech interface for personal assistants applied to knowledge management, *Web Intelligence and Agent Systems: An International Journal* **3**(4) (2005), 217–230. doi:10.1109/CSCWD.2005.194288.

[23] E.C. Paraiso and A. Malucelli, Ontologies supporting intelligent agent-based assistance, *Computing and Informatics* **30** (2011), 829–855.

[24] A. Ricci, M. Piunti and M. Viroli, Environment programming in multi-agent systems: An artifact-based perspective, *Autonomous Agents and Multi-Agent Systems* **23**(2) (2011), 158–192. doi:10.1007/s10458-010-9140-7.

[25] D. Schmidt, A.R. Panisson, A. Freitas, R.H. Bordini, F. Meneguzzi and R. Vieira, An ontology-based mobile application for task managing in collaborative groups, in: *29th International Florida Artificial Intelligence Research Society Conference*, 2016, pp. 522–525.

[26] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, Pellet: A practical OWL-DL reasoner, *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(2) (2007), 51–53. doi:10.1016/j.websem.2007.03.004.

[27] X. Tao, Y. Li and N. Zhong, A personalized ontology model for web information gathering, *IEEE Transactions on Knowledge and Data Engineering* **23**(4) (2011), 496–511. doi:10.1109/TKDE.2010.145.

[28] Q.-N.N. Tran and G. Low, MOBMAS: A methodology for ontology-based multi-agent systems development, *Information and Software Technology* **50**(7–8) (2008), 697–722. doi:10.1016/j.infsof.2007.07.005.

[29] A. Vorobiev and N. Bekmamedova, An ontology-driven approach applied to information security, *Journal of Research and Practice in Information Technology* **42**(1) (2010), 61–76.

[30] L.K. Wickramasinghe and L.D. Alahakoon, Dynamic self organizing maps for discovery and sharing of knowledge in multi agent systems, *Web Intelligence and Agent Systems: An International Journal* **3**(1) (2005), 31–47.