An Ontologist Feedback Driven Ontology Evolution with an Adaptive Multi-agent System

Souad Benomrane [1, *], Zied Sellami [2], Mounir Ben Ayed [1]

[1] REGIM-Lab.: REsearch Groups in Intelligent Machines, University of Sfax, ENIS, BP 1173, Sfax 3038, Tunisia
[2] LIP6 – Equipe ACASA, 4 Place Jussieu, Boite Courrier 169, 75252 Paris Cedex 05, France
* Corresponding Author. E-mail: souad.benomrane@ieee.org

Abstract. In a dynamic environment, it is necessary to make changes to an ontology according to new knowledge and user needs. However, ontology evolution is still a complex and time-consuming task. In this paper, we presented OntoAMAS, an ontologist feedback tool based on an adaptive multi-agent system (AMAS) for ontology evolution. It consists of two components: (i) an AMAS (concept and term agents) that represents the current state of an ontology and (ii) a graphical interface which allows to manage the different interactions between the ontologist and AMAS proposals. First, we defined an adaptive behavior that enables agents to react to the ontologist's feedback. The ontologist gives his/her feedback (elementary and composite changes). He/She can also add new terms and concepts. Then, the AMAS self-organizes and produces an updated ontology with new proposals. It works in an interactive and iterative way until a satisfactory state of the ontology is achieved. Second, we proved that OntoAMAS guarantees that the adaptive skills we added to agents allow them to detect the uselessness of some proposals so as to avoid them together with the wrong ones and to propose others. The experimental results show the relevance of OntoAMAS and the effectiveness in time performance of its Protégé (ontology editor) GUI.

Keywords: Adaptive multi-agent system; Ontology evolution; Ontologist feedback.

1. Introduction

Ontologies developed and used for various applications have been increasing over the years. The major issue faced in ontologies is their change or evolution. Ontology evolution seeks to keep an ontology up to date with the changes of the domain it models, to new knowledge and user needs. These changes should be implemented in the ontology and should be managed in such a way that ensures to safeguard its consistency, structure and continuity across different versions of the ontology. Ontology evolution is a process that involves a number of steps, for which many techniques, tools and approaches might be needed [1]. In the literature, the majority of approaches tried to automatically perform one or two steps of the process to simplify the user's (ontologist's) task. However, throughout the evolution process stages, the ontologist was immediately involved. The ontologist chooses to apply the changes in the ontology, places the new entities (concepts, terms, relations), etc. However, manually evolving an ontology is a costly, complex, and time consuming process.

To the best of the authors knowledge, there are only two systems proposing an automatic ontology evolution namely EVOLVA and DYNAMO. The first [2] is only efficient for the evolution of English ontologies and has difficulties when the concerned domain is very specific. The second system was proposed by Ottens [3] as a first prototype of DYNAMO and then a new tool, called ≪ DYNAMO-MAS ≫ [4], was developed. DYNAMO prototype is based on an adaptive multi-agent system (AMAS) to co-construct and evolve ontology from texts. However, experiments [4] showed that linguistic clues are insufficient and claim that for the work to be more effective the intervention of an ontologist is required. An ontologist is a cognitive engineer using expert interviews and information from texts to construct ontologies.

In conclusion, ontology evolution needs to be more automated without ignoring the role of the ontologist especially during the validation step [5] and [6]. In this paper we aimed to answer two main

questions: (i) How to improve the quality of evolution proposals (changes) by referring to the ontologist's feedback? (ii) How exploiting of the ontologist's feedback may reduce his frequent involvement? Therefore, our objective was to improve the results proposed by the AMAS by exploiting the ontologist's feedback. Consequently, improving the performance of AMAS may enable to reduce the frequent involvement of the ontologist.

By focusing on finding answers to solve the above mentioned challenges in ontology evolution, this paper contributes to the following key issues in two research areas: (a) Ontology Engineering, by proposing an extended approach [7] of an earlier work [4] in ontology evolution. Indeed, some improvements were achieved in the DYNAMO-MAS approach to better evolve the ontology by implementing some existing adaptive rules [4] and extending it with new adaptation behaviors added to the concept and term agents; (b) AMAS, by developing a tool called ≪ OntoAMAS ≫ that can be triggered when a new document is added to the corpus at any time or when the ontologist makes some improvements in the proposed results. OntoAMAS and the ontologist modify the same ontology in a cooperative and adaptive way: this process relies heavily on the strong relation between the action of one of them and the reaction of the other. The adaptation mechanisms added to the agents behaviors enable them to exploit the ontologist's feedback (elementary and composite changes) and self-adapt to personalize the OntoAMAS proposals. That is to say, we aimed to exploit the ontologist's feedback to personalize OntoAMAS proposals.

The remainder of this paper was organized as follows: Firstly, several related works on the existing approaches for evolving ontologies were described in Section 2, justifying the choice of AMAS for the management of ontologies and discussing the limitations of the current version of DYNAMO that led to the creation of OntoAMAS. Section 3, was devoted to the description of the OntoAMAS approach. The implementation of OntoAMAS was presented and the results were discussed in Section 4. Finally in Section 5, the paper major conclusions were drawn and some future works were suggested.

2. Related works

2.1. Ontology evolution

In the literature, ontology evolution appears as a part of a global scope of ontology maintenance. Stojanovic [1] defines ontology evolution as a process to ≪ adapt and change the ontology in a timely and consistent manner ≫. The process of ontology evolution contains 6 steps: (a) change capture, (b) change representation, (c) change semantics, (d) change implementation, (e) change propagation and (f) the validation. In order to manage these different tasks, several approaches propose: (i) tools and guidelines to capture change needs [8] or to identify new knowledge leading to some changes in the ontology [9] and [10]; (ii) models to represent these changes [1] and [11]; (iii) rules to identify the change semantics to avoid semantic inconsistencies that could occur as a result of these changes [1] and [12]; (iv) tools that implement changes [1] and [13]; (v) other studies ensure the propagation of changes and the update of applications and artefacts connected to the modified ontology [9], [14], [15] and [11]. Ultimately, evolution support tools and versioning were subject of discussion of many other approaches [13].

Several approaches and tools have been proposed in the literature [16]. Each tool seeks to automate one or two stages of the process. Stojanovic [1] was interested in a change representation phase where changes are represented following a specific model. This representation is followed by the change semantics phase, during which syntactic and semantic inconsistencies could appear as a result of changes. The KOANontology evolution tool [1] leads the formulation of changes by suggesting ontology improvements. Klein [9] and Rogozan [15] were rather interested in ontology versioning to manage the changes propagation in order to ensure the consistency of the underlying ontology and all dependent artefacts. Rogozan was inspired by the work of Klein and Stojanovic to propose an approach and a tool to manage the ontology evolution and versioning. To implement this approach, Rogozan [15] provides a tool consisting of two modules: ChangeHistoryBuilder and SemanticAnnotationModifier. Djedidi [12] focused on the steps of semantics of change and change

validation. She proposes an approach and a prototype of evolution (Onto-Evoal: Ontology Evolution-Evaluation). It is an automated process driving the change application while maintaining the evolved ontology consistency. In addition, the tool integrates an evaluation activity supported by a defined ontology quality model. This model is used to guide inconsistency resolution by assessing the impact of the resolutions proposed by the evolution process on ontology quality and selecting the resolution that preserves the quality of the evolved ontology. Luong [14] focused on the stage of change propagation to the dependent artefacts namely semantic annotations. Tissaoui [11] focused on change representation step. He proposed an approach and a tool (EvOnto) that supports a coherent joint change management of (termino-ontologies or TOR) and semantic annotations by anticipating all the consequences of a change on the TOR and on the annotations. This allowed avoiding missing some of the impacts of a change.

It is worth noting, however, that the identification of change and changes representation are manual. The ontologist is often the one who is in charge of detecting the need to change the ontology and expresses this evolution. In general, throughout the evolution process stages, the ontologist was involved in each step. The ontologist chooses to apply the changes in the ontology, places the new entities (concepts, terms, relations), etc. It is a laborious process that requires a lot of time and effort. To minimize the frequent involvement of the ontologist, two solutions proposing an automatic ontology evolution from texts were addressed by two systems: EVOLVA[2] and DYNAMO[3].

EVOLVA [2] contributes in a unique way to the ontology evolution by analyzing the existing domain data, and being based on online ontologies as source of background knowledge for the enrichment step. Indeed, EVOLVA relies on these existing ontologies to seek a relation between the entity to be added to the ontology and the current ones already belong to the ontology. EVOLVA is relevant for the English ontologies since the majority of available ontologies on the Web are represented in English. Therefore, EVOLVA is less useful to enrich French ontologies. Experiments have also shown some limitations in EVOLVA when handling a very specific domain [5]. It has difficulties in detecting relations between a new entity and the already existing ones in the ontology.

The second system is the first prototype of DYNAMO [3] (an acronym of DYNAMic Ontologies). It is not able to evolve the ontology but it allows its construction from scratch. DYNAMO is based on an AMAS to construct and maintain an ontology. The agents of the AMAS implement a distributed clustering algorithm to identify clusters of terms from a large corpus of texts and organize these clusters into a set of concepts in a hierarchy. Each agent represents an extracted candidate term. Thanks to the statistical features, similar terms move closer to create and position concepts. The MAS evolves until all agents are hierarchically linked. The final state of the MAS corresponds to the ontology. The ontologist can then validate or reject. Experimentations carried out with this first prototype of DYNAMO confirmed the inefficiency of statistical approaches [5] when dealing with short texts.

To overcome the previous limitations mentioned in EVOLVA and in the earlier prototype of DYNAMO, DYNAMO-MAS [5] was proposed not as an evolution of the first prototype but as a new approach based on AMAS and using linguistic and statistical criteria. As our proposed approach is an evolution of DYNAMO-MAS, the next section is devoted to justify the choice of AMAS to represent the ontology and to present an overview of the second prototype of DYNAMO.

2.2. DYNAMO-MAS: adaptive multi-agent system for dynamic ontology

The main goal of DYNAMO is to reduce the need for manual interventions in building and evolving ontologies. Using an adaptive multi-agent system to manage ontologies is an original approach proposed by DYNAMO [6]. In the literature, whenever ontology and MAS are discussed, the role of the ontology in ensuring the communication between agents by sharing the same vocabulary is easily remarked. Various research works are interested in exploiting ontologies in MAS especially the Semantic Web [17], [18] and [19]. Nevertheless, the need to evolve ontologies highlights new challenges. Some studies have used MAS for ontology evolution [20] and [21]. The MAS is used as a tool to seek knowledge and ensure the consistency of the ontology [22]. The MAS was used for the first time as an ontology in DYNAMO project [3] and [23] and then in DYNAMO-MAS [5]. Several adaptive techniques (Genetic Algorithm, Artificial Neural Network,

Support Vector Machine) [24] are available but not effective in our case where the environment is open and dynamic (addition of new knowledge, actions of the ontologist) for the following reasons:

—In a genetic algorithm, a population of candidate solutions (individuals) evolves toward better solutions to an optimization problem. In our case of ontology evolution, the search space of the best solution is represented by all possible settings of ≪ n ≫ individuals. Initially, ≪ m ≫ random solutions are generated. Then, to achieve the best solution, the ≪ m ≫ solutions are mutated, altered and selected. Commonly, the algorithm terminates when a satisfactory fitness level (function of evaluation) is reached for the population. Unfortunately, in ontology evolution, it is impossible to give a reliable quantification function that can assess the quality of an ontology. Furthermore, a genetic algorithm is time and memory space consuming. Our goal was to help the ontologist to rapidly evolve an ontology. The use of such algorithms would not be suitable.

— In the context of the problem of ontology evolution from texts, a Neural Network could represent a set of functions-rules that allow the evolution of an ontology from any text belonging to the domain of this ontology. For example, a set of neurons could model a function that describes how and when a candidate term is transformed into a term or a term into a concept, etc. However, the learning phase of such a system would require thousands of ontology evolution examples, which is not feasible in the context of our problem.

— A Support Vector Machine (SVM) requires a large amount of samples to represent a class. However, in our case, this technique would not be effective because we are working in a small amount of data in the corpus.

Therefore, we need an adaptation technique that overcomes these limitations. Following the observations of natural phenomena, an emerging system consisting of a set of entities that interact and self-organize seems to be the appropriate solution to our problem. To define this technique we relied on the adaptive multi-agent systems paradigm that enabled the design of an adaptive system using the emergent functionality.

The underlying reasons for the choice of AMAS are both the properties of the environment where the ontology evolves and the qualities offered by AMAS. Ontology evolution is a complex problem incompletely specified nor does an a priori known algorithmic solution exist. The organizational skills of the AMAS ensure a permanent local self-adaptation to the dynamicity of the environment [25]. Each autonomous agent has a local view and uses perception of its environment to make decisions about actions to take. According to the proper rules to pursue and the local objective to achieve, the agent can adjust its cooperative interactions with other agents to finally lead to a common result [26]. AMAS also allows an easier interactive design of a system. In fact, It enables an incremental construction of the ontology by taking into account new data and new user's needs (the ontologist revises his/her choices according to current results at any time). Indeed, the adaptation is guaranteed by AMAS. Designing an AMAS consists in defining and assigning cooperation rules to agents. The problem to solve in our case is to ensure the consistency of the ontology during the evolution process. Therefore, the designer has to (i) define the nominal behavior of the agent, (ii) deduce the Non Cooperative Situations (NCSs) or exceptions [4] which the agent may come across, and finally (iii) define the actions the agent must perform to come back to a cooperative state and to self-adapt to the environmental dynamics and to the ontologists actions. This self-adaptation of an agent is implemented through three behaviors: the nominal behavior directly related to the partial function of the agent that contributes to the overall emerging function; the cooperative behavior which includes the detection and the resolution of NCS as well as the anticipation and the prevention of the occurrence of NCS; the adaptive behavior which allows the agents to react to the ontologist's actions towards the AMAS proposals. These modifications are considered as a local disturbance by the concerned agents and therefore an inconsistency of the ontology. Adding the adaptive skills may help these agents self-adapt and find their right location inside the ontology again.

Using a MAS to represent an ontology is an originality of the DYNAMO project. In recent years, two approaches have been proposed to manage and maintain Terminological and Ontological Resources (TOR): (i) EvOnto [11] and [27] that support a coherent joint change management of ontology and semantic annotations where the ontologist is involved to choose the appropriate

scenario to apply the list of changes and (ii) DYNAMO-MAS [5] and [28] an adaptive MAS based TOR evolution, used to represent the ontology itself and to produce the ontology, as well.

First, DYNAMO-MAS takes a set of relevant candidate terms and lexical relations as input (Hyperonymy, Meronymy, Synonymy, transverse relations). These very candidates were the results of a corpus analyzer carried out by a term extractor YaTea [29] in the form of triplets (Ti, Rel, Tj). Each triplet has a confidence (Q,I): Q is the quality of the relation(Rel) and I: is the number of relation instances in the corpus. Then, these candidate terms will be agentified and added to DYNAMO-MAS which already has an ontology and contains two types of agents: (i) term agents that represent the terminological component of the ontology and (ii) concept agents representing the conceptual part. During their life cycles, the agents are locally seek their right place inside the ontology. Finally, DYNAMO-MAS generates a new ontology draft as well as a set of proposals as an output to be validated by the ontologist.

2.3. Discussions

To sum up, DYNAMO-MAS was a tool that reacts when new document was added to the corpus. A nominal and cooperative behavior was implemented in each type of agent on how to react to pieces of information received from other agents and how to communicate with them. Each agent has a set of parameters and knowledge enabling the MAS self-organization. The term and concept agents aimed at finding their "right position" (the one that optimizes some of their parameters) in the MAS. In order to propose itself to be part of the ontology, the agent computes its relevance value according to a formula (shown in the Section 3.4). However, the previous work does not take into account the impact of the ontologists actions. In DYNAMO-MAS, experiments [4] showed that linguistic clues are not enough to decide the content of an ontology and that the ontologist has a fundamental role in the ontology evolution process. However, the different reactions of the ontologist towards the AMAS proposals are considered as disturbance by the DYNAMO-MAS. To resolve this problem, exploiting the ontologist's feedback seems to be very relevant. Therefore, an extended approach of DYNAMO-MAS called ≪ OntoAMAS ≫ was proposed consisting in personalizing the AMAS output with the actions of the ontologist (who can be more or less strict: non-static feedback [4]) in order to update the current representation of the ontology.

The originality of our system is to ensure the consistency of the ontology without the frequent involvement of the ontologist. It can be seen as a virtual ontologist that helps the ≪ real one ≫ to carry out an ontology self-adaptation and evolution. The ontologist only acts to give his/her reaction towards a proposal. It is an entire automatic adaptation contrary to [11] and [30] approaches where the ontologist gives his/her reaction and the system guides him/her, by proposing the possible evolution strategies, to finally choose the relevant scenario to apply the list of changes.

Table 1 summarizes and shows a comparison of our proposed approach with the previous presented approaches for ontologies evolution. The comparison is based on the following criteria:

— Ontologist actions: represents the different reactions of the ontologist towards the system proposals;

— Feedback ontologist exploit: means that the system takes into account the modifications generated by the intervention of the ontologist in order to update the internal representation of the ontology;

—Automatic adaptation: means that the system adapt to the ontologist's feedback by determining automatically the appropriate strategy to apply the list of changes;

— iterative process: means that the process of validation and reorganization of the ontology is repeated and ends when the ontologist is satisfied by the modified and consistent ontology;

—Graphical user interface (GUI): means the efficiency of the GUI design in minimizing the time spent to construct the final draft.

3. Proposed approach: OntoAMAS tool for ontology evolution

3.1. Case study

OntoAMAS is a part of DYNAMO (DYNAMic Ontology for information retrieval) project. Our contribution in this project was to propose a method and a tool that allow the evolution of Terminological and Ontological Resources (TOR) from a textual documents corpus in order to facilitate the semantic information retrieval driven by user satisfaction in a dynamic context. A TOR is a resource that consists of a conceptual component (an ontology) and a lexical component (a terminology) [31] and [32]. In fact, a TOR covers not only a set of domain concepts but also a set of associated terms (their linguistic manifestations in documents: each term ≪ denotes ≫ at least one concept). These terms are used to annotate documents in order to facilitate semantic information retrieval within the corpus. The TOR (called ≪ ontology ≫ in the rest of the paper) is formalized using the OWL-based TOR model provided by a partner of the DYNAMO project [33]. The used TOR is a meta-model in which the OWL ontology concepts and associated terms are OWL classes. A ≪ concept ≫ class is denoted by one or more classes (terms). Symmetrically, a ≪ term ≫ class must necessarily have a denotation link toward a concept class.

To avoid the confusion between term and concept we resorted to an example. The concept (Car) can be materialized in the mind as an abstract representation of a bodywork with wheels, a steering wheel, windscreens, etc. This idea can also be achieved as a private car (eg. car Mr. X or black Peugeot of the neighbor, etc.). It is an instance of the concept. To express this concept, we use linguistic forms (words) such as (auto), (car), (jalopy), etc. Thus, the terms serve as means to express concepts.

—A concept: is a general, abstract and mental representation of an object. It can be expressed by a term, symbol or others. Thus, the terms represent linguistic expressions of real or immaterial objects that share common properties. Concerning the concept/instance relation, it is similar to that of class/object. The instance is thus a particular object in the world or a member extension of a concept.

—A term: is a meaningful unit consisting of a word (simple term) or more words (complex term). It identifies a concept uniquely within a domain. A term makes sense in its particular area of application. The same term takes one or more meanings according to the fields (polysemic) or even within the same domain.

One of the most important features of the DYNAMO project is to take into account the potential dynamics of the searched document collection, of the domain knowledge as well as the user's needs evolution. The document collections represent task-oriented technical documents. They have a reasonable size (a few hundred documents) that enable the user to manage them and check their annotation. Our project does not aim at dealing with very large web-extracted corpora.

Our specific interest was to develop a tool that allows the system and the ontologist to modify the same ontology in a cooperative and adaptive way. This process relies heavily on the strong relation between the action of one of them and the reaction of the other. The core of our work was to model an AMAS based tool called ≪ OntoAMAS ≫ designed with ADELFE methodology [34].

3.2. OntoAMAS architecture

OntoAMAS approach for ontology evolution is an iterative and interactive process. It contains 4 main modules (Fig. 1):

—module 1: Adding new documents to the corpus. When new texts are added, new knowledge appears containing new terms and their corresponding concepts.

—module 2: Extraction and filtering of terms and lexical relations. The addition of a new text to the corpus triggers the corpus analyzer which prepares the inputs for MAS. The corpus analyzer includes a term extractor named YaTeA [29], a lexical relation generator and a term and lexical relation selector. The corpus analyzer generates triplets (Ti, Rel, Tj) where Ti and Tj are candidate terms or terms (whether the term belongs or not to the ontology) and Rel is a lexical relation. Each triplet has a confidence (Q, I) where Q is the quality of the relation (value between 1 and 10) and I is the number of instances of the relations in the corpus. The triplets are the inputs of the MAS.

—module 3: OntoAMAS: Knowledge interpretation, ontology updating and self-adaptation to the ontologist's feedback. It corresponds to the first main module of our work. Thanks to the extracted lexical relations or to those existing in general ontology, each new term has to be linked to a concept

that already exists or that will be created in the current ontology by a denotation relation. Each entity (term/concept) is associated to an agent and has a confidence value (more details in the next part) varying along the self-organizing process. Thus, the most relevant new concepts and terms will be proposed to the ontologist. These agents act according to their nominal and cooperative behaviors to determine their ≪ adequate position ≫ (the one that enables the optimization of their parameters) in the organization of the AMAS and enhanced with the adaptive behaviors to self-adapt to the ontologist's feedback towards the proposals given by the AMAS. The MAS constitutes, therefore, the resulting ontology.

In this module, OntoAMAS enhances the nominal and cooperative behaviors already defined in DYNAMO-MAS [24] by (i) implementing some existing adaptive rules (adaptation to acceptance, rejection and moving) and (ii) extending them with new behaviors of adaptation towards elementary and composite changes applied by the ontologist (developed in details in Section 3.4).

—module 4: GUI: The ontologist interventions. It corresponds to the second main module of our work. It is an interface, implemented in the Protégé ontology editor, enabling the ontologist to visualize and control the MAS proposals. Through this interface, the ontologist expresses his/her intention to modify the resulting propositions by applying elementary and composite changes (accept, reject, move, delete, create, split, merge, group, etc.). The adaptation mechanisms offered by the MAS enable OntoAMAS to better adapt its results to the ontologist's feedback. This interaction is repeated until a satisfactory state of a consistent ontology is obtained.

Our work focused on the instantiation of AMAS approach to the issue of ontology evolution. Therefore, The remainder of the paper was devoted to present the characteristics of OntoAMAS and GUI. In the DYNAMO-MAS approach, the AMAS trigger was the enrichment of the corpus by adding new text while in our case, the trigger is both the added document and the intervention of the ontologist by expressing his/her reactions towards OntoAMAS proposals.

The two noticeable difference between our work and DYNAMO-MAS are highlighted in Fig. 1 with blue1 dashed lines.

3.3. The fundamental role of the ontologist

The aim of our work is to personalize the results provided by OntoAMAS with the ontologist actions. These actions can be a set of reactions to OntoAMAS proposals or personal suggestions specific to the ontologist. He/She can apply two types of changes: elementary changes and composite changes. Stojanovic [1] defines an elementary change as an ontology change that modifies (adds or removes) only one entity of the ontology model and a composite change as a change of ontology that can be decomposed into several elementary changes. The ontologist can also manually add new concepts and terms and modifies the location of some entities of the ontology. These actions of change require different ontology evolution strategies depending on several criteria. The role of OntoAMAS is to find ≪ automatically ≫ the adequate strategy thanks to the mechanisms of adaptation allowing agents to self-adapt to the ontologist's feedback. OntoAMAS is based on the declarative approach of Stojanovic [1]. This approach does not present in advance the possible strategies for solving change operations. The ontologist expresses declaratively his/her intention (What) and OntoAMAS executes the relevant evolution strategy (How) to respond to ontologist's needs. So, the ontologist only acts to give his/her reaction towards a proposal which argues the rareness of ontologist involvement. The Table 2 shows the types of evolution changes applied in our approach.

The modifications carried out by the ontologist are considered as a local disturbance by the concerned concept/term agents. Therefore, thanks to the added adaptive behavior to these agents, they will self-adapt and find again their right location inside the ontology.

3.4. Agents adaptive behavior in OntoAMAS

Each agent in the OntoAMAS represents a concept or a term and its autonomous and cooperative behavior is to find its appropriate place in the organization, namely in the ontology. Each agent possesses communication skills and adaptive behaviors to modify and structure the ontology according to different rules. OntoAMAS output is the ontology obtained from the interaction between agents, while taking into account the ontologist feedback [7] when he/she modifies the ontology

according to his/her expertise or the application requirements. It is a construction process where OntoAMAS and the ontologist interact in real-time depending on their respective knowledge. In this section, we present the different consequences of evolution changes on the ontology, and the reactive and adaptive behaviors of agents.

3.4.1. Term agent adaptive behavior

A term agent has a status indicating whether it is part of the ontology (valid term agent) or it is at the proposal stage (candidate or an invalid term). Each term agent has to be related to at least one concept agent. It is also connected to other term agents according to the extracted lexical relations from the corpus. Each relation between term agents is characterized by the confidence value of the triplet (Ti, Rel, Tj). The term agent has three types of behavior: nominal, cooperative and adaptive behaviors. The aim of a term agent is to find its appropriate position in the MAS to finally propose itself to the ontologist. To reach its goals, it has to respect the following rules allowing some local organizational modifications:

—Rule 1: each term agent has to denote at least one concept.

—Rule 2: it treats all its lexical relations and the different requests coming from other concept and term agents.

—Rule 3: a term agent computes its confidence or relevance value (varying between 0 and 10). When this score exceeds a threshold (fixed to 5 but adjustable), the term agent can propose itself to the ontologist. The ontologist can vary this threshold. At the beginning of the evolution process, it starts with a low value (to get several suggestions) and when the ontology offers good annotations of the corpus documents, this value increases.

The adaptive behavior: the ontologist intervenes to revise the OntoAMAS proposals by applying one or more evolution actions enumerated above and the concerned agents will adapt according to their adaptive skills. Therefore, the OntoAMAS reorganizes the agents, improves the already made proposals and proposes others to generate a new ontology proposal.

In total, 5 adaptive behaviors were added to each term agent (adaptation to acceptance, rejection, removing, moving, new term creation). Below, a descriptive example for each term agent adaptive behavior was presented.

Adaptation to acceptance or rejection: To ensure the self-adaptation of a term agent to ontologist acceptance or rejection, we enhanced it with the following behaviors.

When the ontologist accepts or rejects a term agent, the latter sends a message to its neighbors (agents with whom it already has interacted) but not yet validated to inform them. Therefore, they recalculate their relevance value (between 0 and 10).

For example (Fig. 2), when the term agent Problem is accepted or rejected, it informs its neighbors: Problem FailureFull-size image (<1 K). This message enables it to update its knowledge and recalculate its relevance value (can be adjusted throughout the evolution process). If it exceeds the threshold (fixed to 5) and respects the ≪ Term Proposal Condition ≫ (TPC), it proposes itself to be part of the ontology.

TPC: a term agent cannot be proposed if the concept it denotes has not be proposed first. When a concept agent is rejected, then its term agents (not yet proposed) wait until the concept agent proposes itself again.

Adaptation to a new term creation: When the ontologist proposes a new term agent to be associated with a concept agent, it sends a denotation request containing its label. The concept agent accepts and notifies the term agent. Therefore, it informs its neighbors not yet validated by sending a message. They recompute their relevance values. If they exceed the proposal threshold, then, they may be proposed to the ontologist.

Adaptation to removal: When the ontologist proposes to remove a term agent, it informs its neighbors not yet validated (term agents and concept agents) by message. They recompute their relevance values. If they exceed the proposal threshold, then, they can be proposed to the ontologist.

If the concerned term agent is the only term that denotes its concept agent, then this concept will disappear automatically from the AMAS (Execution of the concept agent adaptation behavior to

removal). Otherwise, the concept agent will choose a new term agent to denote it and then determines its new label. The preferred term agent is the one having the denotation link with the highest weight.

Adaptation to moving: When a term agent is moved by the ontologist, it informs its neighbors not yet validated, by sending a message. For example, if the term agent (Data Failure) is moved, then it sends a message to the concept agent (Data Failure) and the term agent (Problem Failure).

The term agent (Data Failure) moves to the concept agent (Failure) to denote it Full-size image (<1 K) (Fig. 3). If the concept agent (Data Failure) is already proposed then it will be removed automatically.

If the term agents (Data Failure) and (Problem Failure) are related with a synonymy relation (Data Failure target) then the term agent (Problem Failure) computes its relevance value. If the moving of (Data Failure) to (Failure) increases the relevance of the term agent(Problem Failure) then (Problem Failure) moves to the concept agent (Failure) Full-size image (<1 K).

If there is hyperonymy relation between (Data Failure) and (Problem Failure), then the term agent (Problem Failure) sends a request to the concept agent (Failure) asking for its parent concept agent Full-size image (<1 K). The concept agent (Failure) processes the request and notifies the term agent (Problem Failure) by a message containing his parent concept agent (Default) Full-size image (<1 K). (Problem Failure) sends a request to (Default) to establish a denotation link. (Default) accepts and sends a notification to (Problem Failure). Therefore, (Problem Failure) moves to (Default) to create a denotation link with it Full-size image (<1 K). If the concept agent (Problem Failure) has already proposed then it will be removed automatically.

Adaptive term agent relevance value: OntoAMAS has to learn how to automatically adjust the agent relevance value according to ontologist actions. In fact, the ontologist can be more or less strict. For this reason, we propose to add a new parameter to the formula of relevance value called “Strictness Degree” (Stdegree). We consider that over time, the stricter the ontologist is and refuses proposals, the less luckier the non-proposed ones become to be proposed and therefore the value of Stdegree increases and the agent relevance value has to be decreased. Inversely, the more the ontologist accepts proposals, the more the non-proposed agents could be considered as interesting. Consequently, the relevance value has to be increased by reducing the Stdegree.

Thus, we propose to adjust the Stdegree parameter after a given number of successive ontologist feedbakcs. This adaptation mechanism was implemented with the following formula:

where P1 is the highest relevance value of its lexical relations, P2 represents the quality of its term agent neighbors (valid or not yet); P3 corresponds to the status of its lexical relations (accepted or refused); P4 indicates the diversity of the lexical relations of the term agent. After various experiments with DYNAMO-MAS [5] the values empirically fixed as follows: $\alpha 1\alpha 1 = 0.5$, $\alpha 2\alpha 2 = 2$, $\alpha 3\alpha 3 = 2$ and $\alpha 4\alpha 4 = 1$. These values best weighed the parameters of the agent relevance.

Initially Stdegree = 2; the Coef value varies according to the ontologist feedback:

—if the ontologist accepts the term agent then

where P1 is the highest relevance value of its lexical relations, P2 represents the quality of its term agent neighbors (valid or not yet); P3 corresponds to the status of its lexical relations (accepted or refused); P4 indicates the diversity of the lexical relations of the term agent. After various experiments with DYNAMO-MAS [5] the values empirically fixed as follows: $\alpha 1\alpha 1 = 0.5$, $\alpha 2\alpha 2 = 2$, $\alpha 3\alpha 3 = 2$ and $\alpha 4\alpha 4 = 1$. These values best weighed the parameters of the agent relevance.

Initially Stdegree = 2; the Coef value varies according to the ontologist feedback:

—if the ontologist accepts the term agent then

where TermAgRelevance is the current relevance value of the concerned term Agent and Threshold is fixed to 5.

After 10 successive ontologist actions, OntoAMAS starts supervising his/her behavior.

—If the number of acceptance actions exceeds the number of rejection or removing actions, the TermAgRelevance has to be increased by adjusting the value of Stdegree, Stdegree = 3.

—If the number of rejection or removing actions exceeds the number of acceptance ones, the TermAgRelevance has to be decreased by adjusting the value of Stdegree, Stdegree = 1.

—Otherwise, Stdegree keeps its initial value, Stdegree = 2.

3.4.2. Concept agent adaptive behavior

As mentioned in Section 3.4.1, the concept agent also has three types of behavior: nominal, cooperative and adaptive behaviors. The nominal and cooperative behaviors were summarized in the following rules:

—Rule 1: each concept agent has a status indicating whether it is a candidate concept (at proposal stage) or not (part of the ontology). It has conceptual relations with other concept agents and linked to term agents with a denotation relation. Each conceptual relation has a status indicating whether it is processed, non-processed or rejected relation.

—Rule 2: to define its favorite label (the label of the term having a the denotation link with the highest relevance value).

—Rule 3: if its relevance value exceeds the threshold, it proposes itself to the ontologist.

The adaptive behavior: according to its adaptive skills, the concept agent self-adapts to ontologist actions enumerated in Table 2.

In total, we added 9 adaptive behaviors to each concept agent (adaptation to acceptance, rejection, renaming, removing, sub-concept creation, moving, merging, split, grouping).

A descriptive example was presented below for each concept agent adaptive behavior.

Adaptation to an acceptance: When a concept agent is accepted by the ontologist, it informs its neighbors not yet validated by sending a message. This message allows them to recompute their relevance values. For example, when the concept agent Failure is accepted, it informs the term agent Failure, the concept agents Data Failure and Problem Failure of its acceptance Full-size image (<1 K) (Fig. 4). Therefore, each agent recalcultes its relevance. If it exceeds the proposal threshold and it respects the Term Proposal Condition and the Concept Proposal Condition (CPC), then it proposes itself to be part of the ontology.

CPC: A concept agent cannot be proposed unless its concept parent has already been suggested.

Adaptation to rejection: When a concept agent is rejected, it prevents its other linked agents. Since the concept agent has a cooperative behavior, it proposes a new concept parent to its invalid neighbors agents. It can be either its concept parent agent or the TOP concept agent. The concept agent chooses to propose the one that has the highest relevance value. A valid concept agent is more effective than TOP concept agent and the TOP concept agent is more relevant than an invalid concept agent that has not exceeded the threshold of relevance.

Thus, the rejected concept agent informs its neighbors not yet validated by sending a message including their new parent concept agent. Upon receiving this message, a concept agent recomputes its relevance value. If it exceeds the threshold proposal, it moves to the new parent concept to be able to propose itself. For example, when the concept agent Failure is rejected by the ontologist, it sends an information message of rejection to the concept agents Data Failure and Failure Problem and the term agent FailureFull-size image (<1 K) (Fig. 5). These agents recompute their relevance. The concept agent Data Failure exceeds the proposal threshold. Then it sends to the concept agent Default a request for establishing an (is-a) relation Full-size image (<1 K). The latter accepts and sends to Data Failure a notification message Full-size image (<1 K). Upon receiving this message, the concept agent Data Failure creates a new conceptual relation with the concept agent DefaultFull-size image (<1 K) and removes its old relation with the concept agent FailureFull-size image (<1 K). Therefore, the concept agent Data Failure can propose itself to the ontologist.

Adaptation to renaming: When the ontologist proposes to rename a concept agent, the AMAS automatically creates a new term agent with the same label to denote the concept agent. The new term sends a denotation request to the concept agent and the latter accepts and notifies it. Then, it informs its neighbors,not yet validated, by message containing its new label. They recompute their relevance values. If they exceed the proposal threshold, then they can be proposed to the ontologist.

Adaptation to removal: When the ontologist proposes to delete a concept agent, it informs its neighborhood, not yet validated, by sending a message. This message allows them to recompute their relevance values.

For example, if the concept agent (Failure) has to be removed, it informs its neighborhood: the term agent (Failure), its sub-concept agents (Data Failure) and (Problem Failure) and the concept agent

(Exception) and it proposes a new parent concept agent (Default) Full-size image (<1 K) (Fig. 6). The concept agents (Data Failure) and (Problem Failure) and the term agent (Failure) recompute their relevance values. If they exceed the proposal threshold, then, (Data Failure) and (Problem Failure) send a request to establish an (is-a) relation with (Default) and the term agent (Failure) sends a denotation request to (Default) Full-size image (<1 K). The latter accepts and notifies them. Therefore, (Data Failure), (Problem Failure) and the term agent (Failure) move to (Default). The concept agent (Failure) has no more a term to denote it. Thus, it disappears from the AMAS. Otherwise, the concept agent (Failure) will disappear with their sub-concept agents (they have no more parent concept agent) from the AMAS. If the relevance score of the concept agent (Exception) exceeds the threshold, then it may be proposed to the ontologist.

Adaptation to a sub-concept creation: When the ontologist proposes to create a sub-concept agent, AMAS automatically creates a term agent that has the same label as the new sub-concept. The term agent sends a denotation request to it and the latter accepts and notifies it. Then, the sub-concept agent sends a request message to its parent concept agent to establish an (is-a) relation. The latter accepts and notifies it. Then, it informs its neighbors, not yet validated, by sending a message containing its label and its status (valid). it recomputes its relevance value. If it exceeds the threshold, therefore it may be proposed to the ontologist.

Adaptation to moving: When a concept agent is moved by the ontologist, it informs its neighbors, not yet validated, by sending a message.

For example, the concept agent (Exception) has to move to be a sub-concept of (Default). Thus, it informs the term agent (Failure) and the concept agents (Data Failure) and (Problem Failure) Full-size image (<1 K). Then, it sends a request to establish an (is-a) relation to (Default) Full-size image (<1 K). The concept agent (Default) accepts this request and notifies (Exception) Full-size image (<1 K). Therefore, (Exception) creates an (is-a) relation with (Default) (Fig. 7).

The concept agents (Data Failure) and (Problem Failure) and the term agent (Failure) recompute their relevance values with the intention to move to (Default). If the relevance exceeds the threshold, then, (Data Failure) and (Problem Failure) send a request to establish an (is-a) relation with (Default) and the term agent (Failure) sends a denotation request to (Default). The latter accepts and notifies them. Therefore, the term agent (Failure), (Data Failure) and (Problem Failure) move to (Default). The concept agent (Failure) has no more a term to denote it. Thus, it disappears from the AMAS.

Adaptation to merging: When the ontologist proposes to merge a concept agent with a validated one, the AMAS reacts and adapts to this action by running a set of operations.

For example, when the ontologist proposes to merge the concept agent (Exception) with the validated concept agent (Failure), (Exception) informs its sub-concepts agents (DB Exception) and (System Exception) and the term agent (Exception Error) by sending a message containing the new suggested parent concept agent (Failure) Full-size image (<1 K) (Fig. 8). (DB Exception), (System Exception) send a request message to (Failure) to establish an (is-a) relation and (Exception Error) sends a denotation request Full-size image (<1 K). If their new relevance values(with the new parent concept) increase, then the concept agent (Failure) notifies them by sending a message of acceptance Full-size image (<1 K). Therefore, (DB Exception), (System Exception) and (Exception Error) move to (Failure).

If (Exception) has no more sub-concepts or term agents then it disappears automatically from the AMAS otherwise it will be removed, necessarily, by running the removal concept operation (detailed in section Adaptation to concept removal).

Adaptation to split: When the ontologist proposes to split a concept agent, the AMAS automatically creates a new concept agent and associates it to the same parent concept agent. For example, when the ontologist proposes to split the concept agent Component, it creates a new concept agent.

Component proposes to its sub-concepts to move to the new one by sending a message Full-size image (<1 K) (Fig. 9). This change operation generates a Non Cooperative Situation of ambiguity (which one has to move?). These sub-concepts are linked together thanks to the lexical relation of synonymy between their associated term agents. Thus, according to the statistical features,

Levenshtein distance [35], term agents can estimate their dissimilarity with others. Then, one sub-concept agent sends a message called vote, to its parent agent Component, including the list of its brothers concept agents sorted by dissimilarity Full-size image (<1 K). The selected sub-concept agent Frame sends a request to establish (is-a) relation with the new concept agent Full-size image (<1 K). The latter accepts and notifies (Frame) Full-size image (<1 K). Therefore, the concept agent Frame moves to the created agent Full-size image (<1 K). The split operation generates a Non Cooperative Situation of inefficiency which is the election of a preferred concept label. The new concept agent has to be renamed with an appropriate label. To resolve this problem, the AMAS asks the ontologist for help by executing the operation of ≪ concept agent adaptation to renaming ≫.

Adaptation to grouping: When the ontologist proposes to group two concept agents, the AMAS creates a new concept agent as a parent concept agent for them. For example, the ontologist chooses the two concept agents (Window) and (Panel) for grouping. The AMAS reacts immediately by creating a new concept agent as a sub-concept of the concept parent of (Window) and (Panel). The relevance value of the created concept agent is the maximum one between the relevance values of (Component), (Window) and (Panel).

The concept agent (Panel) removes its relation with (Component) Full-size image (<1 K) (Fig. 10). Then, it establishes an (is-a) relation with the new concept agent Full-size image (<1 K). Simultaneously, (Panel) proposes the new parent concept agent to (Window) Full-size image (<1 K). Therefore, (Window) cancels its old relation with the concept agent (Component) and establishes an (is-a) relation with the new one Full-size image (<1 K).

The grouping operation generates a Non Cooperative Situation(NCS) of inefficiency which is the election of a preferred concept label. The new concept agent has to be renamed with an appropriate label. To resolve this problem, the AMAS asks the ontologist for help by executing the operation of renaming concept.

Adaptive concept agent relevance value: As mentioned above in the section of term agent relevance value, OntoAMAS has to learn how to automatically adjust the concept agent relevance value according to ontologist actions by implementing the adaptation mechanism using the following formula:

where P1 is the highest relevance value of its conceptual relations, P2 represents the quality of its concept agent neighbors (valid or not yet); P3 corresponds to the status of its conceptual relations (accepted or refused); P4 is the depth of the concept agent in the ontology (−1 if it is connected to the concept ≪ TOP ≫, otherwise 1); P5 indicates the quality of the term agents denoting the concept agent (relevant or not) and the different weights α1,α2,α3,α4,α5α1,α2,α3,α4,α5 are respectively fixed to 0.5, 1, 1, 1 and 2 [5].

Initially Stdegree = 2; the value of Coef varies according to the ontologist feedback:

—if the ontologist accepts the concept agent then

Algorithm 1.

General algorithm of the agent adaptive behavior

where ConceptAgRelevance is the current relevance value of the concerned concept Agent and Threshold is fixed to 5.

After 10 successive ontologist actions, OntoAMAS starts supervising his/her behavior.

—If the number of acceptance actions exceeds the number of rejection or removing actions, the ConceptAgRelevance has to be increased by adjusting the value of Stdegree, Stdegree = 3.

—If the number of rejection or removing actions exceeds the number of acceptance ones, the ConceptAgRelevance has to be decreased by adjusting the value of Stdegree, Stdegree = 1.

—Otherwise, Stdegree keeps its initial value, Stdegree = 2.

We present the general algorithm of the agent adaptive behavior in Algorithm 1.

4. Evaluation of OntoAMAS

In this section, we introduced some implementations and experiments to test the feasibility of our proposed approach in addition to an evaluation that validates our hypothesis that the AMAS

performance (quality of OntoAMAS results) has been improved and that user time during evolution has indeed been decreased through the use of the GUI with comparable results to DYNAMO-MAS [4].

4.1. Implementation and experiments

We implemented OntoAMAS tool as a plug-in in Protégé ontology editor. OntoAMAS extends TextViz [33] another Protégé plug-in dedicated to documents semantic annotation. The behaviors of OntoAMAS agents have been developed with the platform Make Yourself Agents [36] (MAY: Eclipse plug-in). When the ontologist adds a new document (one or more) to the corpus, new candidate terms and new lexical relations are extracted by the corpus analyzer. When it sends them to the AMAS, the OntoAMAS is triggered; new term agents are created and possible local interactions with other agents occur in order to find their appropriate location in the organization Full-size image (<1 K) (Fig. 11). Most of the agents update their knowledge, communicate and react to finally reach a stable state. Therefore, a new draft of the ontology is proposed to the ontologist as a set of proposals in the GUI. All the concepts and the terms of the ontology are displayed in the Concepts Browser panel Full-size image (<1 K) and in the Terms Browser panel Full-size image (<1 K). The red-colored proposals represent the terms and concepts with a relevance value that exceeds the threshold of proposition. The ontologist expresses his/her reaction by applying elementary and composite changes via the GUI Full-size image (<1 K). Each ontologist's action raises a feedback to the AMAS and triggers it again. The added adaptation mechanism to the agents behaviors allows them to update their knowledge again and to recompute their relevance values. Therefore, an update of the proposals will take place and potentially leads to an update of the display. Finally, the ontology will be saved in an OWL format (a standard that makes it easily reusable). Only validated proposals by the ontologist will be saved in the OWL.

We start testing our tool by implementing the adaptive behaviors towards elementary changes. We have carried out OntoAMAS tests with two different ontologies and an associated corpora. Each ontology has been manually built and validated regarding to the corpus annotation. The ontology seeks to annotate its associated corpus that is accurate enough to support an efficient document retrieval. In these two domains, the corpus contains a reduced number of small size documents (a few paragraphs).

—Arkeotek ontology: a French ontology on archeology about traditional techniques, built with 380 concepts and 733 terms and associated to a corpus made up of 299 documents (rule-based formulation of scientific papers).

—Artal ontology: an English ontology on software bugs reports made up of 887 terms and 582 concepts; the corpus is composed of 287 documents (bug report files).

We considered two metrics to evaluate OntoAMAS:

1. Quality Evaluation: to prove the relevance of OntoAMAS suggestions: we checked if the ontologist considered them as valid.

2. Performance Evaluation to test the effectiveness of GUI design and its user-friendliness in minimizing the time spent to construct the final draft.

4.1.1. Quality evaluation

To evaluate the quality of OntoAMAS proposals, we firstly compared a manual and an automatic ontology evolution and then we made a comparison between our tool results and the ones of DYNAMO-MAS starting by reproducing the same experiment [5]. The manual evolution was performed by the ontologist who added new terms and new concepts to the ontology. The evolution is automatic if the ontologist uses our tool. In each domain, an ontologist has expressed his/her feedback towards the MAS proposals via the graphical interface (Fig. 11).

After the addition of 12 new documents to the corpus of Arkeotek, the ontology was enriched by the ontologist with the insertion of 7 new concepts and 19 new terms. Starting with the same ontology and the same new added documents, OntoAMAS proposed 27 new concepts (16 were accepted by the ontologist) and 32 new terms (22 were accepted by the ontologist). The details of this result are shown in Table 3. The first row of the table, the AMAS proposal, represents the sum of the rows 2, 3 and 4 (relevant, correct and wrong/useless/refused proposals). The accepted proposals are the result of the

sum of the two table rows representing the relevant proposals (set at the right place in the ontology) and the correct ones (set at a wrong place in the ontology). The third row represents the wrong/useless/refused proposals and the last row represents the proposals suggested both by the ontologist and OntoAMAS.

With the Arkeotek dataset, OntoAMAS reached 68.75% of correct and relevant term proposals and 59.26% of correct and relevant concept proposals. When we compare the manual and the automatic evolution of the ontology, we note that, OntoAMAS suggested 18 terms and 14 concepts, that were not identified by the ontologist, to enrich the ontology. So the AMAS can be seen as an interesting tool to help an ontologist to accurately evolve his/her ontology.

After the addition of 21 new documents to the Artal corpus, the ontology was enriched by the ontologist with the insertion of 9 new concepts and 19 new terms. Starting with the same ontology and the same new added documents, OntoAMAS proposed 18 new concepts (10 were accepted by the ontologist) and 24 new terms (16 were accepted by the ontologist). Table 6 shows more details of these results.

With the Artal dataset, OntoAMAS reached 67% of correct and relevant term proposals and 56% of correct and relevant concept proposals. When we compare the manual and the automatic evolution of the ontology, it can be noted that, our tool suggested 12 terms and 9 concepts not proposed by the ontologist, to enrich the ontology.

As the process of the ontology evolution is iterative, we repeated the step of the ontologist intervention until a satisfactory and consistent ontology was obtained (the consistency is defined by the ontologist and the applications depending on the ontology). In each iteration we tested (i) the number of accepted terms and concepts proposed by OntoAMAS, (ii) the number of wrong, useless and refused ones and (iii) we compared the proposals generated by OntoAMAS to those suggested by the ontologist.

To confirm the reliability of the ontologist's role in improving the quality of OntoAMAS proposals, we tested our tool after two successive interventions of the ontologist. We presented in this evaluation only 3 iterations however the number of ontologist's interventions is not static and depends on the consistency of the ontology and the satisfaction of the ontologist. Table 4 and Table 5 show the following results:

The second iteration indicates that OntoAMAS attains, with Arkeotek ontology, 70% of accepted term proposals and 66.6% of accepted concept proposals. By exploiting the ontologist's feedback, the number of proposals generated by both the ontologist and OntoAMAS increased to 5 terms and 3 concepts. The third iteration reveals an improvement in the number of accepted term proposals to 73.33% and accepted concept suggestions to 69.23% and an increase in the number of similar terms and concepts made both by the ontologist and the OntoAMAS.

In the case of the Artal ontology, the second iteration in Table 7 shows that OntoAMAS reached 69% of accepted term proposals and 70% of accepted concept proposals. By exploiting the ontologist's feedback, the number of proposals generated by both the ontologist and the OntoAMAS increased to 5 terms and 2 concepts. The third iteration reveals an improvement in the number of correct and relevant term proposals to 75% and accepted concept suggestions to 71.5% and an increase in the number of similar terms and concepts made by both the ontologist and the OntoAMAS as shown in Table 8.

To evaluate the efficiency of OntoAMAS, the proposal results of DYNAMO-MAS and OntoAMAS were compared (see Table 9). After the addition of 12 new documents to the Arkeotek corpus, DYNAMO-MAS generated 68.75% of accepted term proposals and 59.26% of accepted concept proposals. However, the same test proved that OntoAMAS reached 73.3% of accepted term proposals and 69.23% of accepted concept proposals.

After the addition of 21 new documents to the Artal corpus, DYNAMO-MAS generated 67% of correct and relevant term proposals and 56% of correct and relevant concept proposals [5]. However, the same test proved that our OntoAMAS reached 75% of correct and relevant term proposals and 71.5% of correct and relevant concept proposals.

4.1.2. Performance evaluation

Besides the improvement of the quality of the results provided by the OntoAMAS, our second challenge was to reduce the time to obtain an evolved ontology. Indeed, The limitation of DYNAMO [3] and [6] and DYNAMO-MAS [5] and [24] stems from the lack of user-friendliness which ensures a reliable interaction with the system results. Even with a limited size of an ontology, the ontologist faces difficulties in following the several changes achieved autonomously in only a very short time (few seconds). On the other hand, a small intervention of the ontologist can potentially generate important impacts on the structure. Consequently, the ontologist spends a lot of time (around three hours [23]) to localize the involved concepts and terms and the new suggestions in the graph display. A great part of the time spent to evolve the final draft ontology is due to the difficult handling of the GUI. In DYNAMO-MAS, experiments [24] showed that the time taken to automatically evolve an ontology is longer than a manual maintenance. This is due to the response time during validation of a proposal using PROMPTDIFF tool. To deal with this problem, we resorted to an ergonomist to design an easy-to-use GUI (Fig. 11). Designing an optimal interface enhances the interaction time and makes it shorter. During the experiments we noticed the efficiency of our interface: it allows displaying the current ontology to the ontologist with red-colored new proposals which dynamically changes.

4.2. Analysis and discussions of results

To summarize, OntoAMAS is independent of the language and the domain of the handled texts. It runs an ontology evolution in two different domains and supports two languages (English and French). The obtained results are promising and confirm the efficiency to take into account the role of the ontologist's intervention in the improvement of the OntoAMAS suggestions. Thanks to the iterative and interactive aspect of the evolution process, after each successive iteration we noticed improvements in the quality of OntoAMAS proposals. In fact, the number of the wrong and useless proposals decreases and the number of relevant ones increases. This due to the following two reasons:

— The implementing of the adaptation mechanism with the formula that represents the agent relevance value: thanks to the parameter ≪ Strictness Degree ≫ (Stdegree) defined above in Section 3, OntoAMAS learns how to vary the relevance value of the agent according to the interactions with the ontologist. Subsequently after a given number of the ontologists actions, the OntoAMAS proposals will be updated by eliminating those that do not exceed the proposals threshold and by adding those that exceed it.

— The cooperative behaviors of the agents: as mentioned in Section 3, each agent has local knowledge about itself and its ≪ neighbor ≫ agents with whom it has already interacted. Each agent has a status (candidate or not) indicating whether the agent is actually part of the ontology(valid agent) or is at proposal stage (invalid agent). It behaves according to cooperative interaction rules. It updates its knowledge and recomputes its relevance value according to the updated parameters values.

When the ontologist makes his/her decision (accepts, rejects, modifies, etc.), the concerned agent will be notified and updates its status. For example if the ontologist accepts an agent to be a apart of the ontology, it updates its status to become a valid agent. Therefore, its neighborhood updates their knowledge and recalculates its relevance value which necessarily will be increased thanks to the parameter P2 representing the quality of its agent neighbors (valid or not yet). When the value of P2 contributes to increasing the agent relevance value above the threshold, then it proposes itself to be part of the ontology. However if the ontologist rejects a proposal, the parameter P2 decreases and therefore the relevance value decreases. P5 is another parameter in the formula which may affect the relevance value of the concept agent. P5 represents the proportion of relevant term agents that denote this concept agent. Thus, when the ontologist validates a term agent, P5 increases; else it decreases and consequently the relevance value of the concerned concept will be reduced and is likely disappear from the updated proposals list.

Moreover, we should note that the OntoAMAS converges and that the average of the number of the ontologist's interventions (actions towards the proposals of OntoAMAS) progressively decreases. After each successive iteration carried out by the OntoAMAS and the ontologist, the improvements in the quality of OntoAMAS proposals (the number of relevant proposals increases and the wrong and

useless proposals decreases) facilitate the task of the ontologist and consequently reduce his/her frequent involvement.

5. Conclusion

The main goal of this paper was to propose OntoAMAS, an ontologist feedback tool based on an adaptive multi-agent system for an ontology evolution. OntoAMAS is an innovative tool from three points of view:

—From the point of view of ontology evolution, this work is a first step showing the relevance of the DYNAMO-MAS extension. The originality comes from the fact that OntoAMAS can be triggered when new document is added to the corpus at any time or when the ontologist makes some improvements in the proposed results. OntoAMAS and the ontologist modify the same ontology in a cooperative and adaptive way: this process relies heavily on the strong relation between the action of one of them and the reaction of the other. That is to say, thanks to the adaptation mechanisms added to the agents' behaviors, OntoAMAS exploits the ontologist's feedback (elementary and composite changes) and self-adapts to personalize its proposals.

—From the adaptive multi-agent modeling point of view, in our situation of self-organization, complexity and emergence through cooperative and adaptive interactions, AMAS (agentification and a number of heuristic rules) has been a relevant solution and proved its efficiency in the context of a dynamic ontology.

—From the point of view of time performance, designing and developing OntoAMAS graphical interface proved its efficiency when the time taken to ensure the interaction between the ontologist and the AMAS gets shorter.

We carried out some experiments of OntoAMAS to evaluate its performance and the quality of its suggestions by (i) implementing the adaptive behavior of the agents towards the elementary changes made by the ontologist and (ii) designing an user-friendly GUI. The obtained results are promising. The current state of our tool enables to decrease the number of the wrong and useless proposals and to increase the relevant ones. OntoAMAS proposes some of the terms and/or concepts that have been produced manually by the ontologist and also concepts and terms that have been forgotten by him.

From these improvements we mostly focused on enhancing the adaptation mechanisms. We are currently implementing the adaptive behavior of the term and the concept agents towards the composite changes (they possess richer semantics to express evolution [37]). We plan also to experiment the quality of OntoAMAS proposals with ontologies concerning different domains, expressed in different languages (French and English). In the validation step, If only one user participated in the evaluation, what if another one performed better? Cross-user agreement would be a great addition to this evaluation.

Acknowledgement

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

References

1. L. Stojanovic, Methods and tools for ontology evolution, Ph.D. thesis, University of Karlsruhe, Germany, 2004.
2. F. Zablith, M. Sabou, M. dAquin, E. Motta, Ontology evolution with evolva, in: The Semantic Web: Research and Applications, Springer, Berlin, Heidelberg, 2009, pp. 908–912.
3. K. Ottens, M.-P. Gleizes, P. Glize, A multi-agent system for building dynamic ontologies, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2007.
4. Z. Sellami, V. Camps, An adaptive multi-agent system to co-construct an ontology from texts with an ontologist, in: Transactions on Computational Collective Intelligence XV, Springer, 2014, pp. 101–132.
5. Z. Sellami, V. Camps, N. Aussenac-Gilles, Dynamo-mas: a multi-agent system for ontology evolution from text, J. Data Semantics 2 (2–3) (2013) 145–161.
6. K. Ottens, Un systme multi-agent adaptatif pour la construction d'ontologies partir de textes, Ph.D. thesis, Universit Paul Sabatier, Toulouse, France, 2007.
7. S. Benomrane, Z. Sellami, M.B. Ayed, A.M. Alimi, An adaptive multi-agent system for ontology co-evolution, in: Proceedings of the International Conference on Agents and Artificial Intelligence, Lisbon, Portugal, 2015, pp.216–221.
8. P. Cimiano, J. Völker, Text2onto, in: Natural Language Processing and Information Systems, Springer, 2005, pp. 227–238.

9. M.C.A. Klein, Change management for distributed ontologies, 2004.
10. P. Plessers, O. De Troyer, Ontology change detection using a version log, in: The Semantic Web–ISWC, Springer, 2005, pp. 578–592.
11. A. Tissaoui, N. Aussenac-Gilles, P. Laublet, N. Hernandez, Evonto: un outil d'évolution de ressource termino-ontologique pour l'annotation sémantique, Tech. Sci. Inform. 32 (2013) 817–840.
12. R. Jedidi, Approche dévolution dontologie guidée par des patrons de gestion de changement, Ph.D. thesis, Citeseer, 2009.
13. G. Flouris, On belief change in ontology evolution, Artif. Intell. Commun. 19 (2006) 395–398.
14. P.H. Luong, Gestion de l'évolution d'un web sémantique d'entreprise, Ph.D. thesis, École Nationale Supérieure des Mines de Paris, 2007.
15. C.D. Rogozan, Gestion de l'évolution des ontologies: méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions d'ontologie, Ph.D. thesis, Université du Québec à Montréal, 2009.
16. F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, M. Sabou, Ontology evolution: a process-centric survey, Knowl. Eng. Rev. 30 (01) (2015) 45–75.
17. V. Tamma, T. Bench-Capon, An ontology model to facilitate knowledge-sharing in multi-agent systems, Knowl. Eng. Rev. 17 (2002) 41–60.
18. L.-K. Soh, Multiagent, distributed ontology learning, in: Working Notes of the 2nd AAMAS OAS Workshop, Bologna, Italy, 2002.
19. D. Greenwood, M. Lyell, A. Mallya, H. Suguri, The ieee fipa approach to integrating software agents and web services, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2007, p. 276.
20. S. Slimani, S. Baïna, K. Baïna, Interactive ontology evolution management using mutli-agent system: a proposal for sustainability of semantic interoperability in soa, in: 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, 2011, pp. 41–46.
21. S. Slimani, S. Baïna, K. Baïna, A framework for ontology evolution management in ssoa-based systems, in: 2011 IEEE International Conference on Web Services (ICWS), IEEE, 2011, pp. 724–725.
22. M. Hadzic, D. Dillon, An agent-based data mining system for ontology evolution, in: On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Springer, 2009, pp. 836–847.
23. K. Ottens, N. Hernandez, M.-P. Gleizes, N. Aussenac-Gilles, A multi-agent system for dynamic ontologies, J. Logic Comput. 19 (2009) 831–858.
24. Z. Sellami, Gestion dynamique d'ontologies à partir de textes par systèmes multi-agents adaptatifs, Ph.D. thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2012.
25. S.A. Deloach, W.H. Oyenan, E.T. Matson, A capabilities-based model for adaptive organizations, Autonom. Agents Multi-Agent Syst. 16 (2008) 13–56.
26. L. Panait, S. Luke, Cooperative multi-agent learning: the state of the art, Autonom. Agents Multi-Agent Syst. 11 (2005) 387–434.
27. A. Tissaoui, N. Aussenac-Gilles, N. Hernandez, P. Laublet, EVONTO – joint evolution of ontologies and semantic annotations, in: International Conference on Knowledge Engineering and Ontology Development (KEOD), INSTICC – Institute for Systems and Technologies of Information, Control and Communication, Paris, 2011, pp. 226–231.
28. Z. Sellami, V. Camps, N. Aussenac-Gilles, S. Rougemaille, Ontology coconstruction with an adaptive multi-agent system: principles and casestudy, in: Knowledge Discovery, Knowledge Engineering and Knowledge Management, Springer, 2011, pp. 237–248.
29. S. Aubin, T. Hamon, Improving term extraction with terminological resources, in: Advances in Natural Language Processing 5th Int. Conference on NLP, FinTAL, Springer, Turku, Finland, 2006, pp. 380–387.
30. A. Parvizi, Y. Ren, M. Vigo, K. van Deemter, C. Mellish, Z.J. Pan, R. Stevens, C. Jay, Language and ontologies 2015, Association for Computational Linguistics, 2015. Chapter: Ontology Authoring Inspired By Dialogue.
31. A. Maedche, Ontology learning for the semantic web, Springer Science & Business Media, 2002.
32. P. Cimiano, Ontology learning from text, Springer, 2006.
33. A. Reymonet, J. Thomas, N. Aussenac-Gilles, Modelling ontological and terminological resources in owl dl, in: OntoLex 2007-Workshop at ISWC07, Busan, South-Korea, 2007.
34. G. Picard, Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente, Ph.D. thesis, Université Paul Sabatier Toulouse III, 2004.
35. V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady, vol. 10, 1966, p. 707.
36. V. Noel, J.-P. Arcangeli, M.-P. Gleizes, Component-based agent architectures to build dedicated agent frameworks (regular paper), in: International Symposium From Agent Theory to Agent Implementation(AT2AI), Austrian Society for Cybernetic Studies, Vienna, 2010.
37. A. Tissaoui, Évolution cohérente des ressources termino-ontologiques et des annotations sémantiques, Ph.D. thesis, Université Paul Sabatier, Toulouse, France, 2013.