Distributed multi-agent communication system based on dynamic ontology mapping

Sally M. El-Ghamrawy* and Ali I. El-Desouky

Department of Computers and Systems,
Faculty of Engineering,
Mansoura University,
Mansoura, Egypt
E-mail: Sally@mans.edu.eg
E-mail: ali_eldesouky@yahoo.com
*Corresponding author

Abstract: Communication is the most important feature for meaningful interaction among agents in distributed multi-agent systems. Communication enables agent’s interaction to achieve their goals. Agent communication languages provide a standard in the protocol and language used in the communication, but cannot provide a standard in ontology, because ontology depends on the subject and concept of the communication. This lack of standardisation is known as interoperability problem. In order to obtain semantic interoperability, agents need to agree on the basis of different ontologies. In this paper, a communication layer is proposed to outline the communication between agents, multiplatform communication system (MPCS) architecture is proposed to provide a highly flexible and scalable system. In addition a dynamic ontology mapping system for agent communication (DOMAC) is proposed based on different mapping approaches.

Keywords: agent communication language; ACL; interoperability; knowledge query and manipulation language; KQML; multi-agent system; MAS; ontology mapping; distributed intelligent system.


Biographical notes: Sally M. El-Ghamrawy is an Assistant Teacher in MET Academy in Egypt. She received her MA in Computer Science from the Computers Engineering and Systems Department in 2006 and is currently a PhD candidate in the Department Computers Engineering and Systems at the Faculty of Engineering, Mansoura University. She has taught and gave a practical training in the grants from the Ministry of Communications and Information Technology in collaboration with IBM. She received a certificate A+ International Inc. CompTIA. She is also an IEEE Member.

Ali I. El-Desouky is a Full Professor for Computers Engineering and Systems Department at the Faculty of Engineering, Mansoura University in Egypt. He is also a Visiting Part-time Professor for MET Academy. He obtained his MA and PhD from the University of Glasgow in, the USA. He teaches in American and Mansoura universities. He also took over many positions of leadership and supervision of many scientific papers. He has published more than 150 articles in well-known international journals.
1 Introduction

The proposed distributed multi-agent system (DMAS) framework (El-Desouky and El-Ghamrawy, 2008) provides the basis for an open environment where agents interact with each other to reach their individual or shared goals in evolving environment. To interact in such environment, agents need to overcome many challenges. One of the most important challenges that the agents must overcome is how they must be able to communicate with each other. So the development in agent communication module must be considered in designing the DMAIS in order to give agents the ability to have successful cooperation, negotiation, and scheduling between each other. In other words, the communication is the kernel of any MAS; without communication there would not be any interaction between agents. Agent framework is a set of programming tools for constructing agents and its infrastructure provides regulations that agents follow to communicate and to understand each other, thereby, enabling knowledge sharing. Agent infrastructures mostly deal with the communication among agents based on a communication language using common ontological system. Communication is the most important feature for meaningful interaction among agents in MAS; as it enables agents to interact and share information to perform tasks to achieve their goals. In order to achieve this objective, agent communication languages (ACL) has been proposed based on the speech-act theory. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions (Farooq, 2002). ACL provides a standard in the protocol and language used in the communication, but cannot provide a standard in ontology, because ontology depends on the subject and concept of the communication, and it is almost impossible for two agents can share a same semantic vocabulary, they usually have a heterogeneous private vocabulary defined in their own private ontology. The development of generally accepted standards will take a long time (Wang and Hongshuai, 2010). This lack of standardisation is known as interoperability problem. In order to obtain semantic interoperability in DMAIS, agents need to agree on the basis of different ontologies.

In this paper, our main concern is to develop the communication module that helps in the improvement of DMAIS performance. In brief, the organisation of the paper is as follows: in Section 2, the related work of communication in MAS is reviewed, and then the definition of ontology is introduced, also an outline of the researches uses ontology is presented. The concept of ontology mapping is discussed, showing the different approaches proposed to solve the ontology mapping problem and some comprehensive surveys of some famous ontology mapping systems were introduced too, finally an example of ontology mapping between agents in MAS is illustrated. In Section 3, a communication layers is proposed to outline the communication between agents. In Section 4, multiplatform communication system (MPCS) architecture is proposed to provide a highly flexible and scalable system that allows agents written in different languages to send and receive messages using the KQML standard, as well as it allow agents to maintain several dialogues (Ds) at a time. In Section 5, a dynamic ontology mapping system for agent communication (DOMAC) is proposed based on different mapping approaches, in order to provide help in the conversation among different agents. Finally, Section 6 shows the experimental evaluation and the results obtained after implementing the proposed systems. Section 7 summarises major contribution of the paper and proposes the topics for future research.
2 Related work for communication in MAS

The communication in MAS has been subject of interests for many researches (Sycara et al., 1996; Suarez-Romero et al., 2005; Mellah, 2007), because communication is one of the most important issues in MAS design. The communication module in any MAS is responsible of how the agent communicates with other agents using an ACL. There are a few common ACLs such as the knowledge query and manipulation language (KQML) (Finin and et al., 1994) and the Foundation for Intelligent Physical Agent’s communication language (FIPA’s ACL) (Labrou et al., 1999). The exchanging of data between agents is vitally important to the efficiency of MAS. Communication is required to ensure cooperation between agents. Each agent’s actions depend critically on knowledge that is accessible only from another agent. Researchers investigating ACLs mention three key elements to achieve multi-agent interaction (Labrou et al., 1999):

1. a common ACL and protocol
2. a common format for the content of communication: content representation language
3. a common ontology.

There are several definitions of ontology have been introduced (Weiss, 1999; Noy and McGuinness, 2001). Some of them have been used and some of them are contradictory. A definition accepted in the MAS area, says that an ontology is a formal representation of concepts, characteristics and relations in each specific domain, allowing the common agreement of the people and software agents, and enabling a machine to use the knowledge of some application, multiple machines to share knowledge and still enabling the knowledge reuse. Ontologies play a key role in communication in distributed MAS, because they can provide and define a shared vocabulary about a definition of the world and terms used in agent communication. Ontology mapping is a primary problem that has to be solved in order to allow agents with different backgrounds to adjust themselves before starting any form of cooperation or communication. Using a common ontology is impractical, because it would result in assuming a standard communication vocabulary and it does not take into account the conceptual requirements of agents that could appear in future (Trojahn et al., 2008). In order to reach interoperability, two problems must be dealt with, namely: structural heterogeneity and semantic heterogeneity (Wache et al., 2001).

Several projects have used ontologies in agent-based systems. Tian and Cao (2005) presents an ontology-driven multi-agent architecture that supports the sharing and reuse among different types of knowledge acquisition agents. Other projects use ontologies to describe agent behaviour. Laclavik et al. (2006) has developed an architecture using a semantic knowledge model to define the behaviour of agents. Şandru et al. (2005) proposes a generic multi-agent task-oriented architecture based on a formal model described using the unified problem solving method description language (UPML) (Fensel et al., 2003). Gómez et al. (2001) also describes a UPML-based framework to build information agents by reusing a library of domain-independent problem solving components. Hajnal et al. (2007) developed software to generate JADE agent code from ontology-based descriptions for the K4Care system (K4CARE, 2008). There are different approaches have been proposed to solve the ontology mapping problem and some comprehensive surveys of some famous ontology mapping systems were introduced too, such as GLUE (Doan et al., 2002), QOM (Ehrig and Staab, 2004), PROMPT (Noy and
The most common systems that participated in ontology alignment evaluation initiative (OAEI) campaign are: Falcon-AO (Qu et al., 2006) is a similarity-based generic ontology mapping system. It consists of three elementary matchers, i.e., V-Doc, I-Sub, and GMO, and one ontology partitioner, PBM. V-Doc constructs a virtual document for each URIref, and then measures their similarity in a vector space model. RiMOM (Tang et al., 2006) is a general ontology mapping system based on Bayesian decision theory. It utilises normalisation and NLP techniques and integrates multiple strategies for ontology mapping. LILY (Wang and Xu, 2007) is a generic ontology mapping system based on the extraction of semantic sub-graph. It exploits both linguistic and structural information in semantic sub-graphs to generate initial alignments. Then a subsequent similarity propagation strategy is applied to produce more alignments if necessary. ASMOV (Jean-Mary and Kabuka, 2007) is an automated ontology mapping tool that iteratively calculates the similarity between concepts in ontologies by analysing four features such as textual description and structure information. It then combines the measures of these four features using a weighted sum. The weights are adjusted based on some static rules. PRIOR+ (Ming et al., 2008) introduces a new generic ontology mapping approach; the approach measures both the linguistic and the structural similarities of the ontologies. More specifically, three kinds of similarity, i.e., edit distance-based similarity, profile similarity and structural similarity, are calculated.

Figure 1 The proposed communication layers (see online version for colours)

3 The proposed agent communication layers

The process of communication between agents in DMAIS is divided into five main layers, each layer provides information needed to the layer above it, and receives information from the layer below it, the proposed five layers is shown in Figure 1. The communication layers are constituted based on the message transport protocol [transmission control protocol/internet protocol (TCP/IP)] extend it by the network
infrastructure layer, ACL, content language and the ontology layer. Each layer is illustrated below:

3.1 Message transport protocol layer

Message Transport layer is the lowest layer in the proposed agent communication layers. Since data transmission from source machines to destination machines through networks is realised in this layer, so this layer can be recognised as the transport service provider. This layer is the physical world consisting of agent host machines. There are many protocols that can be used as a standard for the network transport service provider, namely TCP/IP, UDP, HTTP, FTP, IIOP, RMI, and SMTP. Hence, the communication between agents, which located on distributed machines, must be constituted by some kind of network protocol. In the proposed agent communication layers, the TCP/IP is used.

3.2 Network infrastructure layer

The network infrastructure layer above the message transport layer plays a vital role in connecting the agent layers (the logical layer) with the network layers (the physical layer), starting by the message transport layer. Such connection is guaranteed and realised by two well-defined interfaces, one between the agent layers and this infrastructure, and the other between this infrastructure and the network layer (message transport layer). Most of agents are running on different machines and need to communicate and exchange different kinds of data over the networks. If the DMAIS is directly built on the distributed network, this makes the communication between agents has great overhead, cost and time consuming. So, this layer is proposed to solve the problem that might occur by building this interface that makes the agents not aware of the physical-network issues. Once this layer has been defined, agents can send and receive messages to/from any other agent without concern about network issues. So, when the messages are sent from agents to any message transport layer of a specific agent, they should be delivered to their destination without further interaction with agents. This raises the need to provide two fundamental services agent name resolution and agent message delivery in this layer. Finally, this layer provides transparent support for network communication between agents. This transparency makes the agents deals with the problem of when and with whom to interact only, and leaves the problem of how to interact to the network layer (message transport layer). In this way, the agent layers (the logical layer) and the network layer (the physical layer) can be independently implemented.

3.3 ACL layer

To establishing a communication between any two agents, they communicate by languages. At present there are two mainstreams in communication languages. One is FIPA-ACL (Labrou et al., 1999) proposed by European FIPA institute., the other is KQML [7] proposed by knowledge share effort (KSE) research group of American DARPA, and based on the linguistic theory of the speech act. To express communication between agents in the ACL layer, the KQML is used. The KQML is a high-level communication language and protocol for exchanging information and sharing
knowledge, which provides the basic format of expressing and processing messages and supports sharing information among agents (Farooq, 2002), it was conceived both as a message format and a message handling protocol to support run-time knowledge sharing among agents.

3.3.1 Knowledge query and manipulation language

In this layer, KQML language is chosen to be the internal format of the agent’s messages, and then this message will be translated to any other language according to the destination agent. The main advantages that motivate us to choose the KQML language:

1. KQML messages are declarative, simple, readable and extensible.
2. Since it has a layered structure and KQML messages are unaware of the content of the message they carry, KQML can easily be integrated with other system components.
3. KQML is independent of the network transport mechanisms (TCP/IP, SMTP, IIOP, etc.) and the content languages (SQL, PROLOG, SL, etc.).
4. KQML has the potential to enhance the capabilities and functionality of large scale integration and interoperability efforts in communication and information technology.

Any KQML message structure consists of three layers: content, message and communication layers.

- **The content layer**: It contains the actual content of the message specified in any language. KQML supports for ASCII language and binary code symbols.

- **The message layer**: It is the core of KQML. Its basic function is a protocol to identify the protocol used to send a message. In addition, it also includes optional part described content information, such as languages, communication theme descriptor. So that, KQML can analyse, route and deliver the content.

- **The communication layer**: It realises news features, uses specific identifier to mark parameters of low-level message; it consists of low-level communication parameters, such as sender, receiver, and message identities.

3.3.2 The speech act theory

The Artificial Intelligence researchers exploited the speech act theory to model communication between software agents. Austin suggests that the role of languages in communication is to impart actions (Labrou and Finin, 1996). Speakers do not simply utter sentences that are true or false, but rather perform speech actions such as requests and suggestions. Consequently, all utterances are speech acts, i.e., they are actions of some sort. The speech act theory (Labrou and Finin, 1996; Searle et al., 1980) considers three aspects of utterances. **Locution** refers to the act of utterance itself. **Illocution** refers to the ‘type’ of utterance as in a request to turn on the heater. It conveys the speaker’s intentions to the listener. **Perlocution** refers to the effect of an utterance, i.e., how it influences the recipient.
3.4 The content layer

Above the agent communication layer, there is the content language layer that contains the actual information of a message. Different content languages within a single agent or MAS can be used like semantic language (SL), SQL, PROLOG or any other representation means. In this layer, knowledge interchange format (KIF) is used.

3.4.1 Knowledge interchange format

The KIF is used in this platform, it is a general purpose content language developed in knowledge sharing effort. The Interlingua Group is developing a common language for expressing the content of a knowledge-base. This group has published a specification document describing the KIF (Genesereth and Fikes, 1992). KIF was chosen for many reasons:

1. It is a recognised representation format for agent ontologies.
2. It is extremely expressive language, supporting full first-order logic. The expressiveness of KIF allows the exchange of information with arbitrary complexity and with variable degrees of completeness.
3. The semantics of sentences written in KIF is declarative and interpreter independent, so there is no possible ambiguity exists in the information exchanged (indirectly) between agents.
4. Defining a mapping from KIF to XML is more acceptable.
5. It can be used to support translation from one content language to another, or as a common content language between two agents that use different native representation languages.

3.4.2 Dialogues

After defining the content language layer that contains the actual message, these messages between agents are grouped into Ds. A D must be established first, when any agent wants to communicate with another agent. The benefit of using these Ds in the proposed communication layers is that when an agent wants to communicate with two or more agents in the same time, the agent can maintain several Ds at a time with the same or with different receiving agents.

3.5 The ontology layer

The ontology layer is used to define a common vocabulary for agents to communicate with one another, it is used to represent the content in the messages exchanged among agents to reduce the conceptual and terminological confusion that often appear among different people and organisations. Ontology determines the semantics of the concepts used in the content language. The actual meaning of the message content is captured in the ontology layer. This layer gives detailed definitions of the syntax and the semantics of the message, these definitions is vital in the ontology mapping process. To enable mapping of different ontologies, the ontology layer must provide an explicit description of ontologies in a way that is understandable to agents. So when agents want to
communicate, they have to share the ontology used for communication by using this layer. This means that this layer in each agent must do the following:

1. specify the conceptualisation of the domain they operate on
2. share the specification of this conceptualisation by recognising the same objects and the same relations among the objects in the domain they operate on
3. use the same language and the same words for describing the same objects
4. define how to model this domain together with possible restrictions.

The concept of ontology is used for the formalisation of knowledge in terms of classes, properties, instances and the relations. There are many ontology languages proposed as a formal language to encode the ontology, such as, resource description framework (RDF) (Lassila and Swick, 1999), RDF schema (RDFS) (http://www.w3.org/TR/1998/WD-rdf-schema-19980409/), ontology inference layer (OIL) (Fensel et al., 2001), DAML+OIL (2001), or web ontology language (OWL) [35]. The development of the most used ontology languages is shown in Figure 2. In this layer, the web ontology language OWL is used.

Figure 2  The development of the ontology languages (see online version for colours)

3.5.1 The web ontology language

The OWL (McGuinness and Van Harmelen, 2004) is a language used to describe ontologies in a form of classes and relations among them together with further restrictions and intended use of them. It is designed primarily for the WWW documents and applications by W3C (http://www.W3C.org); it is characterised by formal semantics and RDF/XML-based serialisations for the semantic web, but it can be used for any other domain as well. OWL is built on RDF/RDFS and uses the XML schema constructs (Thompson et al., 2001). It is not simply a language for a message format, like XML language, but it is a language intended for knowledge and ontology representation. OWL is chosen to be the language used to represent the ontologies in the proposed ontology layer for bunch of reasons:

1. The OWL is going to be standardised by W3C provides an ontology modeling language with defined formal semantics.
2. OWL provides a standard way to explicitly express the semantics. XML itself does not provide any explicit description of intended use of data.
3. By explicitly expressing the semantics this can lead to separating the semantics from the code and thus leads to writing less code for the implementation of the
communicating agents. OWL provides a common language to define semantics so that anyone can understand it.

4 OWL uses the XML syntax and builds on RDF syntax and semantics, i.e., it builds on existing widely used technologies so that common XML and RDF tools can work with OWL as well.

4 The proposed MPCS architecture

In the proposed DMAIS framework (El-Desouky and El-Ghamrawy, 2008), there are eight main modules that make the agents have the ability to interact and coordinate with each other. The most critical module and the kernel of DMAIS is the communication module; it is responsible for all the interactions between agents, as well as enables communication between other modules in DMAIS, by means of message passing. It plays an essential role for agents to exchange information and to coordinate their actions. In this sense, a communication module in DMAIS is proposed to control the communication process in DMAIS. First a MPCS is designed as a modular architecture, as shown in Figure 3, that permits flexibility, scalability and interoperability, in which it allow the system to be more extensible.

Figure 3 The proposed MPCS (see online version for colours)

MPCS has been designed with a decentralised architecture, this done by distributing the functions of the system among different interchangeable modules based on using separate communication layers for each agent, as showed in Figure 3, this leads to ensure the efficiency of the system by avoiding the bottlenecks that might occurs in using centralised architecture. The agents involved in the communication may be local to a single Platform or on different platforms. Two modes of communication are involved for message delivery in MPCS which includes local and global communication. There are several advantages of MPCS platform:

1 First, allowing agents written in different languages to send and receive messages using the KQML standard.
Second, provide a highly flexible and scalable system which support large number of agents to be loaded. In this way, agents developed in any other platforms can communicate, providing that the necessary modules have been implemented.

MPCS has the property of distributed architecture, as its functions are distributed among different interchangeable modules. This distributed feature, thanks to the network transparency, is naturally and easily obtained.

Furthermore, MPCS has reliable and fault tolerant features, and it can be easily be developed.

The core modules of MPCS distributed on multiple machines. This ensures that the failure of one machine will not cause the whole system to come down and does not affect the agent system working on its current machines.

These advantages have been achieved through the use of interchangeable modules as shown in Figure 3, which ensure efficiency and avoids bottlenecks by distributing the function of the system among different modules. MPCS has three main phases, that groups the modules and sub-modules that has direct interactions with each other to achieve a specific task. The three modules are illustrated in the next sub sections.

4.1 Initiation phase

This phase contains the modules that are responsible for the interaction and registrations of agents and it contains the central unit of control that distributes the tasks to the modules that can accomplish it.

- The task distributor (TD) (central unit of control) is the core of the system. This module ultimate goal is to distribute the main tasks of the platform to the appropriate modules. It also creates a list of all the agents that are using the system.
- The agent registration (AR) module responsible for the registration of all the agents connected to the system at a given time. Any agent that wants to communicate with other agents needs to register in the system through this module.
- Agents interact directly through agent interface (AI). So, the communication platform is separated from the agents, in order to simplify the management of the platform. The Interface has been designed to provide a dynamic interface to the programmer to utilise the features of AR module.

In this phase, a library of interaction protocols has been provided allowing MPCS agents to communicate based on KQML specifications.

4.2 Handling phase

Handling phase contains the modules that are responsible for handling the messages between agents. The messages between agents are grouped into Ds. If an agent wants to communicate with another agent, it needs first to create a D with it, to which all subsequent messages between both are sent/received. An agent can maintain several Ds at a time with the same or with different receiving agents. The main goal of the MPCS is to provide a high-level management to the Ds between agents registered in the system to ensure the delivery of messages between them.
Distributed multi-agent communication system

• Each agent has a dialogue master (DM) module to manage all these Ds. Once this module registered in the system, it is automatically created. This module, with cooperation of others, responsible for the maintenance of the agent’s Ds, creating new Ds or sending/receiving messages within a D.

The DM can potentially receive a large number of D requests. To avoid bottlenecks this module distributes its work among D sub-masters modules.

• A dialogue sub-master module exists for each pair of sending and receiving agents, responsible for creating Ds and assigning messages to them.

• The policy manager (PM) module: its goal is to check whether a message is allowed in a given D. It can permit any message sequence within a D or limit it to a specific course or several action courses. For example, it might be restrict the communication between specific agents to a simple question/answer D, while other agents might be allowed free communication. This can be achieved by using the library of the interaction protocols (FIPA Interaction Protocol Library, 2001), the protocols may range from simple query and request protocols, to more complex ones.

For this reason, the PM is a completely independent module that can adapt to user requirements.

4.3 Processing phase

Processing phase contains the modules that is responsible for sending the message to a specific agent by checking its address if it’s in different platform other than the sending agent’s platform. And if the message received written in different language, this phase is responsible for translating this message.

• A message distributor (MD) module, similar to the DM, is responsible for distributing the messages, avoiding a possible bottleneck by delegating the request to send a message to a message sub-distributor (MSD).

• This MSD is responsible for processing all the messages that belong to the same destination platform. The system will contain as many MSDs as platforms with which any communication is maintained.

• The message translator (MT) module serialises the message from the internal format of the system to the format of the destination platform. Its main goal is to detect the communication language used by the sender agent, if the language used is different from the receiver agent, or vice versa, then a translation between those different languages must be done with cooperation with the ontology manager module. The data flow diagram for the translation approach is shown in Figure 4.

• The transport protocol manager (TPM) module is defined in order to send/receive messages to/from other platforms. This module is used depending on the destination platform. There is an ability to insert new transport protocols, by separation of the code in an independent module, in order to allow different platforms to be able to communicate with each others. External agents can either use the implemented default transport protocol, or another protocol. The default protocol can be used to modify the necessary communication parameters in order to produce a new protocol.
The ontology manager module: stores this information about the ontology and provides the facilities that system administrators need to set up and evolve the Ontology. Additionally, it provides means for defining mappings between autonomous ontologies.

This module uses the DOMAC system proposed in the next section to perform the mappings between different ontologies. The ontology manager has two main roles: First, it distributes copies of ontologies to requesting agents. Second, it informs committing agents of changes in ontology. The ontology manger module manages the whole Ds trying to help when it is needed, it provides some services:

1. the ability to translate expressions between two different ontologies
2. learn with the ontology services already provided so that it could use this information in a future negotiation
3. the capability for defining, modifying and deleting expressions and ontology definitions.

The communication language we are using to implement the module is KQML. The ontology manager module includes a basic domain ontologies represented in OWL.

Figure 4 The data flow diagram for the MT module (see online version for colours)
4.4 How the message is handled in MPCS

A description of how message is transmitted is represented in Figure 5; this example is used for better understanding of the function of the proposed platform. The sending agent must register first and also that the corresponding D must be previously created. First, the sending agent indicates to AI that it wants to send a message to another agent through a D. Then, the AI delegates the message transmission to the DM, which in turns assigns the message to the dialogue sub-manager (DSM).

**Figure 5** Sending local or global message in MPCS (see online version for colours)
Both agents may maintain several Ds at the same time, so the sending agent must inform the DSM of the D to which the message is to be delegated. Then, the D verifies the message state with the help of the PM. If the state of the message makes semantic sense then the D communicates with the TD to locate the receiving agent. When the TD receives the outgoing message, it checks the agent’s location. If the sending and the receiving agents are in the same platform then it will send the message without the help of MD, otherwise if the receiving agent is in an external platform, then the TD delegates the outgoing message transmission to the MD, which will search the MSD corresponding to the destination platform.

Figure 6 The interaction messages between MD, MSD, AR and MT modules (see online version for colours)

If the MSD does not exist, it is created at this point, with the necessary parameters to establish the communication obtained from the AR module. Once the MSD has been located, it is assigned with the message to be sent, and in turn assigns the message to the appropriate MT. The interaction between these four modules, namely, MD, MSD, AR, and MT, is shown in Figure 6. The message is translated into the format acceptable to the destination platform, as showed in Figure 5. Then it will be sent by the MSD, then to the
TPM used for communication with the given platform. Finally, the TPM sends the outgoing message to the agent in the other platform. A preliminary experiment is then conducted in Section 6, indicating that MPCS has the scalability advantage, as it behaves efficiently under full-load conditions comparing to recent systems.

5 The proposed DOMAC

Ontology mapping takes two ontologies as input and creates a semantic correspondence between the entities in the two input ontologies. The ontology manager, described in the previous section, will monitor and help the communication process at the moment when it is happening, without having to do a mapping of all the ontologies involved. As shown in Figure 7, there are two ontologies (Ontology1 and Ontology2) belonging to two agents Agent1 and Agent2, respectively. Suppose that Agent1 need some information to complete its task, and it knows that the information is probably available in a database managed by Agent2. As a result, Agent1 sends a message to Agent2; the message is translated and sent to Agent2.

Figure 7 Ontology mapping between two agents (see online version for colours)

Then both agents must detect first, if both use the same ontology or whether use different ontologies. If they use different ontologies and if no mapping is known, the agents should try to establish a mapping. Then alignment between ontology1 and ontology2 is established to generate a link between them and send this link to Agent2 to manage it in understanding Agent1 message. In this sense, there must be an ontology mapping algorithm used in the ontology manager proposed in the MPCS. As a result, a DOMAC is proposed to show agents how to establish a mapping between two ontologies. The DOMAC is shown in Figure 8; its main goal is to map different ontologies. The input of DOMAC is two ontologies, O1 and O2, stored in ontologies repository, expressed in the form of formal taxonomies or ontologies, the language used for describing the ontologies is the OWL. The output is a mapping, also called the mapping result, between the input taxonomies or ontologies. Mapping can be represented in different ways depending on its use. For example, mappings can be represented as queries, bridging axioms or an instance in a mapping ontology.
5.1 The input phase

The input to the DOMAC is the heterogeneous ontologies stored in the Ontologies Repository, and these different ontologies are going to be mapped by the DOMAC system. The ontologies stored in this repository expressed in OWL language, OWL is built on RDF/RDFS and uses the XML Schema constructs. The repository built by three main ways:

1. Downloading the ontologies from the ontology libraries. The protégé ontology library (Grosso et al., 1999) is used; it is a free, open source ontology editor and knowledge base framework. The protégé-OWL editor is an extension of Protégé that supports the owl. It enables users to: load and save OWL and RDF ontologies.

2. The translated ACL messages to OWL ontologies by the MT module in the MPCS, as showed in Figure 3.

3. The messages sent by external agents in form of OWL ontologies to the MPCS.

Figure 8  Dynamic ontology mapping system for agent communication (see online version for colours)

5.2 The DOMAC processing phase

There are four main modules in the processing phase of DOMAC, the first module is the Parsing Module, and its main goal is to deal with the OWL ontologies stored in the ontologies repository. First the XML converter converts the OWL message into ontologically annotated XML document (i.e., The content of the OWL message have been encoded to XML document), this is because parsing the OWL messages is a big overhead in agent development and XML encoding is easier to develop parsers as anyone can use off-the-shelf XML parsers. The XML-encoding enhances the canonical syntactic encoding. Then the XML ontologies will be parsed and pre-processed by removing stop words, stemming, and tokenising. Then the parsing module sends the parsed document to the similarity computation module, which measures three kinds of similarity: edit distance similarity, cosine similarity and structural similarity.
• The *edit distance-based similarity* (Mao et al., 2007) is calculated between the names of elements based on their Levenshtein distance. The similarity is defined as:

\[
\text{NameSim}(e_{1i}, e_{2j}) = 1 - \frac{\text{EditDist}(e_{1i}, e_{2j})}{\max(l(e_{1i}), l(e_{2j}))}
\]  

(1)

where the \(\text{EditDist}(e_{1i}, e_{2j})\) is the Levenshtein distance between elements \(e_{1i}\) and \(e_{2j}\), and \(l(e_{1i})\) and \(l(e_{2j})\) are the string length of the name of \(e_{1i}\) and \(e_{2j}\) respectively.

• The *structural similarity* (Mao et al., 2007) between two elements comes from their structural features (e.g., the number of direct property of a class). The structural similarity of the classes in two ontologies is defined as follows:

\[
\text{StructSim}(e_{1i}, e_{2j}) = \frac{\sum_{k=1}^{n}(1 - \text{diff}_k(e_{1i}, e_{2j}))}{n}
\]

(2)

where \(e_{1i}\) and \(e_{2j}\) are two class elements in ontology \(O_1\) and \(O_2\) respectively, \(n\) is the total number of structure features, and the \(\text{diff}_k(e_{1i}, e_{2j})\) denotes the difference for feature \(k\), and its defined as:

\[
\text{diff}(e_{1i}, e_{2j}) = \frac{|\text{sf}(e_{1i}) - \text{sf}(e_{2j})|}{\max(\text{sf}(e_{1i}), \text{sf}(e_{2j}))}
\]

(3)

where \(\text{sf}(e_{1i})\) and \(\text{sf}(e_{2j})\) denote the value of structure features of \(e_{1i}\) and \(e_{2j}\) respectively.

• *Cosinesimilarity* is a non-Euclidean distance measure between two vectors (Jiayi et al., 2008); it is a common approach to compare documents in the field of text mining. Given two feature vectors \(i\) and \(j\), the similarity score between concepts \(i\) and \(j\) is represented using the dot product as follows:

\[
\text{CosSim}(i, j) = \frac{c_i \cdot c_j}{||c_i|| \cdot ||c_j||}
\]

(4)

The resulting score is in the range of \([0, 1]\) with 1 as the highest relatedness between concepts \(i\) and \(j\).

Then for each similarity computed in the similarity computation module, harmony estimator estimated a measurement of harmony in the estimator module; it is used to provide a measurable number that can tell which similarity is more reliable and trustful so that we can give it a higher weight during aggregation. To establish the joint attention, agent1 makes an announcement containing a unique representation of a concept and instance of the concept. After agent2 receive the announcement, it investigates whether it has a concept of which an instance matches to a certain degree the communicated instance, by measuring the proportion of words that two instances have in common. The instance with the highest proportion of corresponding words, together with the communicated instance, the joint attention provided that the correspondence is high enough. Then the estimator module sends the result to the mapping module, which its main goal is to establish mapping between the primitive concepts that make up the
concept. The process of generating mapping from O1 to O2 is known as dynamic ontology mapping.

5.3 The output phase

After applying the proposed dynamic ontology mapping, the following is the form of the mapping results from agent1 to agent2, the result of this process is called Mapping:

\[ A1.\text{node.Instructor.has.firstname} \leftrightarrow A2.\text{Node.lecturer.has.name}. \]

\[ A1.\text{node.Instructor.has.Phone number} \leftrightarrow A2.\text{Node.lecturer.has.cell phone}. \]

6 Experimental evaluation

A number of experiments were performed in two stages to validate the effectiveness of the proposed MPCS and the DOMAC.

6.1 Stage I: validation of the MPCS architecture

To investigate the effects of MPCS, a MAS has been implemented to provide a testing platform. The whole system is implemented on a 5 PC’s with an Intel Pentium 4 processor at 300 GHz, with 2 GB of Ram, connected with network Ethernet 512 Mbps. A network of cooperative agents is designed, the number of agents range from 100 to 1,000, depending on the specific test. The experiments are focused on evaluating the scalability of the system with an increasing number of agents. Generally, scalability refers to how well the capacity of a system to do useful work increases as the size of the system increases. Thus, in distributed software engineering, the term ‘scalability’ is sometimes used when ‘increased environmental loading due to an increase in the number of distributed components’ is arguably more appropriate. Thus, the scalability of MAS is the average measure of the degree of performance degradation of individual agents in the society as their environmental loading, caused by an expansion in the size of the society, increases. The scalability has been achieved by using threading feature. So, any of the modules described in MPCS architecture that may act as bottlenecks are executed as a separate thread. There are two experiments are done to test the scalability in local and global communication. For each experiment, several parameters have to be specified: number of agents: It is easily seen that the number of agents is one of the most important parameters in a MAS experiment. Number of hosts: the number of hosts is limited by the available resources only. Agent platform: whether it is a local agent (in same platform) or global agent (in different platform). Computational time in milliseconds

6.1.1 Experiment 1: test scalability of local communication in MPCS

First experiment: in the local communication, when the sending and receiving agents are in the same platform, the MPCS are compared with two systems: JADE (Hajnal et al., 2007) and MOZART (Suarez-Romero et al., 2005) system, as shown in Figure 9, and the computational time for the message delivery is measured when increasing the number of agents.
6.1.2 Experiment 2: test scalability of global communication in MPCS

Second experiment: in the global communication, when the sending and receiving agents are in the different platform, MPCS is compared with JADE and MOZART system, as shown in Figure 10, and also the computational time for the message delivery is measured when increasing the number of agents.

From the figures, it is observed that the computational time for delivering a message increases with the number of agents increases, as expected, but linearly. As can be observed also, our MPCS, behaves better for both measures than JADE and Mozart systems especially when managing many threads. JADE does not scale well for simulation sizes involving a large number of agents. The major reason for this is the inefficiency of the JADE agent directory service. Because this services used frequently by the other platform services, its inefficiency affects other services too. In global communication may cause substantial delays. When a message is delivered, the JADE message transport service needs to know the receiver agent’s status (whether it is active or dead) and address (if it is on the same node or not) by accessing the directory service every time. Since the default directory service which employs LDAP has slow response behaviour, it is overwhelmed by a large number of concurrent requests (Wang et al.,
This experiment shows that our MPCS architecture has the scalability advantage, as it behaves efficiently under full-load conditions comparing to recent systems.

6.2 Stage 2: validation of the DOMAC

A number of experiments were performed to validate the effectiveness of the proposed DOMAC. In each experiment, we used the precision and recall to evaluate the experiment results which can be defined as follows:

\[
\text{Precision} : \quad P = \frac{|B \cap A|}{|A|} \\
\text{Recall} : \quad R = \frac{|B \cap A|}{|B|}
\]

In the formulas above, \( A \) presents the number of correct mappings recognised by algorithm, \( B \) presents the number of reference mappings. There is always a tradeoff between precision and recall. Therefore, \( F \)-measure is leveraged to combine both metrics. It is a weighed harmonic mean of precision and recall. In other words, it is the weighed reciprocal of the arithmetic mean of the reciprocals of precision and recall. It is computed as:

\[
F\text{-measure} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}
\]

Figure 11 The precision and recall of DOMAC, JA and KMS (see online version for colours)

6.2.1 Experiment 1: test performance of DOMAC modules

In experiment 1, we evaluated the performance of the joint attention module. First ontology1 and ontology2 were randomly generated. Taking into account, for each ontology there are 1,000 instances. Given these ontologies, the agents established a
mapping between them. Finally, the experiments were carried out for different sizes of the set of words. The precision and recall were determined for the joint attention. To evaluate the joint attention module in our DOMAC, we compare our results with the results obtained by JA (Floris and Nico, 2004) and KMS (Ruixue and Zhanhong, 2008), as shown in Figure 11. The experiment results show that our DOMAC performance is better than the system developed in JA and KMS there are two reasons for this: first Using XML document helps better address the pragmatic aspects through the use of links. Links point to additional information. Links can assist with ontological problems (defining and sharing ontologies). Links can point to agent capability and identity information, protocols, even semantics. Second, the similarity computation module (the input to joint attention module) consists of three kinds of similarity: edit distance similarity, cosine similarity and structural similarity. And those three similarity kind listed (Ming et al., 2008) to be the most effective and reliable than most of similarity methods.

6.2.2 Experiment 2: evaluate ontology mapping approach in DOMAC

In experiment 2, to evaluate our ontology mapping approach in DOMAC we use the benchmark tests from OAEI, OAEI 2008 (http://oaei.ontologymatching.org/2008/), ontology matching campaign 2008. We choose it for many reasons:

a The annual OAEI campaign has become an authoritative contest in the area of ontology mapping, and thus attracts many participants including both well-known ontology mapping systems and new entrants.

b The campaign provides uniform test cases for all participants so that the analysis and comparison between different approaches is practical.

c The ground truth of benchmark tests is open.

Thus we can use it to comprehensively evaluate different components of our approach. We concerned in this experiment with the ontology Mapping approach, so we compare ours with recent most common systems that participated in OAEI campaign. Figure 12 shows the comparison between the precision, recall and f-measure of the DOMAC and top ranked systems on benchmark tests in OAEI campaign.

Figure 12 The comparison between DOMAC and top ranked systems (see online version for colours)
7 Conclusions and future work

In order to make possible interaction between agents in DMAIS, it is necessary to have a communication platform, a communication language and an ontology mapping system. In this sense, an outline of the communication between agents has been described by mean of the proposed communication layers, a MPCS Architecture is proposed to provide a highly flexible and scalable system that allows agents written in different languages to send and receive messages using the KQML standard. A DOMAC is also proposed based on different mapping approaches. A survey of recent work in communication in MAS is reviewed; also an outline of the researches uses ontology is presented. In addition, some comprehensive surveys of some famous ontology mapping systems were introduced too. A preliminary experiment is then conducted, indicating that DOMAC can be evidently helpful to discovering semantic mappings for dynamic agent-based ontology. And other experiments are used to indicate that MPCS has the scalability advantage, as it behaves efficiently under full-load conditions comparing to recent systems. As future work, we plan to propose new interaction protocols in our architecture. In addition, we intend to present an agent negotiation model for ontology mapping.

References

Available at http://oaei.ontologymatching.org/2008/.


Distributed multi-agent communication system


