Received: June 25, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

BOEM: A Model for Automating Detection and Evolution of Distributed Ontologies in Multi-Agent Environment

Ashraf Soliman[1*] Akram Salah[2] Hesham Hefny[1]

[1]*Institute of Statistical Studies and Research, Department of Computer and Information Sciences, Cairo University, Egypt*
[2]*Faculty of Computers and Information, Department of Computer Sciences, Cairo University, Egypt*
* Corresponding author's Email: ashrafcs2007@gmail.com

Abstract: Knowledge gives a strong support to autonomous agents in multi-agent systems and thus the evolution of agent's knowledge needs a great attention since it has a control on agents' behaviors and has effect on their decisions making. The problem is to allow agents to detect and decide whether they need more domain knowledge and allow their knowledge to evolve consistently and automatically. This paper utilizes ontologies to represent the internal knowledge of agents instead of utilizing them only as a shared conceptualization. Consequently, the paper proposes a model of bottom-up instance-driven ontology evolution that allows the internal ontologies of agents to evolve automatically and consistently in run time based on agents' interactions. Experiments are designed and implemented to evaluate our model in different situations. One of its results shows that an empty internal ontology of one agent could evolve automatically in runtime by 88.3% through its interactions with other agents. Moreover, a comparison between the proposed approach and literature review approaches is presented to compare between their different features and techniques. This paper is considered a step forward to automate ontology evolution for agents in multi-agent environment.

Keywords: Knowledge evolution, Multi-agent system, Knowledge distribution, JADE, FIPA, Protocol, JSON, JAKSON.

1. Introduction

Ontologies are designed to provide a formal and explicit representation of knowledge of a domain in terms of concepts (or classes), relations between these concepts and instances of these concepts [6]. The importance of ontologies is increasing especially in multi-agent environments since they enable agents to communicate, interact, and understand each other [2, 6]. Moreover, they provide agents with intelligence, reasoning, and support the main four characteristics of agents which are autonomy, social ability, reactivity, and proactiveness [24].

These ontologies need to evolve over time to reflect changes in the domain. Ontology evolution means modifying or upgrading the ontology when there is a certain need for change or there comes a change in the domain knowledge [1]. The current ontology evolution techniques such as in [5-10, 19-23] have several hidden weaknesses such as automating ontology evolution and resolving inconsistencies after applying new changes. The automation is important because human intervention is time consuming and error prone.

This paper proposes a bottom-up instance-driven model for automating both change detection and ontology evolution of agent's internal knowledge represented through agents' interactions in runtime. The paper sets a number of experiments to evaluate the proposed model in different cases. One of its results shows that the internal ontology of an agent has evolved *automatically* and *consistently* in *runtime* from scratch to 88.3%. The rest of paper is organized as follows. Section 2 presents other research efforts related to our work. Section 3 explains the proposed ontology evolution model. Section 4 presents the implementation of our model. Section 5 evaluates our model through a number of experiments and a

comparison with related work is presented. Finally, section 6 is for conclusion and future work.

2. Related work

There are some efforts exploit multi-agent systems as a mean to facilitate the process of ontology evolution. For example, Zhang et al. in [20] uses autonomous agents and Pi-Calculus to model changes in ontologies. Autonomous agents are used to represent ontology entities and Pi-Calculus is used to describe and formalize agent actions and information exchanging between agents. Sellami et al. in [9] develop DYNAMO-MAS, an interactive tool based on an adaptive multi-agent system (AMAS). It aims to build and evolve ontologies from text. It only suggests enriching the initial ontology with new concepts, terms, or relations without any change in its content. Then it is up to ontologists to manually modify this content. Benomrane et al. in [19] present an ontologist feedback tool, called OntoAMAS as an extension of [9]. This extension allows ontologists to modify or add new concepts or terms to the initial ontology. Then the AMAS self-organizes and produces an updated ontology with new proposals which can be modified by ontologists again. Thus, it works in an interactive and iterative way until a satisfactory state of the ontology is achieved.

Other researchers have realized the important role of instances and the bottom-up approach in ontology evolution either for automating the detection of changes or for update instances descriptions in a new version of an ontology. For example, in [21], principles, rules, and algorithms are presented to reexamine and suggest instance descriptions to ontologists after migrating instances to the latest version of an ontology. Also, it provides flexible interactive techniques for updating the available descriptions after ontologists accept or reject such recommendations. Xie et al. in [10] exploit instances to automate changes detection in an ontology and provide ontologists with proper recommendations for ontology evolution. This detection method is based on analyzing new extracted instances from related databases with current instances in the ontology. This analysis enables the system to detect changes in conceptualization and recommend ontologists with merge concepts, split concept, or extract super concept. Santoso et al. in [23] provide a bottom-up approach for change detection. It uses the difference between the ontology metadata and the related database metadata as a trigger for change detection. So, its detection algorithm is based on detecting new components in related databases such as classes, instances, properties, or axioms to be added into the current ontology. This approach focuses on automating changes detection but does not present a solution for ontology evolution.

Another kind of efforts attempts to support ontology engineers on maintaining consistency after ontology evolution. For example, Touhami et al. in [6] propose an ontology evolution activity that assists ontologists for carrying out the ontology changes. It lists all possible changes to ontologists, identifies all consistency constraints to be checked after applying each change. Then a kit of additional changes associated with each change is applied automatically to keep the consistency state if the required change from ontologists violates one or more of the predefined consistency constraints. These additional changes work in iterative way as long as one of applied changes either additional or required leads to inconsistency state. Javed et al. in [5] present a layered change operator framework for ontology evolution that allows ontology engineers to deal with generic changes at level one and level two and other users (such as domain experts, content managers) to deal with domain-specific changes at level three. It also presents a layered change log model that works in line with the given layered change operator.

There are other approaches depend on various sources as a background knowledge towards automating change detection for ontology evolution. For example, Maree and Belkhatir in [22] propose an automatic framework for enriching ontologies depending on the Web as background knowledge. It combines semantic relatedness measures, automatic pattern acquisition techniques, named entity extraction algorithm (GATE), and NLP techniques to extract missing knowledge from the Web. Zablith et al. in [7, 8] present an ontology evolution framework, called Evolva, that uses structured and unstructured sources as background knowledge to reduce or even eliminate user involvement in exploring new concepts to add to an ontology. Additionally, it uses external sources such as WordNet and Semantic Web ontologies to discover relations between new concepts and others already exist in the ontology. Although most of these tasks are performed automatically, it needs ontologists' approval and waits their selection of applying these changes in the base ontology or in a new version.

Most of these approaches attempt to automate some steps of ontology evolution but it needs ontologists either to trigger the evolution process or to give approval about evolution results. Our approach is an attempt for automating both change detection and ontology evolution in run time. It is based on the main agents' characteristics of autonomy, social ability, reactiveness, and

proactiveness to detect and apply that evolution. Moreover, it depends on instances and background knowledge to automate change detection.

3. The proposed model: bottom-up ontology evolution model (BOEM)

This model is proposed to enable ontologies in multi-agent environment of a specific domain to evolve *automatically* and seamlessly without intervention of ontologists. The basic idea of this proposed model stands upon considering ontologies as a formal representation of agent's internal knowledge. Furthermore, agents completely depend on their internal ontologies to interact with their environment. At the same time these internal ontologies have a chance to evolve *automatically* over time through their agents' interactions. As shown in Fig. 1, the model consists of three aspects: (1) the knowledge distribution model, (2) the detection process, and (3) the evolution process. All of these three aspects of the model are explained in detail in the following sections.

3.1 Knowledge distribution model

This section discusses our proposed model of knowledge distribution in multi-agent environment that has previously published in [4]. As shown in Fig. 1, the knowledge distribution model is the basic component of our proposed bottom-up ontology evolution model. It proposes a domain as a set of agents and each one of these agents has its own internal ontology that formally represents agent's knowledge. Consequently, the domain's knowledge is distributed on agents' internal ontologies. As shown in Fig. 2, some types of these agents have deeper knowledge about the domain and thus these types of agents are considered *sources* of domain's knowledge. Other agents in the domain depend on these types of agents to gain more necessary knowledge and to overcome difficulties in their interactions. So, the model calls these types of agents that have deeper domain knowledge *knowledge-source* agents and also calls the other types of agents *individual* agents.

When there is more than one *knowledge-source* agent in a domain, the knowledge of each one of these agents will represent a specific aspect of the domain. Therefore, the knowledge K_{Sm} of a *knowledge-source* agent S_m cannot intersect with the knowledge K_{Sn} of another *knowledge-source* agent S_n. Thus the relationship between *knowledge-source* agents can be represented as follows:

$$K_{Sm} \cap K_{Sn} = \emptyset \quad (1)$$

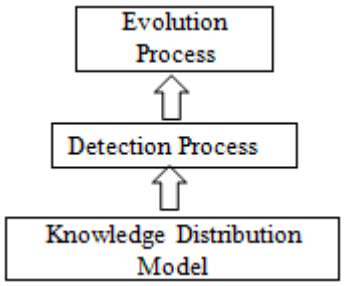

Figure.1 Aspects of the bottom-up ontology evolution model

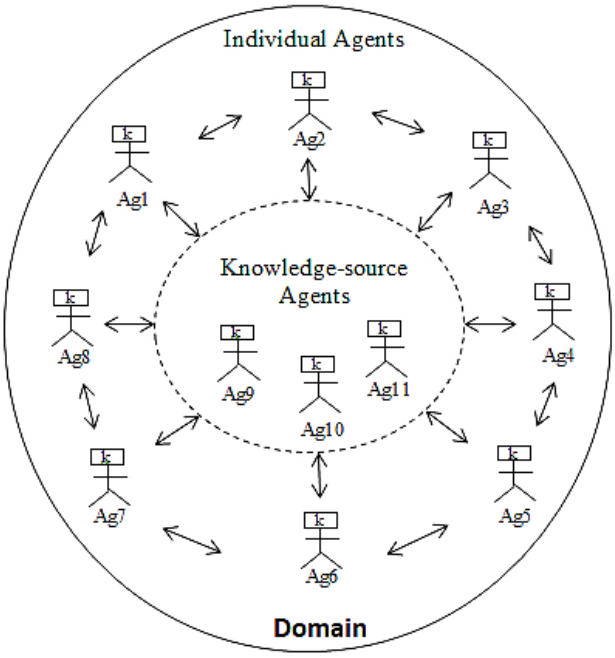

Figure.2 The dependency relationship between agents of a domain

So the question is how we can reach to the entire knowledge of a domain. The answer that is proposed by this paper is that the entire domain's knowledge can be reached by gathering the knowledge of each *knowledge-source* agent in the domain. So, a domain's knowledge K_D is union of all *knowledge-source* agents' knowledge K_{Si}. This answer can be represented as follows:

$$K_D = \bigcup K_{Si} \quad (2)$$

With regard to *individual* agents, an *individual* agent has knowledge that is derived from *knowledge-source* agents. Unlike *knowledge-source* agents, *individual* agents' knowledge may be intersected or equal. Notice that the knowledge of *individual* agents is unlikely to be conflicted since all of them are derived from specific knowledge sources of the domain, i.e., the *knowledge-source* agents'

knowledge. So, the relationship between *individual* agents and *knowledge-source* agents can be represented as follows:

$$K_I \subseteq K_S \quad (3)$$

Where K_I is knowledge of an *individual* agent and K_S is knowledge of a *knowledge-source* agent.

Moreover, as shown in Fig. 3, the model proposes that the entire knowledge of a domain is implicitly shared as a virtual ontology. This virtual ontology is explicitly represented by and distributed on several sub ontologies owned by the *knowledge-source* agents. Therefore, an *individual* agent interacts only with *knowledge-source* agents to get more and necessary domain knowledge that enable it to overcome difficulties while it interacts with another *individual* agent.

3.2 Detection Process

The detection process is mainly based on the proposed knowledge distribution model in section 3.1, as well as, it focuses only on detecting the need of *individual* agents' internal ontologies to evolve.

This process starts when the knowledge of an *individual* agent *B*, which is represented by its internal ontology O_B, cannot answer a query *Q* from another *individual* agent *A* about an instance in the domain. Therefore, the *individual* agent *B* and *automatically* sends a detection query Q_D to all *knowledge-source* agents. This detection query Q_D is a request for checking the existence of a given instance. The detection process ends with one of the following possibilities according to the reply of the *knowledge-source* agents:

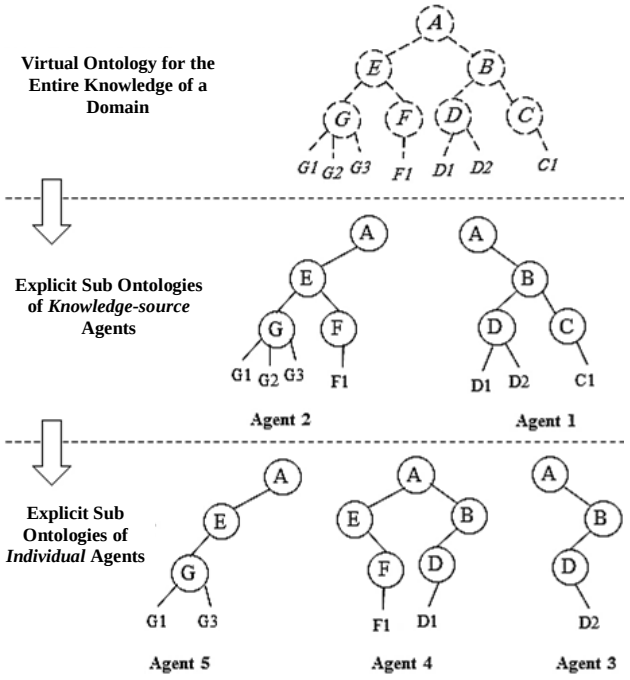

Figure.3 Domain's knowledge as a virtual ontology distributed on agents' internal ontologies

1) If the *individual* agent *B* does not receive a reply with confirm message to its detection query Q_D from any *knowledge-source* agents in the domain, it means that the instance is not known in the domain at all. So, the *individual* agent *B* will reply the *individual* agent *A* with *disconfirm* message and detects that its internal ontology O_B does not have a need to evolve.
2) If the *individual* agent *B* receives a reply with *confirm* message to its detection query Q_D from at least one *knowledge-source* agent in the domain, it means that the instance is known in the domain. So, the *individual* agent *B* will reply the *individual* agent *A* with *confirm* message and detects that its internal ontology O_B has a need to evolve.

3.3 Evolution Process

When the detection process of an *individual* agent ends with detecting that there is missing knowledge (i.e., missing instance) in its internal ontology, the *individual* agent starts the evolution process *immediately* and *automatically* by opening an *evolution channel* with the *knowledge-source* agent that has the missing instance. The *evolution channel* is opened by sending the proposed evolution query Q_E as a request to the *knowledge-source* agent from the *individual* agent to get not only the missing instance but also to get its class hierarchy.

So, the *individual* agent sends the first evolution query Q_{E1} to the *knowledge-source* agent to request the direct class of the new instance. If the received instance's class is not in the *individual* agent's internal ontology, the *individual* agent will send the second evolution query Q_{E2} to the *knowledge-source* agent to request the first direct superclass of the received instance's class. If the received superclass also is not in its internal ontology, the *individual* agent will send the third evolution query Q_{E3} to request the second direct superclass of the received instance's class. Thus, the *individual* agent will send sequence of evolution queries Q_{En} until it finds one of instance's superclasses in its internal ontology or gets all superclasses of the received instance's class. Therefore, the evolution process finishes when the new instance and all its class hierarchy are populated in the internal ontology of the *individual* agent.

As explained, the proposed evolution model adopts and applies the evolution bottom-up strategy for ontology evolution. This strategy starts firstly with populating missing instances in the internal ontologies of *individual* agents then the evolution moves up from populating the instances' direct classes to populate all superclasses of instances' class hierarchy. By this way, the proposed evolution model enables the internal ontology of an *individual* agent to evolve by populating any new instances from *knowledge-source* agents with their important semantics. This result keeps the evolution of the internal ontologies of *individual* agents away from inconsistencies and conflicts. Another significant result is that the proposed evolution model provides an *automatic* and seamless ontology evolution of *individual* agents' internal ontologies.

4. Implementation

The proposed bottom-up ontology evolution model is implemented with JADE [14] as a multi-agent environment that provides all requirements for agent management. It also enables agents to interact through ACL messages based on FIPA standards [15, 16]. The model also implements internal ontologies of agents by OWL-DL ontology language [18] and using Jena Library as ontology API [17]. The proposed detection query Q_D and evolution query Q_E are also implemented based on object serialization. Therefore as shown in Fig. 4, a Java class called *IsInstance* is created to make a detection query object; another two Java classes called *ClassOf* and *SuperClassOf* are created to make the two types of evolution query objects.

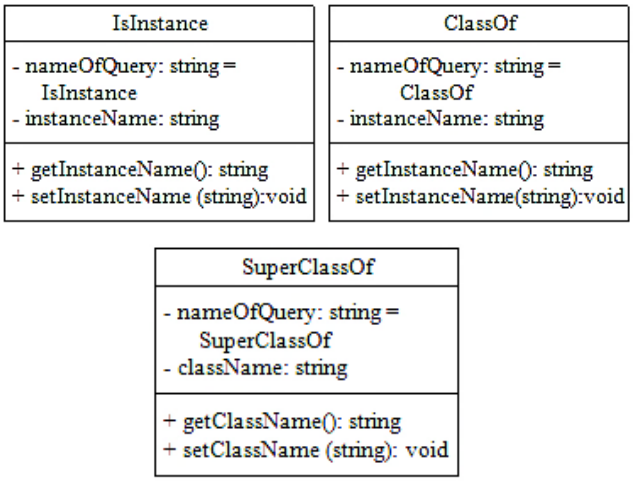

Figure.4 Classes of detection query and evolution query objects

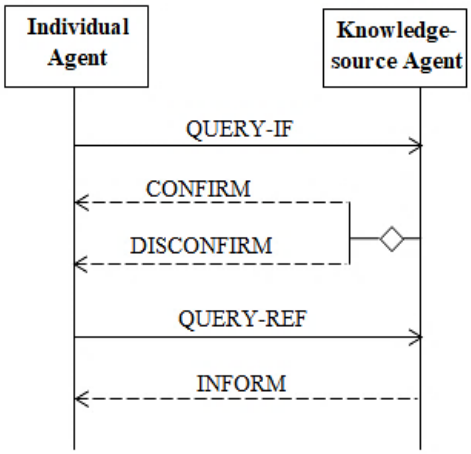

Figure.5 The protocol of detection and evolution process based on FIPA standards

All these query objects are serialized into JSON [12] string format and deserialized into their proper objects with support of JSON API called JAKSON [13]. The serialized object of a detection query Q_D is serialized as the following example:

{"nameOfQuery": "IsInstance", "instanceName": "Book_1"}

The serialized objects of an evolution query Q_E are serialized as the following example:

{"nameOfQuery": "ClassOf", "instanceName": "Book_1"}

{"nameOfQuery": "SuperClassOf", "className": "Book"}

As shown in Fig. 5, the detection query serialized object is sent in ACL message with QUERY-IF communicative act and the reply will be with ACL message with CONFIRM or DISCONFIRM communicative act. The evolution query object is sent in ACL message with QUERY-REF communicative act and the reply will be with ACL message INFORM communicative act contains either the class name of the new instance or the superclass name of the instance's class.

5. Evaluation

5.1 Experiments

This section is for evaluating the proposed evolution model by using public ontologies as *source* ontologies for *knowledge-source* agents. A lightweight public ontology is chosen from Protégé ontology library [25] for our experiment. This ontology is Library Management System Ontology

(LMSO) that includes all major class-subclass hierarchy, properties, characteristics, restrictions and instances of a Library Management System of any Educational Organization [11]. The LMSO describes the library domain based on four main classes, *LibraryMember*, *LibraryPersonnel*, *LibraryResource*, and *LibraryService*. Each one of these classes has its own class-subclass hierarchy with its instances. The whole class-subclass hierarchy of LMSO consists of 32 classes, 60 instances, 28 subclass relations, 18 object properties, and 1 data property (139 entities).

The experiment consists of three agents, Clerk agent, Librarian agent, and Visitor agent. The role of each agent is shown as follows:

- **The Librarian agent** plays the role of the *knowledge-source* agent that has all knowledge about the domain. Therefore the public ontology LMOS is assigned to it as its internal ontology. Also it is responsible for replying to all queries from the Clerk agent about the domain.
- **The Clerk agent** is an *individual* agent and it is responsible for answering queries from the Visitor agent. The experiment stands up specifically to evaluate the evolution of the Clerk agent's internal ontology through its interactions with Visitor and Librarian agents.
- **The Visitor agent** is an *individual* agent that plays the role of library's visitors. It is designed to test the evolution process of the Clerk agent's ontology by sending queries to the Clerk agent that would trigger the evolution process of its internal ontology.

As shown in Table 1, there are eight experiments to evaluate the evolution of the Clerk agent's ontology. The Clerk and the Visitor agent are mainly the interacting agents but the Clerk agent may need to consult the Librarian agent. For this case, Table 1 in the column of interacting agents shows whether Librarian agent is involved as consultant to the Clerk agent in the experiment or not. The Results column in Table 1 states that whether the settings of the experiment trigger the evolution process of the Clerk agent's ontology or not. In addition, it shows the status of Clerk agent's ontology after the experiment, evolved or not and why.

In Table 1, it is observed that the Clerk ontology only evolved in two experiments, number 1 and 3. This evolution is restricted in both experiments since the queried instances do not exist in the Clerk ontology but they already exist in the Librarian ontology. It is also observed that the Librarian agent is involved in all the Clerk agent interactions, as *knowledge-source* agent, except in three experiments number 5, 6, and 8. This is because, in these three experiments, the queried instances already exist in the Clerk ontology so the Clerk agent does not need to consult and involve the Librarian agent.

As shown in Table 2, the empty ontology of the Clerk agent evolved after experiment No.1 and became containing a class-subclass hierarchy of 25 classes, 60 instances, and 21 subclass relations (106 entities). This means that the empty Clerk ontology became containing 88.3 % of the source ontology LMSO. Note that all instances and their class-subclass hierarchy are added except 7 classes because they do not have any instances. Thus this result explains why our approach is called instance driven approach. In experiment No. 3, the Clerk ontology is not empty this time but has 41 entities before starting the interaction, then after interaction the number of its entities became 106 entities with evolution percentage 54.2%.

The evolution percentages of the Clerk ontology in both experiments number 1 and 3 in Table 2 is computed according to the proposed ontology evolution measurement. This proposed evolution measurement is shown in the following equation:

$$\text{Evolution} = (|(E_I)_{After}| - |(E_I)_{Before}|) / |E_S| \quad (4)$$

Where E_I is the entities set of the *individual* agent's ontology, Clerk ontology. E_S is the entities set of the *knowledge-source* agent's ontology, Librarian ontology. $|(E_I)_{After}|$ is the count of the entities set E_I after interaction. $|(E_I)_{Before}|$ is the count of the entities set E_I before interaction. $|E_S|$ is the count of the entities set E_S.

Since the paper focuses on the *automatic* evolution based on instances and their class-subclass hierarchy, the basis of the proposed evolution measurement is based on the class-subclass hierarchy of LMSO that consists of 32 classes, 60 instances and 28 subclass relations (120 entities). So, the evolution of experiment No. 1 is computed based on Eq. (4) as follows:

$$\text{Evolution} = (106 - 0) / 120 \times 100 = 88.3\ \%$$

Similarly, the evolution of experiment No. 3 is computed based on Eq. (4) as follows:

$$\text{Evolution} = (106 - 41) / 120 \times 100 = 54.2\ \%$$

According to additional experiments, it has found that upper ontologies such as SUMO are not relevant to evaluate our model. In addition, public ontologies that use classes instead of instances to represent individuals in a domain are also not relevant. Both types of ontologies are irrelevant because our

Table 1. Evaluation Experiments

Experiment No.	Description	Clerk Ontology Status Before Experiment	Interacting Agents	Results (Clerk Ontology Status After Experiment)
1	The Visitor agent asks the Clerk agent about **all instances** in the ontology of the Librarian agent.	The Clerk agent **has no knowledge**; it has an empty ontology.	Main: Visitor Agent Clerk Agent Involved: Librarian Agent	**Evolved** The Clerk ontology evolved by all instances and their class hierarchy.
2	The Visitor agent asks the Clerk agent about **an instance does not exist** in the ontology of the Librarian agent.		Main: Visitor Agent Clerk Agent Involved: Librarian Agent	No evolution Since the instance is unknown in the Librarian ontology and so in the domain.
3	The Visitor agent asks the Clerk agent about **all instances** in the ontology of the Librarian agent.	The Clerk agent has some knowledge. It has ontology contains only the *LibraryMember* class-subclass hierarchy. That includes 41 entities (8 classes, 26 instances, 7 subclass relations)	Main: Visitor Agent Clerk Agent Involved: Librarian Agent	**Evolved** The Clerk ontology evolved by all its missing instances and their class hierarchy.
4	The Visitor agent asks the Clerk agent about **an instance does not exist** in the ontology of the Librarian agent.		Main: Visitor Agent Clerk Agent Involved: Librarian Agent	No evolution Since the instance is unknown in the Librarian ontology and so in the domain.
5	The Visitor agent asks the Clerk agent about **instances known** in Clerk's ontology.		Main: Visitor Agent Clerk Agent **Not** Involved: Librarian Agent	No evolution Since the queried instances already exist in the Clerk ontology.
6	The Visitor agent asks the Clerk agent about **all instances** in the ontology of the Librarian agent.	The Clerk agent **has the same ontology** of the Librarian agent.	Main: Visitor Agent Clerk Agent **Not** Involved: Librarian Agent	No evolution Since the queried instances already exist in the Clerk ontology.
7	The Visitor agent asks the Clerk agent about **an instance does not exist** in the ontology of the Librarian agent.		Main: Visitor Agent Clerk Agent Involved: Librarian Agent	No evolution Since the instance is unknown in the Librarian ontology and so in the domain.
8	The Visitor agent asks the Clerk agent about **instances known** in Clerk's ontology.		Main: Visitor Agent Clerk Agent **Not** Involved: Librarian Agent	No evolution Since the queried instances already exist in the Clerk ontology.

Table 2. The evolution results

Experiment No.	Count of entities in LMSO	Count of entities in Clerk ontology								Evolution Results
		Before				After				
		Class	instance	subclass relation	Total	Class	instance	subclass relation	Total	
1	120	0	0	0	0	25	60	21	106	88.3%
3	120	8	26	7	41	25	60	21	106	54.2%

automatic evolution model is mainly driven by instances and these types missed them. These findings are not considered limitations but they violate requirements of our model.

5.1 Comparative analysis

In this section, Table 3 summarizes and shows a comparison of our proposed approach with the previous presented approaches for ontologies evolution. There are three categories of comparison criteria. The first category is evolution type that has the following criteria:

- **Automatic:** It determines whether in the given approach ontology evolution is performed without intervention of ontologists.
- **Runtime:** It means that ontologies evolve while they are used for communication between agents or for replying queries to users. Otherwise, the evolution is considered at design time.

The second category is evolution techniques which has the following criteria:

- **Instance:** it determines whether the approach depends on instances to perform an evolution.
- **Agent:** it determines whether the approach depends on multi-agent systems to perform an evolution.
- **Background knowledge:** it determines whether the approach depends on background knowledge such as ontologies, lexicons, or databases to perform an evolution.
- **Others:** it means that there are other techniques different than the techniques of instance, multi-agent systems, and background knowledge the given approach depends on it to perform an evolution.

The third category is **consistency** that checks whether the given approach maintain the consistency after evolution is performed.

Table 3. Comparison between evolution approaches and our approach

Approaches	Evolution Type		Evolution Techniques				Consistency
	Automatic	Runtime	Instance	Agent	Background Knowledge	Others	
Our Approach	√	√	√	√	√		√
Zhang et al. [20]	√			√			
Sellami et al. [12]				√			
Benomrane et al. [19]	semi-automatic			√			√
Tzitzikas et al. [21]	semi-automatic		√		√		√
Touhami et al. [6]	semi-automatic					√	√
Javed et al. [5]						√	√
Maree and Belkhatir [22]	√					√	
Xie et al. [10]	semi-automatic					√	√
Zablith et al. [7, 8]	semi-automatic				√		√

The main advantage of our approach is that it attempts to be closer from how humans' knowledge evolves and how the need to evolve their knowledge is detected. To achieve that goal, our approach combines between automatic and runtime for both change detection and ontology evolution with keeping ontologies in consistent state after evolution. On the one hand, it depends on interactions between *individual* agents about instances to detect its need to evolution. On the other hand, it depends on the interaction between *individual* agents and *knowledge-source* agents, i.e., background knowledge, to perform the desired evolution. Although the proposed approach is presented to allow internal ontologies of agents to evolve, it also can be used to allow distributed ontologies to evolve by agents.

6. Conclusion

This paper focuses on how to enable agent's knowledge represented by an ontology to evolve *automatically* and *consistently* without user intervention in *runtime*. So, it proposes a model of bottom-up instance-driven ontology evolution to allow an agent's ontology to evolve in run time through agent's interactions with other agents. This evolution model consists of three phases: (1) knowledge distribution model, (2) detection process, and (3) evolution process. The first phase is the knowledge distribution model that defines two types of agents; one of them is *knowledge-source* agents that are considered the *source* of domain knowledge. The second type is *individual* agents that interact with *knowledge-source* agents to obtain its missing knowledge. The second phase is the detection process which is triggered *automatically* through an interaction between *individual* agents. Specifically, when an *individual* agent receives a query about an *undefined* instance in its ontology but it is already defined in one of *knowledge-source* agents' ontologies. The third and last phase is the evolution process which starts *automatically* after detection process. In this phase, an *individual* agent opens an *evolution channel* with the *knowledge-source* agent that has the missing knowledge. Through this channel, the *individual* agent's ontology evolves and its knowledge upgrades by adding the *undefined* instances and their class hierarchies to its internal ontology. Experiments are designed and implemented to evaluate our approach using a light-weight public ontology. One of its results shows the ability of our model to allow the internal ontology of an *individual* agent to evolve with 88.3% from scratch. Moreover, a comparison with other evolution approaches is shown in Table 3.

The limitation of our model is that it depends only on the class-subclass relations and instance relations to describe the semantics of instances. So, our model can be enhanced by enabling *individual* ontologies to evolve with more semantics about instances such as involving axioms and properties to these semantics. Also in the future we plan to extend our model with an ontology maintenance process to keep semantics of instances in *individual* ontologies up to date with any changes related to them in the *source* ontologies.

References

[1] A. M. Khattak, R. Batool, Z. Pervez, A. M. Khan, and S. Lee, "Ontology Evolution and Challenges", *Journal of Information Science and Engineering*, Vol.29, pp.851-871, 2013.

[2] F. P. Pai, I. C. Hsu, and Y. C. Chung, "Semantic Web Technology for Agent Interoperability: A Proposed Infrastructure", *Applied Intelligence*, Vol.44, No.1, pp.1–16, 2016.

[3] G. Weichhart and Y. Naudet, "Ontology of Enterprise Interoperability Extended for Complex Adaptive Systems", *Lecture Notes in Computer Science*, Vol. 8842, Springer, Berlin, Heidelberg, 2014.

[4] A. Soliman, A. Salah, and H. Hefny, "Modeling Distribution and Exchanging Domain Knowledge in Multi-agent Environment", *International Journal of Computer Sciences and Computer Security*, Vol.15, No.4, pp.1-13, 2017.

[5] M. Javed, Y. M. Abgaz, and C. J. Pahl, "Ontology Change Management and Identification of Change Patterns", *Journal on Data Semantics*, Vol.2, No.2, pp.119–143, 2013.

[6] R. Touhami, P. Buche, J. Dibie, and L. Ibanescu, "Ontology Evolution for Experimental Data in Food", *Communications in Computer and Information Science*, Vol.544, pp.393-404, 2015.

[7] F. Zablith, M. Sabou, M. d'Aquin, and E. Motta, "Ontology Evolution with Evolva", In: *Proc. of International Conf. on The Semantic Web: Research and Applications, Springer, Berlin, Heidelberg*, pp.908–912, 2009.

[8] F. Zablith, M. Sabou, M. d'Aquin, and E. Motta, "Using background knowledge for ontology evolution", In: *Proc. of International Conf. On Ontology Dynamics, Karlsruhe, Germany*, 2008.

[9] Z. Sellami, V. Camps, and N. Aussenac-Gilles, "DYNAMO-MAS: A Multi-Agent System for Ontology Evolution from Text", *Journal on Data Semantics*, Vol.2, No.2, pp.145–161, 2013.

[10] C. Xie, L. Jiang, and H. Cai, "Instance-Driven Ontology Evolution Mechanism towards Enterprise Data Management", In: *Proc. of International Conf. On e-Business Engineering, Beijing, China*, pp. 24-30, 2011.
[11] A. Banu, S. S. Fatima, and K. U. R. Khan, "Building OWL Ontology: LMSO-Library Management System Ontology", *Advances in Computing and Information Technology*, Vol. 178, Springer, Berlin, Heidelberg, 2013.
[12] JavaScript Object Notation (JSON) website. [Online]. http://www.json.org/. Accessed: November 25, 2016.
[13] Jackson JSON API website. [Online]. Available: https://github.com/FasterXML/jackson-docs/wiki/JacksonHome. Accessed: November 25, 2016
[14] Java Agent Development Framework (JADE) Website. [Online]. Available: http://jade.tilab.com. Accessed: November 25, 2016.
[15] FIPA Query Interaction Protocol. FIPA Std. SC00027, 03/12/2002, Document No. SC00027H. [Online]. Available: http://www.fipa.org/specs/fipa00027/SC00027H.pdf. Accessed: November 25, 2016.
[16] FIPA communicative act library specification. FIPA Std. SC00037, 03/12/2002, Document No. SC00037J. [Online]. Available: http://www.fipa.org/specs/fipa00037/. Accessed: November 25, 2016.
[17] Apache Jena Website. [Online]: https://jena.apache.org/.
[18] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax", *Technical report, W3C, W3C Recommendation*, 2004.
[19] S. Benomrane, Z. Sellami, M. B. Ayed, "An ontologist feedback driven ontology evolution with an adaptive multi-agent system", *Advanced Engineering Informatics*, Vol.30, No.3, pp.337-353, 2016.
[20] R. Zhang, D. Guo, W. Gao, L. Liu, "Modeling ontology evolution via Pi-Calculus", Information Sciences, Vol.346, No. C, pp.286-301, 2016.
[21] Y. Tzitzikas, M. Kampouraki, A. Analyti, "Curating the Specificity of Ontological Descriptions under Ontology Evolution", *Journal on Data Semantics*, Vol.3, No.2, pp.75-106, 2014.
[22] M. Maree and M. Belkhatir, "Coupling Semantic and Statistical Techniques for Dynamically Enriching Web Ontologies", *Journal of Intelligent Information Systems*, Vol.40, No.3, pp.455-478, 2013.
[23] H. Santoso, S. Haw, and C. Lee, "Change Detection in Ontology Versioning: A Bottom-Up Approach by Incorporating Ontology Metadata Vocabulary", In: *Proc. of International Conf. On Database Theory and Application, Bio-Science and Bio-Technology*, pp.37-46, 2010.
[24] G. Beydoun and G. Low, "Centering Ontologies in Agent Oriented Software Engineering Processes", *Journal of Complex and Intelligent Systems*, Vol.2, No.3, pp.235–242, 2016.
[25] Protégé Ontology Library website. [Online]. Available: https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library.