

# Agent negotiation based ontology refinement process and mechanisms for service applications

Li Li · Yun Yang

Received: 30 November 2006 / Revised: 19 July 2007 / Accepted: 8 February 2008 / Published online: 14 March 2008  
© Springer-Verlag London Limited 2008

**Abstract** Nowadays organisations are willing to outsource their business processes as services and make them accessible via the Web. In doing so, they can dynamically combine individual services to their service applications. However, unless the data on the Web can be meaningfully shared and is interpretable, this objective cannot be realised. In this paper, a new agent-based approach for managing ontology evolution in a Web services environment is exploited. The proposed approach has several key characteristics such as flexibility and extensibility that differentiate this research from others. The refinement mechanisms which cope with an evolving ontology are carefully examined. The novelty of our work is that inter-processes between different ontologies are studied from the agent's perspective. Based on this perspective, an agent negotiation model is applied to reach an agreement regarding ontology discrepancy in an application. The efficiency and effectiveness of reaching an agreement over an ontology dispute is leveraged by the private negotiation strategy applied in the argumentation approach. An extended negotiation strategy is discussed to enable sufficient information in decision making at each negotiation round. A case study is presented to demonstrate ontology refinement in a Web services environment.

**Keywords** Ontology evolution · Multi-agent systems · Agent negotiation · Service applications

## 1 Introduction

It is evident that service applications are key characteristics of the e-business world. Service-oriented computing provides a means for different organisations to connect their applications with one another to conduct business spanning organisation boundaries. With an ever-increasing number of businesses entering into e-business equipped with Web services features, there is a trend to compose available services on the Internet. Thereby increasing competitiveness by taking a customer-centric paradigm. For example, a supply chain, formed by composition of existing services, is defined as a network of facilities that procure raw materials, transform them into intermediate goods and then finished products, and deliver the products to customers through a distribution system [8].

There is no denying that there are great challenges to be faced in order to fully realise the above potential. In order to facilitate information exchange between different organisations, the first thing that urgently needs to be considered is how these organisations understand each other based on their individual ontologies. It is well known that an ontology is the conceptual backbone that provides meaning to data on the Web. The vision of the Web can only be realised through proliferation of well-known ontologies describing different domains. In other words, ontologies are the foundations of service applications. However, an ontology is not a static resource and may evolve over time, especially where different domains are concerned. Ontologies are constantly being adapted to the changing requirements, hence are subject to continuous changes. Changes in the domain,

---

L. Li (✉)  
CSIRO ICT Centre, GPO Box 664, Canberra, ACT 2601, Australia  
e-mail: lily.li@csiro.au

L. Li · Y. Yang  
Centre for Information Technology Research,  
Swinburne University of Technology,  
Melbourne, VIC 3122, Australia  
e-mail: yyang@swin.edu.au

conceptualisation and explicit specification can cause changes in an ontology [15]. In detail, changes can take place as follows:

- new concepts need to be added to the ontology;
- outdated concepts need to be eliminated from the ontology;
- the common ontology (in integration) is required to change (e.g. add, delete, and update) with the changes brought by new information resources; and
- better ways of organising information are available.

In a nutshell, an ontology evolves with changes in the environment. Hence, ontology evolution can be defined as a timely adaptation of an ontology and consistent propagation of changes to the dependent artifacts. The complexity of ontology evolution increases as the number and the size of ontologies grow, so an insight into potential solutions in ontology refinement is required. In this paper, an agent negotiation based approach is developed. We provide an innovative solution to cope with ontology changing on the Web, which has become a noticeable issue when innovative enterprises advance to service applications. With the presence of agent technologies, it is possible that involved agents are more flexible and credible in their methods of modelling business processes with interaction protocols.

To this end, this paper is presented to tackle ontology change from a multi-agent system (MAS) perspective. It is focused on the refinement of individual ontologies. First, problems in ontology refinement are identified in Sect. 2. Related work is briefly reviewed in Sect. 3. Based on the in-depth analysis of current problems in ontology refinement, Sect. 4 extends a strategic-negotiation model to reflect ontology change in an particular environment. A refinement process and corresponding mechanisms are presented in Sect. 5. Section 6 presents a case study to demonstrate the proposed approach and discusses the relevant issues of agent negotiation is given. Finally, Sect. 7 concludes our work and points out future work.

## 2 Problems in ontology refinement

A business environment is changing since business objectives and resource constraints vary from organisation to organisation. A changeable environment forces underlying ontologies to evolve over time. For this reason, ontology refinement needs to investigate, in depth, the requirements of dynamic changes in the environment and rapid development of the Web. The research in ontology evolution is still in its infancy [15]. Undoubtedly, there are some critical issues that ontology management needs to address before the proliferation of ontologies is made possible in the field such as

supporting semantics-based search, interoperability, service compositions and Semantic Web applications. The major issues are as follows:

- addressing requirements of environmental change;
- reflecting environmental change and guiding corresponding ontology refinement accordingly; and
- developing an ontology refinement architecture.

The very viable and rapid growth of the Web has made ontology change even more prominent than ever before. The timely handling of ontology refinement and provision of a conformance view for agents involved in a certain scenario are thus seen as the foundations for further communication to achieve the goal. Therefore, a much more detailed analysis of ontology evolution becomes a higher priority on the agenda of ontology research. In this paper, we attempt to provide potential solutions to some of the aforementioned challenges.

Although developing procedures for reflecting ontology change and for selecting appropriate actions to be taken by agents in a MAS is an essential requirement for successful applications of ontologies (e.g. ontology evolution), methodologies and tools to support this complex task are largely missing. Among available techniques, agent technique, which promises to bring enormous benefits by permitting the inclusion of learning and self-improvement capabilities in an adaptive way, and participating in interaction with built-in knowledge [17], is identified as a suitable way of coping with ontology change in a dynamic and heterogeneous environment. Agent interaction, a potential solution for reflecting dynamic changes in an environment, is seen to be promising in resolving problems such as what information is known by agents in a MAS and how interactions can affect agents' succeeding actions. More importantly, interactions facilitate the semantical understanding between agents through provided ontologies.

Given multiple ontologies for different organisations, finding a conformance view for all participating agents in advance seems impossible as far as heterogeneity, distribution, autonomy and evolution are concerned. A run-time solution is highly needed in this regard. In other words, we expect that agents engaged in ontology management at run-time are able to make decisions on how ontologies evolve by themselves, rather than by using previously specified rules. This is because few rules defined a priori will be effective, given that changes are unpredictable.

With all these problems in mind, we are seeking a semantic understanding in a business scenario with the support of agent technologies. It is worth noting that we attempt to provide a flexible solution by modelling agents' interactive and negotiated activities in a collaborative environment. In saying so, our assumption is that initial ontologies should be available. Also we do not intend to expand our work to include

ontology change recursively. In other words, we focus on the stage of how an agent refines its ontology in adapting to the change of an environment rather than tracing the impact of change propagation to other ontologies or ontology-based applications. The core idea underlying our approach is to provide the corresponding strategy and mechanisms to cope with ontology evolution over time.

### 3 Related work

Ontologies are becoming an integral part of many industrial and academic applications such as supporting semantics-based search, interoperability support, service compositions and other applications. As ontology development becomes a more ubiquitous and collaborative process, ontology dynamics management is becoming an important area of ontology research. Even though coping with ontology change is an essential requirement in ontology usage and ontology engineering, appropriate tools and strategies for enabling and managing evolution are still missing. Only very few approaches exist.

In [13] the author presents some guiding principles for building consistent and principal ontologies in order to facilitate their creation, usage and maintenance in widely distributed environments.

The approach for ontology evolution given in [19] is very interesting. In this work an ontology is used to specify the semantics of possible changes to a knowledge base. It presents a six-phase evolution model to check the ontology after changes have been made with the possibility of reversing these changes. In this paper, an ontology for specifying change operations is presented. Other major approaches are the evolution approach and versioning approach. The former tries to manage the problem of dynamics in its total complexity, whereas the latter relies on the use of different versions of ontologies to reduce the complication of the problem [2]. For example, Noy and Klein [15] discuss an ontology evolution system by extracting the operations from two versions of one ontology. They develop a framework [5] for managing ontology evolution by extracting the operations from one ontology version and transferring to another. For another example, the KAON project (<http://kaon.semanticweb.org/>) [12] is flexible, by enabling the users to control and customise the manner in which changes are resolved. However, this technique requires that all change resolution strategies be specified a priori. One possible solution is to allow the system to determine its own course of action by satisfying the user's requirement.

Ontology evolution that can be treated as part of an ontology versioning mechanism is analysed in [4]. Klein and Fensel provide an overview of causes and consequences of the changes in ontology. However, the most prominent

shortage is the lack of a detailed analysis of the effect of specific changes on the interpretation of data.

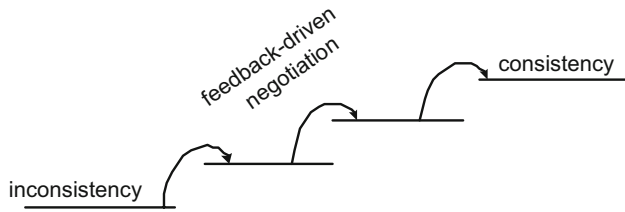
Tamma and Bench-Capon [21] present an extended ontology knowledge model to describe what is known by agents in a MAS. However, the model is not used for supporting ontology evolution.

Other research communities have also influenced the research into ontology evolution, which benefits from many years of research into database and knowledge-based system evolution [14].

In contrast to the previous work that addresses ontology evolution in one form or another, we go one step further by allowing agents to take part in the evolution process and reflect ontology change in the environment. Applying an agent-based approach to ontology refinement is twofold. First, it fills various gaps in understanding of a specific issue by providing an appropriate approach to capture the differences whenever they arise. Second, reflections from agents have a great impact on agent which have identified a discrepancy in understanding between themselves and other agents on the basis of their ontology. In turn, this prompts the agent to perform some managerial tasks to its ontology repository. With widespread use of ontologies, we believe this approach is applicable in areas such as e-marketplaces, virtual organisations and service applications where interaction plays a vital role in overcoming the problems that evolution has faced for a long time.

### 4 Improved strategic-negotiation strategy for ontology refinement

Negotiation in MAS includes a negotiation set, a protocol, a collection of strategies, and rules [23]. In this section, a strategic-negotiation strategy and argumentation-based model [16,20] are detailed followed by the definition of the agent utility function to allow an agent to estimate how “good” the current result/state is. Negotiation here aims to eliminate conflicts/inconsistencies in order to reach an agreement over specific negotiation issues. Figure 1 graphically demonstrated that agents approach the goal gradually after several rounds of negotiations. Obviously, developing individual strategies which allow agents to efficiently negotiate towards the desired goal is the primary goal of this stage. Normally a strategy is private, i.e., each agent has its own strategy profile which is invisible to other agents. In contrary to reaching an agreement for a price or shipping time, the term negotiation here is an *open cry* setting to reach the agreement if conflicts in understanding each other occur. The private strategy advises an agent what sort of reactions is appropriate, whereas the argumentation model allows an agent to put forward arguments, justify the acceptability of these

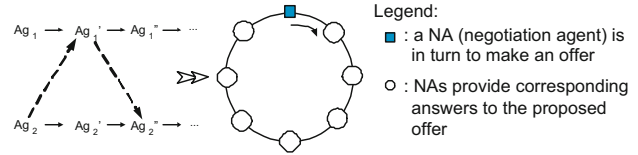


**Fig. 1** Feedback-driven negotiation

arguments. As such, an ontology may be refined by the yielded arguments. More details are below.

The private strategy used here is a strategic-negotiation strategy based on Rubinstein’s model of alternative offers [22]. Please refer to [6] for a general view of negotiation between self-interested agents in a MAS. In the strategic-negotiation model, a set of agents is defined as  $\mathcal{AG} = \{Ag_1, Ag_2, \dots\}$ . It is assumed that the agents can take actions defined as  $\mathcal{Ac} = \{\alpha_1, \alpha_2, \dots\}$ , which are available from agents’ action repertoire in a time sequence set  $T = \{0, 1, 2, \dots\}$ , that is known to the agents. A sequence set is defined as  $\mathcal{S} = \{s_1, s_2, \dots\}$ , corresponding to the time sequence.  $\forall t \in T$  of a negotiation period, if negotiation is still going on without any agreement being reached, the agent, with its turn to make an offer at time  $t$ , will suggest a possible solution to other agents that may choose one of the following three answers. Each of them may either accept an offer by choosing (Yes), reject an offer by choosing (No), or opt out of an offer by choosing (Opt). If an offer is accepted by all agents, negotiation is terminated and then followed by implementation. Let  $fo = (G_{index}, t)$  be the released offer that an agent makes at time  $t$ , where  $G_{index}$  annotates a subgraph of a particular taxonomy ontology. Generally speaking, in order to reach an agreement in ontology management, simply choosing either Yes, No, or Opt is insufficient. For the purpose of providing more information in each negotiation round, the spectrum of the feedback is extended to include the following activities:

- 1: No.  
This indicates that the agent knows nothing about the offering question.
- 2: Yes.  
It explicitly indicates that the agent knows the content. Implicitly, it is one hundred percent in agreement with a specific negotiation issue suggested by the agent which is currently making an offer.
- 3: Yes<sub>x</sub> (Yes to some extend).  
This provides the feedback with an approximate percentage to describe the agent’s opinion.
- 4: Opt (Opt out).  
If at least one of the agents opts out of the negotiation, then the negotiation terminates.



**Fig. 2** Negotiation process—strategy determined by interactions and agent’s status

By studying the interaction from an agent’s perspective, we intend to reflect agents’ responses to received messages which were sent by a particular agent in the previous or current round of negotiation. An agent’s perception-and-action process will decide which kind of strategy this agent will have when its turn comes. In other words, the strategy indicates what to offer at time  $t + 1$  by this agent. The process is shown in Fig. 2, where each agent has its own strategy profile decided by its individual global view at a certain time. On the left of Fig. 2, the interaction between agents is represented by agents’ interaction along with their ongoing perception-and-action processes. For simplicity, only two agents/processes are shown. The picture on the right of Fig. 2 depicts how the negotiation process is conducted based on the reflections from the previous round. It is expected that any reflection from the previous or current round will affect an agent’s next action.

The process is thus described as a restricted token passing cycle with each agent making offers in turn, but other rational agents provide positive/negative feedback as much as possible. According to the above discussion, the potential solution to a corresponding offer is defined as:  $fs = (\lambda, fo)$ , where  $\lambda$  means the current agent’s agreement upon a specific concept in the form of a percentage.

In reaching an agreement, agents work by constructing a series of arguments (i.e. if an agreement is among No, Yes and Yes<sub>x</sub>) of interest by following an argumentation approach [23]. The basic form of arguments is defined as:  $\mathcal{T}(\Sigma) \vdash (\varphi, \Gamma)$ , where  $\mathcal{T}(\Sigma)$  is an ontology represented by a formal ontology language,  $\varphi$  is an attempt offer (by this agent) at the current round, and  $\Gamma$  is a subset of the ontology  $\mathcal{T}(\Sigma)$ , i.e.  $\Gamma \subseteq \mathcal{T}(\Sigma)$  such that  $\Gamma \vdash \varphi$ .

With these in mind, different feedback defined earlier in this section can be formally represented as follows:

- No:  $\Gamma \vdash \emptyset$
- Yes:  $\Gamma \vdash \varphi$  where  $\varphi \in \mathcal{T}(\Sigma)$
- Yes<sub>x</sub>:  $\Gamma \vdash \varphi_\lambda$  where  $\varphi_\lambda \in \mathcal{T}(\Sigma)$ , and  $\lambda$  is a numeric number in a form of percentage

It is worth defining utility functions to tell an agent whether or not a series of arguments are “good” and how “good” they are.

We assume that an agent is proactive, it should obviously act in the direction of maximising its welfare at each stage of a process by considering environmental changes. In our approach, we adopt a utility function defined in [23] as:  $u : \Omega \rightarrow R$ , where  $\Omega = \{v_1, v_2, \dots\}$  is the set of proposition that agents have preferences over,  $R$  is the set of real numbers. For two elements  $v$  and  $v'$  ( $v, v' \in \Omega$ ), if  $u_i(v) \geq u_i(v')$ , then we say  $v$  is preferred by agent  $i$  at least as much as  $v'$ .

In this paper, the following two classes of arguments are identified:

- *Equivalent argument* An argument  $(\varphi, \Gamma)$  is equivalent if  $\Gamma = \emptyset$ . Three subclasses are defined with the context of an ontology.
  - (1)  $v_1$ . The class of all same label arguments that may be made from  $\mathcal{T}(\Sigma)$ .
  - (2)  $v_2$ . The class of all synonymic arguments that may be made from  $\mathcal{T}(\Sigma)$ .
  - (3)  $v_3$ . The class of all same attribute (of a concept) arguments that may be made from  $\mathcal{T}(\Sigma)$ .
- *Consistent argument* An argument  $(\varphi, \Gamma)$  is consistent if  $\Gamma$  is consistent. One generic subclass of argument is identified under this category.
  - (4)  $v_4$ . The class of all arguments that may be made from  $\mathcal{T}(\Sigma)$ .

Intuitively, some types of argument are more desirable than others. In saying so, there exists an order,  $\succeq$ , over the above classes:  $v_1 \succeq v_2 \succeq v_3 \succeq v_4$ .

From the above discussion, it is clear that an agent has preference over  $\{v_1, v_3, v_3, v_4\}$  in decreasing order.

## 5 Ontology refinement process and mechanisms

As an ontology is changing over time, it is clear that there must be a dedicated process to look after ontology dynamics to save us from overlooking it. In the framework presented in [10, 11], different kinds of agent shown in Fig. 3 have been defined as follows:

- *User Agent (UA)* A user agent is designed to only know the interface agent (*IA*). This agent interacts with a particular GUI. This includes getting the business scenario from the GUI and passing it on to *IA* to display the proper results on the user interface (e.g. GUI) when it receives a return message from *IA*.

- *Interface Agent (IA)* This agent interacts with the *UA*, which includes getting the business scenario from a particular GUI and passing it on to the virtual community,<sup>1</sup> and then presenting the *UA* with the expected results. It acts as a broker in an attempt to help various agents find each other in a distributed environment.
- *Integration Agent (InA)* This is responsible for ontology integration based on a certain business scenario. The major tasks of an *InA* are to: (1) count the appearance of each specified concept; and (2) filter unexpected items before sending them to the *OA*.
- *Mapping Agent (MA)* This is shaped to provide linkages to pave the way for the interoperability of heterogeneity of various ontologies on the Web. It is the foundation of further ontology management towards interoperability. The major task of an *MA* is to estimate whether two given concepts map to each other according to its knowledge.
- *Refinement Agent (RA)* This maintains ontology coherence and integrity in an environment where they may change frequently. The major tasks of a *RA* are to: (1) obtain up-to-date information for a specified concept; and (2) locate any differences between a previous description and the current one.
- *Ontology Agent (OA)* This agent acts on behalf of a certain ontology. It behaves properly in a specified agent platform. It is equipped with the functionalities of a certain ontology, e.g. it operates over the ontology structure and a particular intermediate result structure. The main purpose of an *OA* is to perform ontology related tasks which are isolated from external *functionary agents*. The presence of an *OA* allows flexible system organisation.
- *Negotiation Agent (NA)* This agent takes part in negotiation setting with an attempt to obtain evolving ontology information for the corresponding *OA*.
- There are many formal languages, such as frame-based language, RDF(s) (<http://www.w3.org/RDF/>), and OWL (<http://www.w3.org/TR/owl-guide/>) for ontology representation. They are shown on the right of Fig. 3. The integrated ontology is the resulting ontology of the proposed approach.

Particularly, the *RA* is designed to be responsible for tackling evolution during run-time. Incorporating ontology refinement into ontology mapping and integration to provide a holistic view of ontology management takes our approach beyond the existing work in this field. Properly bound ontology refinement with ontology mapping and integration makes our approach unique in that it can directly obtain changes and seamlessly use them to take succeeding actions. In this paper, our focus is on the shadowed items within the

<sup>1</sup> A virtual community is a group of partners commonly interested in a certain business scenario.

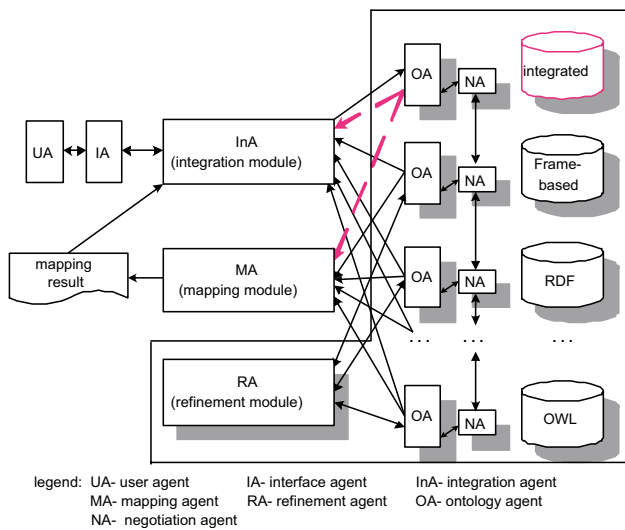


Fig. 3 Ontology refinement in ontology management

boundary in Fig. 3. Ontology reuse has also been taken into account in the architecture. It is envisaged that our design is potentially able to adequately cope with ontology management requirements in a dynamic and heterogeneous service application environment.

Note that in this paper we tend not to expand our work to include ontology change recursively. In other words, we focus on the stage of how an agent refines its own ontology in adapting to the change of an environment rather than tracing the impact of change propagation to other ontologies.

### 5.1 Refinement process

In our work, the refinement module starts when the RA is enacted. The refinement process runs in the background to provide OAs with ontology modification in a timely fashion. Briefly speaking, the RA monitors the NAs negotiation processes via individual OAs. Agreements reached by NAs during negotiation are fed back to corresponding OAs, which take appropriate actions to modify their ontologies. We assume that OAs are able to present integral and consistent ontologies at a certain point whenever they are asked (e.g. by MA and InA). The refinement process is outlined as follows:

- (1) NAs get to start when the RA starts up;
- (2) NAs start negotiation according to the strategic-negotiation model (see Sect. 4);
- (3) OAs modify their ontologies based on negotiation results respectively.

Figure 4 displays interactions (in AUML as defined in [1]) among agents involved in the refinement process. The refinement process influences existing ontologies and OAs as well. After NAs reach agreements or when the negotiation process

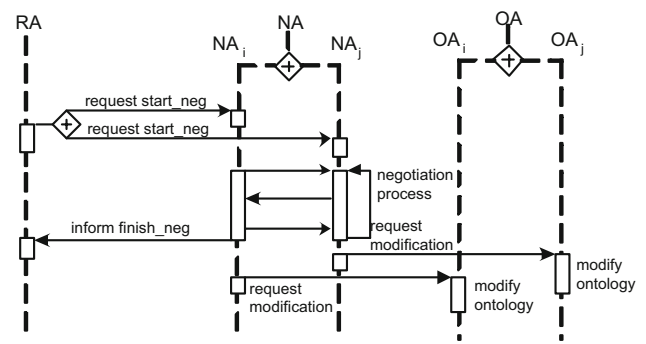


Fig. 4 Interactions from RA's view in AUML

is terminated, OAs then modify corresponding ontologies. As ontologies are changing over time, the checking agent (CA) runs continuously to guarantee the above process by presenting consistent ontologies for further ontology operations.

### 5.2 Refinement mechanisms

The refinement module is responsible for ontology refinement. The RA is in charge of refining ontologies whenever possible by means of the negotiation process. The negotiation process includes an agent presenting an initial offer, together with other agents providing potential solutions in terms of the initial offer. After that, the same agent checks potential solutions from others to make a decision on its next action (e.g. exit the process or keep track of refinement for the next time). Five relevant functions (i.e. Pseudocodes A-E) are presented below to describe the negotiation module. They are described in the same sequence as follows: (1) defining data structures and variables used; (2) defining some functions; and (3) describing corresponding process algorithms in pseudocode. The pseudocode of an agent's behaviour which is not in turn to give an initial offer, is presented first. The pseudocode of an agent's behaviour which is in turn to give an initial offer is shown second. The pseudocode to describe the negotiation process is then presented followed by the ontology modification pseudocode. Finally, based on the above processes, the pseudocode of the refinement process is given.

#### Pseudocode A. Agent behaviour (i.e. an answering agent):

/\*assuming four potential solutions from an agent which is not in turn to give an initial offer: "Yes", "No", "Opt" and "Yesx" (see Sect. 4);

*m*: the number of available ontologies, also the number of OAs;

*j*: a randomly generated integer number between 1 and *m* standing for one of the OAs;

*solution*: a variable for one of the above four solutions;

*λ*: a variable of estimated percentage (i.e. it is acceptable to

some extent);  
 evaluate: a function to evaluate the initial offer;  
 send1: a function to send back  $\lambda$  percentage to indicate how many percentages upon an agreement;  
 send2: a function to send back *solution*.  
 \*/

```

1. Function solution_offer {
2. for ( $i = 1; i < m; i++$ ) {
3.   if ( $i! = j$ ) {
4.     solution=evaluate initial offer;
5.     switch (solution) {
6.       case “Yesx”: send1; break;
7.       default: send2;
8.     } // end switch
9.   } // end if
10. } // end for
11. } // end function

```

Function *solution\_offer* describes an agent’s behaviour in response to an offer. This agent has Yes, No, Opt and Yesx four options. It sends back a numeric percentage number,  $\lambda$ , when this agent is not very sure about the answer.

#### Pseudocode B. Agent behaviour (i.e. an offering agent):

/\*assuming four possible solutions: see Pseudocode A for detail;  
 $m$ : the number of available ontologies, also the number of OAs;  
*solution*: a variable for one of the above four solutions;  
*threshold1*: a percentage threshold to filter unexpected items;  
*threshold2*: a threshold used in ontology refinement;  
*change\_list*: a list keeping track of changes;  
 $k_{j,y}, k_{j,n}, k_{j,o}$ : the number of occurrences of potential solutions “Yes”, “No” and “Opt”, respectively;  
 $\lambda$ : a variable of estimated percentage;  
*decision*: decisions that an agent makes according to its knowledge and current change of the environment;  
 make\_decision: a function to decide what the next action (for a particular agent) is;  
 modify: a function to modify the *change\_list* with the potential *decision* made in *decide\_next*.  
 \*/

```

1. Function check_initial_offer {
2. do {
3.   switch (solution) {
4.     case “Yes”:  $k_{j,y}++$ ; break;
5.     case “No”:  $k_{j,n}++$ ; break;
6.     case “Opt”:  $k_{j,o}++$ ; break;
7.     default: if ( $\lambda \geq \text{threshold1}$ )  $k_{j,y}++$ ;
8.       else  $k_{j,n}++$ ;
9.   } // end switch
10. if ( $k_{j,o} == m - 1$ ) return  $k_{j,o}$ ;

```

```

11. if ( $k_{j,n} \geq \text{threshold2}$ ) exit;
12. if ( $k_{j,y} == m - 1$ ) {
13.   make_decision;
14.   modify;
15. } // end if
16. } while (exists potential solutions from agents in negotiation setting); // end do
17. } // end function

```

Function *check\_initial\_offer* checks other agents’ answers, and counts them individually.

#### Pseudocode C. Negotiation process:

/\*  $m$ : the number of available ontologies, also the number of OAs;  
 $j$ : a randomly generated integer number between 1 and  $m$  standing for one of OAs;  
 generate\_nextInt: to generate a random integer number;  
 initial\_offer: an agent (decided by a randomly generated integer number) to issue an initial offer;  
 solution\_offer: previously defined function;  
 check\_initial\_offer: previously defined function.  
 \*/

```

1. Function negotiate {
2. do {
3.    $j = \text{generate\_nextInt}(m) + 1$ ;
4.   initial_offer( $j$ );
5.   solution_offer( $j$ );
6.   check_initial_offer( $j$ );
7. } while (negotiation time is not out); // end do
8. } // end function

```

Function *negotiate* presents the process of agent negotiation which is particularly designed to reflect ontology change. The process (shown in Fig. 2), similar to a restricted token passing cycle, is able to take place by an agent randomly. This agent first generates an initial offer, then it will be checking answers from other agents to help make a decision upon what to do next. This process may continue if negotiation is still going on without any agreement being reached.

#### Pseudocode D. Ontology modification:

/\* *change\_list*: a list keeping track of changes (as defined in Pseudocode B);  
*top*: an element from a list;  
 insert: a function for inserting operation on the hierarchical structure of a specified ontology;  
 delete: a function for deleting operation on the hierarchical structure of a specified ontology;  
 update: a function for updating operation on the hierarchical structure of a specified ontology;  
 get\_head: get the head element of a list;  
 \*/

```

1. Function ontology-modify {
2. do {
3.   top=get_head(change_list);
4.   switch {
5.     case "insert": insert; break;
6.     case "delete": delete; break;
7.     default: update;
8.   } // end switch
9. } while (top!=Null); // end do
10. } // end function

```

Function `ontology_modify` may take a suitable operation depending on what kind of change has been made in `change_list`.

#### Pseudocode E. Refinement algorithm:

```

/* change_list: a list keeping track of changes (as defined
in Pseudocode B);
initialise: initialise the process;
negotiation: previously defined function;
ontology_modification: previously defined function
in which OAs modify their acting ontologies according to
negotiation results, referring to change_list for any changes.
*/

```

```

1. Function refine {
2.   initialise;
3.   negotiate;
4.   ontology-modify;
5. } // end function

```

Function `refine` makes an attempt to reflect the main idea of this paper. basically, ontology modification only takes place as a result of agent negotiation.

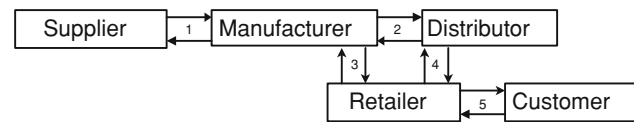
## 6 Case study and discussion

In this section, a case study is presented to demonstrate the proposed approach and mechanisms followed by a discussion.

### 6.1 Case study

A representative example of a supply chain is to transform raw materials and components into a finished product that is delivered to the end customer. Supply chain management (SCM) [7] aims to integrate suppliers, manufacturers, distributors, retailers and customer logistics to dramatically reduce time, redundant effort, and inventory costs.

Advanced information techniques, especially recent development of agent technologies and SOC infrastructure, make SCM more efficient by helping companies coordinate, schedule, and control procurement, production, inventory



**Fig. 5** A simplified e-business supply chain example

management, and delivery of products and services. Haworth's SCM systems [3] and P&G supply chain network [18] are successful running examples.

With software agents representing the individual organisations, such as suppliers and distributors of a supply chain, it is expected that participating agents would optimise the entire system by providing firm-wide enterprise service applications within cost, time frame and quality requirements.

Let us look at an e-business supply chain shown in Fig. 5. Agents' behaviours are modeled by their interactions with the changing environment. In saying so, we mean agents can adapt to the environment to perform whatever they are supposed to do in order to achieve the assumed objectives. Different functionalities that agents might need to accomplish to build an optimal system, described below, correspond with the numbering in Fig. 5.

- (1) Agents negotiate with potential suppliers and make their selections based on the suppliers' bids upon an agreed cost and time frame, or some other previously agreed measure. Agents may need to negotiate with other potential suppliers to schedule delivery and make an alternative delivery schedule if needed.
- (2) Agents, based on the orders, schedule shipments from manufactures to distribution companies' warehouses. Lower or stockless inventory and just-in-time methods are available by using service-oriented technology.
- (3) Agents collect transaction data from retailers and send them to manufacturing and production.
- (4) Agents transmit transaction data to sales and marketing.
- (5) Agents allow retailers to adjust stock items to meet customers' needs.

It is not unusual for different agents, which act on behalf of corresponding components, to have different ontologies. Bearing this in mind, we take a close look into different parties in Fig. 5. It is worth noting that the supply chain illustrated in Fig. 5 has been simplified. It is common that there are multi-tiered suppliers, manufactures, distributors and retailers/customers in a supply chain, especially in large business. Usually a few potential service providers are available to roughly meet the requirements. These service agents cannot reach an agreement unless they can understand each other. Moreover, interested agents are free to join and leave this virtual community at any time. Usually it is most unlikely to regulate participating agents to conform to a uniform

ontology. In this sense, involved agents need to resolve misunderstandings that arise due to the dynamic environment and different ontologies.

Let us assume a wine (beer) industry e-business supply chain with Supplier, Manufacturer, Distributor and Retailer/Customer in place, each with its own ontology. Practically, multi-tiered structure in a supply chain is common. Suppose there is a supply chain with four Distributors (i.e.  $D_1, D_2, D_3$  and  $D_4$ ) and one Retailer, each with an ontology.  $D_1$  has an ontology (i.e.  $\mathcal{T}_1(\Sigma_1)$ ,  $\mathcal{T}_1$  for short) from DAML ontology library (<http://www.daml.org/ontologies/66>).  $D_2$  is with an ontology (i.e.  $\mathcal{T}_2$ ) built on the definition of term “beer” from WordNet (<http://wordnet.princeton.edu/>).  $D_3$  maintains an ontology (i.e.  $\mathcal{T}_3$ ) which is based on the basic types of beer provided by the website [http://www.dma.be/p/bier/1\\_2\\_uk.htm#](http://www.dma.be/p/bier/1_2_uk.htm#), while  $D_4$  possesses the Australian beer types ontology (i.e.  $\mathcal{T}_4$ ) based on information from websites: (1) [http://www.australianbeers.com/beers/beer\\_types/beer\\_types.htm](http://www.australianbeers.com/beers/beer_types/beer_types.htm); and (2) [http://www.fosters.com.au/beer/about/beertypes/beer\\_types.asp](http://www.fosters.com.au/beer/about/beertypes/beer_types.asp). Retailer holds an ontology (i.e.  $\mathcal{T}_R$ ) from the website <http://www.purl.org/net/ontology/beer.owl>.

With the presence of these ontologies, it is natural for Retailer to take them into consideration because an interaction between Retailer and Suppliers is necessary in reaching an agreement about everything concerned. An ontology is of fundamental importance to the participating Suppliers and Retailer.

According to the argumentation-based negotiation, an argument is formally defined as a pair,  $(\varphi, \Gamma)$ , over an ontology  $\mathcal{T}(\Sigma)$ , such that  $\Gamma \vdash \varphi$ , where  $\Gamma \subseteq \mathcal{T}(\Sigma)$ . The following are possible arguments built by an agent on behalf of Retailer if we focus on a scenario in which Retailer may need to contact different service providers (i.e. Distributors), and refine its ontology whenever needed.

- $a_1: (Beer(\mathcal{T}_R) \sim Suds(\mathcal{T}_2), \emptyset)$
- $a_2: (Lager(\mathcal{T}_2) \sim Lager(\mathcal{T}_R), \emptyset)$
- $a_3: (Beer(\mathcal{T}_1) \sim Beer(\mathcal{T}_R), \emptyset)$
- $a_4: (Oktoberfest(\mathcal{T}_2) \sqsubseteq Lager(\mathcal{T}_R),$   
 $\{(Oktoberfest(\mathcal{T}_2) \sqsubseteq Lager(\mathcal{T}_2)),$   
 $Lager(\mathcal{T}_2) \sim Lager(\mathcal{T}_R)\})$
- $a_5: (SweetStout(\mathcal{T}_1) \sqsubseteq Stout(\mathcal{T}_R),$   
 $\{SweetStout(\mathcal{T}_1) \sqsubseteq Stout(\mathcal{T}_1), Stout(\mathcal{T}_1) \sim Stout(\mathcal{T}_R)\})$
- $a_6: (Stout(\mathcal{T}_R) \sqsubseteq \neg Porter(\mathcal{T}_R),$   
 $Stout \sqsubseteq \neg Porter) (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4)$  (i.e.  $Stout \sqsubseteq \neg Porter$   
holds in  $\mathcal{T}_1, \mathcal{T}_2$ , and  $\mathcal{T}_3$ )
- $a_7: (Stout(\mathcal{T}_R) \sqsubseteq Ale(\mathcal{T}_R),$   
 $Stout \sqsubseteq Ale) (\mathcal{T}_2, \mathcal{T}_4)$  (i.e.  $Stout \sqsubseteq Ale$  holds in  $\mathcal{T}_2$ ,  
and  $\mathcal{T}_4$ )

Argument  $a_1$  is in  $v_2$  (see Sect. 4), while  $a_2, a_3$  are in  $v_1$ . All other arguments  $a_4, a_5, a_6$  and  $a_7$  are in  $v_4$ . As such, the

*Refinement Agent (RA)* on behalf of Retailer will be able to carry out its ontology refinement in relation to other agents’ feedback. The question we need to answer next is the justification. How can we justify that all derived  $\varphi$  are correct? As we noticed that  $\varphi$  is proved by  $\Gamma$ , where  $\Gamma \subseteq \mathcal{T}(\Sigma)$ . Obviously, this claim would still hold if ontology consistency has been maintained throughout its life cycle. In saying so, we are aware that extra work is needed to support it. Through the development of a particular agent, i.e. *Checking Agent (CA)* in [11], we now have better understanding of two functions (i.e. `make_decision` and `modify`) presented in Sect. 5.2. Briefly, a randomly chosen agent (i.e. Retailer in this example) will make an initial offer first (with Pseudocodes E, D, C), then other participating agents (i.e. different service providers) will provide possible feedback to this offering agent (with Pseudocode A). Again, this agent will check the feedback from other involved agents (with Pseudocode B). In Pseudocodes B `check_initial_offer`, function `make_decision` will invoke *CA* to check ontology consistency once there is any newly generated  $\varphi$  (e.g.  $\varphi$  in  $a_1, a_2$ , etc.), while function `modify` will modify *change\_list* to reflect which ontology modification operation (i.e. `inserting`, `deleting` and `updating`) is appropriate under certain circumstance. Again, take the above scenario for example, arguments  $a_4, a_5$  and  $a_7$  involve inserting operations, while  $a_6$  leads to a deleting operation. The refinement of a participating ontology is thus obtained by this agent engaging in the feedback-driven negotiation process. Consequently, *RAs* would refine their ontologies accordingly via negotiation.

## 6.2 Discussion

In Sect. 5, we mentioned that the agent in turn keeps track of agents’ feedbacks. That is to say, agents need to maintain linkages with other agents in case other agents need to access the recorded information of the previous round of negotiation. It seems that it is a heavy burden for agents to have this kind of one-to-one linkage. In fact, the number of agents which are involved in a business scenario and at the same time with a direct linkage with one another is less than expected. In this sense, the computation complexity can be reduced.

On the other hand, agents engaged in negotiation are keen to choose “good” strategies to maximise their utilities respectively. However, scoring criteria vary from circumstance to circumstance. The negotiation algorithm is neither just simply Tit-For-Tat (against its opponents on all rounds), nor just reciprocal (reciprocating whatever its opponent did on the previous round). In order to prevent agents from opting (e.g. `opt out`) or responding nothing (e.g. `no`), suggesting useful offers from the viewpoint of the agent which makes an initial offer in our case is desirable. Some heuristic

algorithms, for example the scoring system, may need to be developed to direct the negotiation to the defined objective for the reason that in negotiation, there is no absolute opinion of *for* and *against*.

High quality of the negotiated result is most likely to be achieved only after the above issue has been resolved. The convergence in ontology evolution is becoming decidable if participating agents would take every opportunity to improve their performance in negotiation.

## 7 Conclusions and future work

In this paper, our general focus has been on developing flexible refinement mechanisms. The mechanisms facilitate agents with their negotiation strategy to deal with inconsistency involved in understanding concepts. By doing so, a conformance to the change can be achieved and individual ontologies can be refined accordingly. We argue that deploying agents' interactions at run-time is well suited to substantially support and reflect changing service application environments. These mechanisms also make it possible for agents to adapt themselves to changes through the interaction since the interaction between agents depicts ontology evolution as well. Each agent's strategy profile is determined by the feedback from the environment and its current local global view. We have highlighted the motivation for using an ontology and how an ontology innovatively adapts in accordance with ontology evolution. In addition, we have improved negotiation strategies to reflect the dynamic changes in the evolving cycle. A real world case has been discussed with the proposed approach.

Although the approach in this paper is a promising solution to run-time ontology refinement, there are some issues which need to be addressed further. We address interactions between agents predominately so that agents are most likely to be able to rationally conform to the proposed strategies in order to achieve a particular goal. However, limited resources may lead to unexpected scenarios. For instance, the agents, with individual objectives, may respond nothing to the offer. It will thus have had quite an impact on ontology evolution. On the other hand, an opinion of "yes to all" from the agents did nothing better to soften the impact of the problem. Further work is necessary to improve negotiation agents' performance. We are also planing to include evaluating the performance of refinement especially for large ontologies in our future work.

**Acknowledgements** Work reported here is partly supported by Swinburne VC's Strategic Research Initiative Grant for project "Internet-based e-business ventures". This paper is based on an earlier published version [9]. Implementation details of the proposed approach can be found in [10]. The authors are grateful for the constructive comments

of the anonymous reviewers and Richard Moore's effort on English correction.

## References

1. Bauer B, Müller JP, Odell J (2001) Agent UML: a formalism for specifying multiagent interaction. *Int J Softw Eng Knowl Eng* 11(3):207–230
2. Bergamaschi S, Guerra F, Vincini M (2005) Critical analysis of the emerging ontology languages and standards, d1.r1, 2005. [http://www.dbgroup.unimo.it/wisdom/deliverables/fase\\_1/d1r1.pdf](http://www.dbgroup.unimo.it/wisdom/deliverables/fase_1/d1r1.pdf)
3. Haworth, Refurnishing the Supply Chain. <http://www.computerworld.com/softwaretopics/erp/story/0,10801,93607,00.html>. Accessed 15/06/07 June 2007
4. Klein M, Fensel D (2001) Ontology versioning on the Semantic Web. In Proceedings of international semantic web working symposium, California pp 75–91
5. Klein M, Noy NF (2003) A component-based framework for ontology evolution. Proceedings of the IJCAI'03 workshop: ontologies and distributed systems, international joint conference on artificial intelligence 2003 (IJCAI'03), Acapulco <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-71/>
6. Kraus S (2001) Automated negotiation and decision making in multiagent environment. Proc of Advanced Course Artificial Intell (ACAI 2001), LNAI 2086, pp 150–172
7. Laudon K, Laudon J (2006) Management information systems: managing the digital firm, 9th edn. Prentice-Hall, Englewood Cliffs
8. Lee LL, Billington C (1995) The evolution of supply-chain-management models and practice at Hewlett-Packard. *Interfaces* 25(5):42–63
9. Li L, Yang Y, Wu B (2005) Agent-based approach towards ontology refinement in virtual enterprises. In: Proceedings of the 3rd international conference on active media technology (AMT 2005), pp 220–225, Kagawa
10. Li L (2006) Agent-based ontology management towards interoperability, PhD thesis, Swinburne University of Technology, <http://www.it.swin.edu.au/personal/lii/thesis.pdf>
11. Li L, Yang Y (2008) Agent-based ontology mapping and integration towards interoperability. In: expert systems: the journal of knowledge engineering. Blackwell, Oxford (in press)
12. Maedche A, Motik B, Stojanovic L, Studer R, Volz R (2003) An Infrastructure for searching, reusing and evolving distributed ontologies. In: Proceedings of international conference of WWW (WWW2003), Budapest, pp 439–448
13. McGuinness D (2000) Conceptual modelling for distributed ontology environments. In: Proceedings of International Conference on Conceptual Structures: Logical, Linguistic, and Computational Issues (ICCS 2000), Darmstadt, pp 100–112
14. Menzies T (1999) Knowledge maintenance: The State of the Art. *Knowl Eng Rev* 14(1):1–46
15. Noy NF, Klein M (2004) Ontology evolution: not the same as schema evolution. *Knowl Inf Syst* 6(4):428–440
16. Parsons S, Sierra CA, Jennings NR (1998) Agents that reason and negotiate by arguing. *J Log Comput* 8(3):261–292
17. Papazoglou MP (2001) Agent-oriented technology in support of e-business. *Commun ACM* 44(4):71–76
18. P&G, Agents of Change. <http://www.computerworld.com/printthis/2003/0,4814,77855,00.html>. Accessed on 10 July 2007
19. Stojanovic L, Maedche A, Motik B, Stojanovic N (2002) User-driven ontology evolution management. In: Proceedings of the 13th european conference on knowledge engineering and knowledge management (EKAW 2002) Madrid, Spain pp 285–300

20. Sycara KP (1989) Multiagent Compromise via negotiation, Distributed artificial intelligence. In: Gasser L, Huhns M (eds) vol 2. Pitman, London and Morgan Kaufmann, San Mateo, pp 119–138
21. Tamma V, Bench-Capon T (2001) A conceptual model to facilitate knowledge sharing in multi-agent systems. In: Proceedings of Ontologies in agent systems (OAS 2001), Montreal, pp 69–76
22. Rubinstein A (1982) Perfect equilibrium in a bargaining model. *Econometrica* 50(1):97–109
23. Wooldridge M (2002) An introduction to multiAgent systems. Wiley, London