International Journal of Innovative Computing, Information and Control
Volume **8**, Number **7(A)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 4547–4564

IMPLEMENTING A DYNAMIC ONTOLOGY MAPPING APPROACH IN MULTIPLATFORM COMMUNICATION MODULE FOR DISTRIBUTED MULTI-AGENT SYSTEM

Sally M. Elghamrawy*, Ali I. Eldesouky and Ahmed I. Saleh

Computers and Systems Department
Faculty of Engineering
Mansoura University
No. 60, El Gomhoria Street, Mansoura, Egypt
*Corresponding author: Sally@mans.edu.eg

Received April 2011; revised October 2011

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 standardization is known as interoperability problem. In order to obtain semantic interoperability, agents need to agree on the basis of different ontologies. In this paper, agent communication layers are proposed to outline the communication among agents, and 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), Ontology mapping, Interoperability, KQML, Multi-agent system (MAS), Distributed intelligent system

1. **Introduction.** The proposed Distributed Multi-Agent System (DMAS) framework [1] 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 one another. 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 among one another. In other words, the communication is the kernel of any MAS; without communication there would not be any interaction among agents. Agent framework is a set of programming tools for constructing agents, and its infrastructure provides regulations that agents must follow to communicate and understand one another, 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 multi-agent systems, 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 [2]. 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 to share the 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 [3]. This lack of standardization 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 organization 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, and also an outline of the researches use 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 among agents in MAS is illustrated. In Section 3, communication layers are proposed to outline the communication process among 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 allows agents to maintain several Dialogues 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. Section 6 shows the experimental evaluation and the results obtained after implementing the proposed systems. Finally, Section 7 summarizes major contribution of the paper and proposes the topics for future research.

2. **Related Work for Communication in Multi-Agent Systems.** The communication in MAS has been subject of interests for many researches [4-6], 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 agent communication language (ACL). There are a few common ACLs such as the Knowledge Query and Manipulation Language (KQML) [7] and the Foundation for Intelligent Physical Agent's communication language (FIPA's ACL) [8]. The exchanging of data among agents is vitally important to the efficiency of MAS. Communication is required to ensure cooperation among agents. Each agent's actions depend critically on knowledge that is accessible only from another agent. Researchers investigating agent communication languages mention three key elements to achieve multi-agent interaction [8]: (1) A common Agent Communication Language (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 [9,10,11]. Some of them have been used and some of them are contradictory. A definition accepted in the multi-agent systems 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 Multi-agent System, 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 [12]. In order to reach interoperability, two problems must be dealt with, namely: structural heterogeneity and semantic heterogeneity [13]. Several projects have used ontologies in agent-based systems. [14] presents an ontology-driven multi-agent architecture that supports sharing and reusing among different types of knowledge acquisition agents. Other projects use ontologies to describe agent behavior. [15] has developed an architecture using a semantic knowledge model to define the behavior of agents. [16] proposes a generic multi-agent task-oriented architecture based on a formal model described using the Unified Problem solving Method description Language (UPML) [17]. [18] also describes a UPML-based framework to build information agents by reusing a library of domain-independent problem solving components. [19] developed software to generate JADE agent code from ontology-based descriptions for the K4Care system [20]. 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 [21], QOM [22] and PROMPT [23]. The most common systems that participated in OAEI campaign are: Falcon-AO [24] is a similarity-based generic ontology mapping system. It consists of three elementary matchers: V-Doc, I-Sub, 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 [25] is a general ontology mapping system based on Bayesian decision theory. It utilizes normalization and NLP techniques and integrates multiple strategies for ontology mapping. LILY [26] 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 [27] is an automated ontology mapping tool that iteratively calculates the similarity among concepts in ontologies by analyzing 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+ [28] 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 are calculated: edit distance based similarity, profile similarity and structural similarity.

3. **The Proposed Agent Communication Layers.** The process of communication among agents in DMAIS is divided into five main layers, each layer provides information

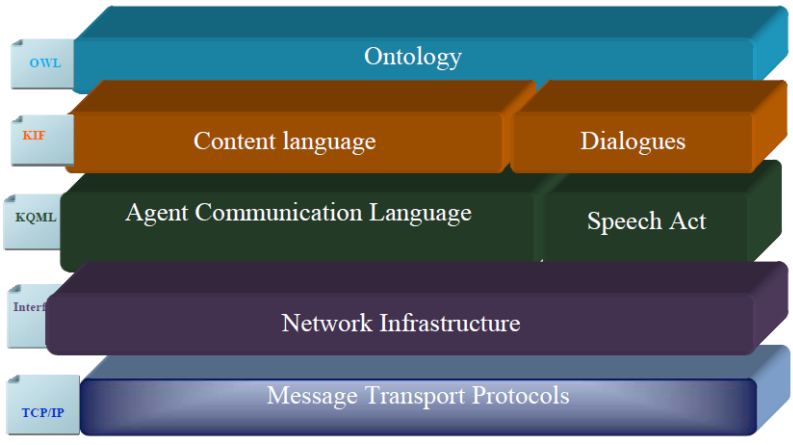

FIGURE 1. The proposed communication layers

needed to the layer above it, and receives information from the layer below it, the proposed five layers are shown in Figure 1. The communication layers are constituted based on the Message Transport Protocol (TCP/IP protocol) extend it by the network infrastructure layer, agent communication language, 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 network is realized in this layer, so this layer can be recognized 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 among agents, which located on distributed machines, must be constituted by some kind of network protocol. In the proposed agent communication layers, the Transmission Control Protocol/Internet Protocol (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 realized 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 make the communication among agents has a 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. Finally, this layer provides transparent support for network communication among agents. This transparency makes the agents deal 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. **Agent communication language layer.** To establish a communication between any two agents, they communicate by languages. At present there are two mainstreams in communication languages. One is FIPA-ACL [8] proposed by European FIPA institute, the other is KQML [7] proposed by KSE (Knowledge Share Effort) research group of American DARPA, and based on the linguistic theory of the speech act. To express communication among agents in the Agent Communication Language layer, the KQML (Knowledge Query and Manipulation Language) 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 [2], it was conceived both as a message format and a message handling protocol to support run-time knowledge sharing among agents.

•*Knowledge Query and Manipulation Language (KQML).* 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. Any KQML message

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

•*The Speech Act Theory.* The Artificial Intelligence researchers exploited the speech act theory to model communication among software agents. Austin suggests that the role of languages in communication is to impart actions [29,30]. 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.

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 Multi-Agent System can be used like Semantic Language (SL), SQL, PROLOG or any other representation means. In this layer, KIF (Knowledge Interchange Format) is used. KIF 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 [31].

•*Dialogues.* After defining the content language layer that contains the actual message, these messages among agents are grouped into Dialogues. A dialogue must be established first, when any agent wants to communicate with another agent. The benefit of using these dialogues, 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 maintains several Dialogues 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 organizations. 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. There are many ontology languages proposed as a formal language to encode the ontology, such as, Resource Description Framework (RDF) [32], RDF Schema (RDFS) [33], Ontology Inference Layer (OIL) [34], DAML+OIL [35], or Web Ontology Language (OWL) [36]. The development of the most used ontology languages are shown in Figure 2.

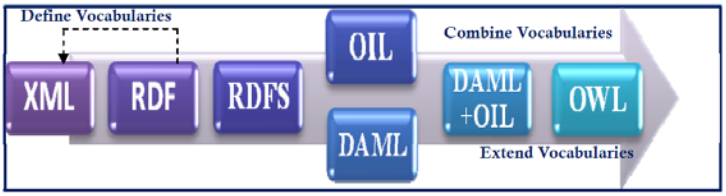

FIGURE 2. The development of the ontology languages

• *The Web Ontology Language (OWL).* The Web Ontology Language (OWL) [36] is a language used in this layer 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 [37]; it is characterized by formal semantics and RDF/XML-based serializations 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 [38]. It is not simply a language for a message format, like XML language, but it is a language intended for knowledge and ontology representation.

4. **The Proposed Multiplatform Communication System Architecture (MPCS).** In the proposed DMAIS framework [1], there are eight main modules that make the agents have the ability to interact and coordinate with one another. The most critical module and the kernel of DMAIS is the communication module; it is responsible for all the interactions among 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 Multiplatform Communication System (**MPCS**) is designed as a modular architecture, as shown in Figure 3, which permits flexibility, scalability and interoperability, in which it allows the system to be more extensible.

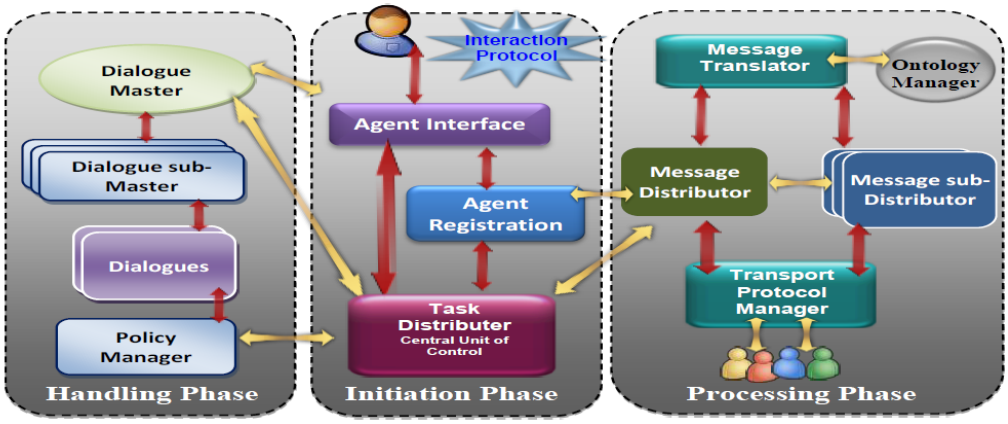

FIGURE 3. The proposed multiplatform communication system architecture

MPCS has been designed with a decentralized 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, which leads to ensure the efficiency of the system by avoiding the bottlenecks that might occurs in using centralized 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. Allows agents written in different languages to send and receive messages using the KQML standard.
2. Provides a highly flexible and scalable system which supports 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.
3. 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.
4. Furthermore, MPCS has reliable and fault tolerant features, and it can be easily be developed.
5. 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 one another 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 distributer* (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 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 agent needs to register in the system through this module. Agents interact directly through *Agent Interface*. 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 utilize the features of agent registration 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.** It contains the modules that are responsible for handling the messages among agents. The messages among agents are grouped into Dialogues. If an agent wants to communicate with another agent, it needs first to create a Dialogue with it, to which all subsequent messages between both are sent/received. An agent can maintain several Dialogues 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 dialogues among agents registered in the system to ensure the delivery of messages among them. Each agent has a *Dialogue Master Module* to manage all these Dialogues. Once this module registered in the system, it is automatically created. This module, with cooperation of others, responsible for the maintenance of the agent Dialogues, creating new Dialogues or sending/receiving messages within a Dialogue. The Dialogue Master can potentially receive a large number of Dialogue requests. To avoid bottlenecks this module distributes its work among Dialogue Sub-Masters Modules. A *Dialogue Sub-Master Module* exists for each pair of sending and receiving agents, responsible for creating dialogues and assigning messages to them. The *Policy Manager Module:* its goal is to check whether a message is allowed in a given Dialogue. It can permit any message sequence within a Dialogue or limit it to a specific course or several actions of courses. For example, it might be restrict the communication among specific agents to a simple question/answer dialogue, while other agents might be allowed free communication. This can be achieved by using the library of the interaction protocols [39], the protocols may range from simple query and request protocols, to more complex ones. For this reason, the Policy Manager is a completely independent module that can adapt to user requirements.

4.3. **Processing phase.** It contains the modules that are responsible for sending the message to a specific agent by checking its address if it is 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 module*, similar to the Dialogue master, is responsible for distributing the messages, avoiding a possible bottleneck by delegating the request to send a message to a Message Sub-Distributor. This *Message Sub-Distributor* is responsible for processing all the messages that belong to the same destination platform. The system will contain as many messages Sub-Distributors as platforms with which any communication is maintained. The *Message Translator Module* serializes 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

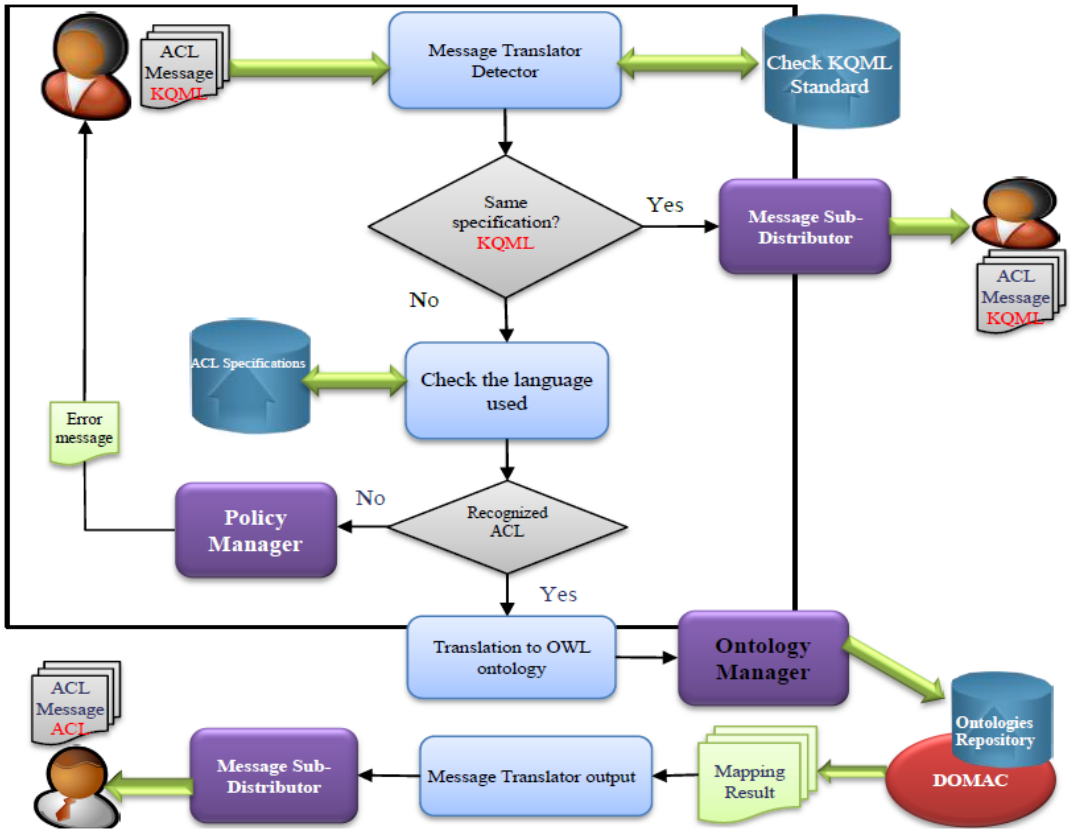

FIGURE 4. The data flow diagram for the message translator module

used by the sender agent, if the language used is different from the receiver agent, or vice versa, then a translation among 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 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 other's. External agents can either use the implemented default Transport Protocol, or another protocol. The default protocol modifies 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 among autonomous ontologies. This module uses the DOMAC system proposed in the next section to perform the mappings among 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 dialogues 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 ontology manager module includes a basic domain ontologies represented in OWL.

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.

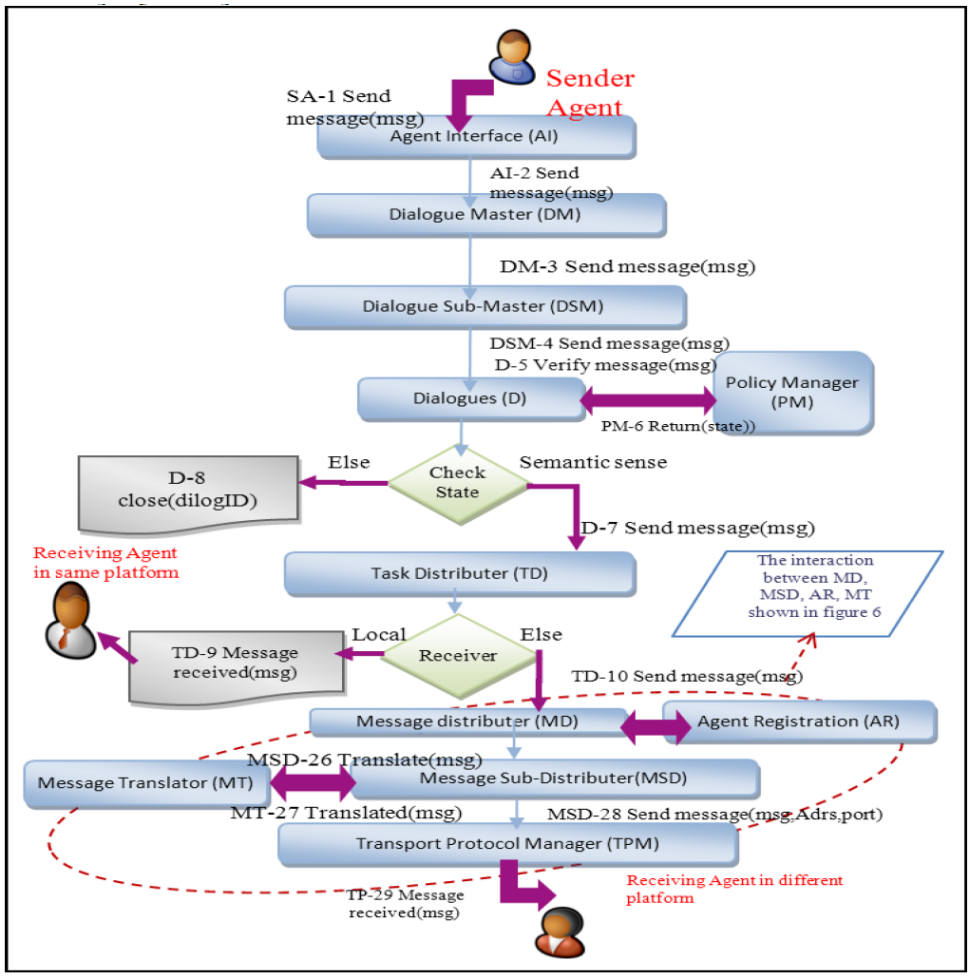

FIGURE 5. Sending local or global message in MPCS

The sending agent must register first and also that the corresponding dialogue must be previously created. First, the sending agent shows to Agent interface ***(AI)*** that it wants to send a message to another agent through a dialogue ***(D)***. Then, the *Agent interface* delegates the message transmission to the dialogue Master ***(DM)***, which in turns assigns the message to the dialogue Sub-manager ***(DSM)***. Both agents may maintain several dialogues at the same time, so the sending agent must inform the dialogue Sub-manager ***(DSM)*** of the dialogue to which the message is to be delegated. Then, the dialogue verifies the message state with the help of the Policy Manager ***(PM)***. If the state of the message makes semantic sense then the dialogue communicates with the task distributer ***(TD)*** to locate the receiving agent. When the task distributer ***(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 message distributor ***(MD)***, otherwise if the receiving agent is in an external platform, then the task distributer ***(TD)*** delegates the outgoing message transmission to the Message Distributor ***(MD)***, which will search the Message Sub-Distributor ***(MSD)*** corresponding to the destination platform. If the Message Sub-Distributor ***(MSD)*** does not exist, it is created at this point, with the necessary parameters to establish the communication obtained

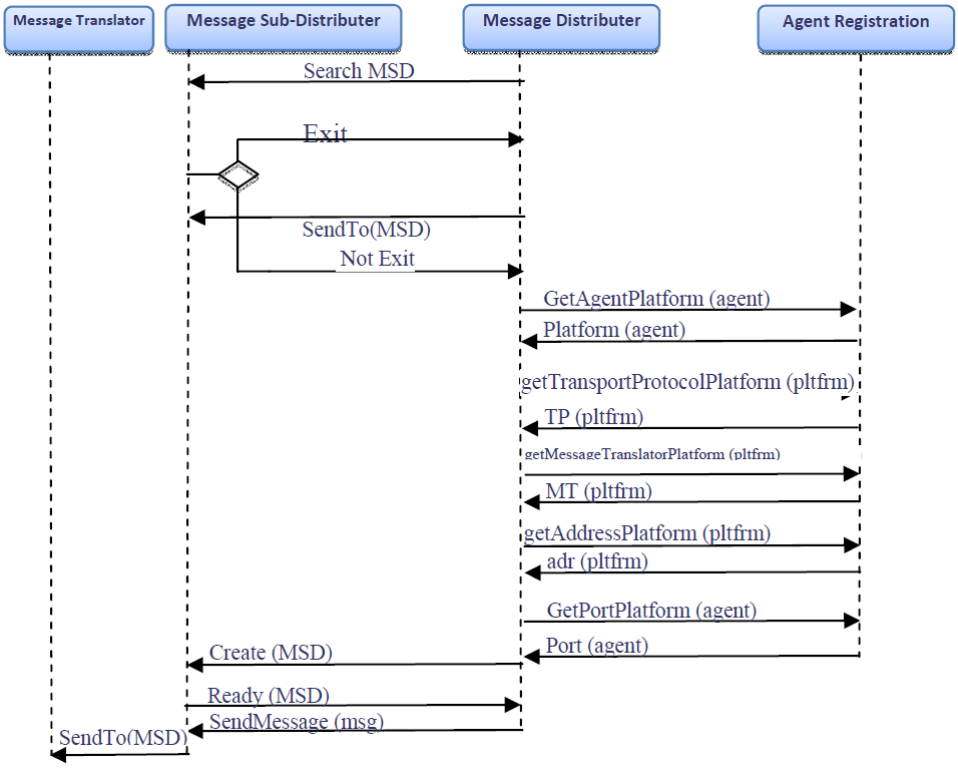

FIGURE 6. The interaction messages among MD, MSD, AR and MT modules

from the agent registration module ***(AR)***. Once the Message Sub-Distributor ***(MSD)*** has been located, it is assigned with the message to be sent, and in turn assigns the message to the appropriate Message translator ***(MT)***. The interaction among these four modules, namely, Message Distributor ***(MD)***, Message Sub-Distributor ***(MSD)***, Agent Registration ***(AR)***, and Message Translator ***(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 Message Sub-Distributor ***(MSD)***, then to the Transport Protocol Manager ***(TPM)*** used for communication with the given platform. Finally, the Transport Protocol Manager 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 Dynamic Ontology Mapping System for Agent Communication (DOMAC).** The Ontology mapping process takes two ontologies as input and creates a semantic correspondence among 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. There must be an ontology mapping algorithm used in the ontology manager proposed in the MPCS. As a result, a Dynamic Ontology Mapping System for Agent Communication (DOMAC) is proposed to show agents how to establish a mapping between two ontologies. The Dynamic Ontology Mapping System for Agent Communication (DOMAC) is shown in Figure 7; 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,

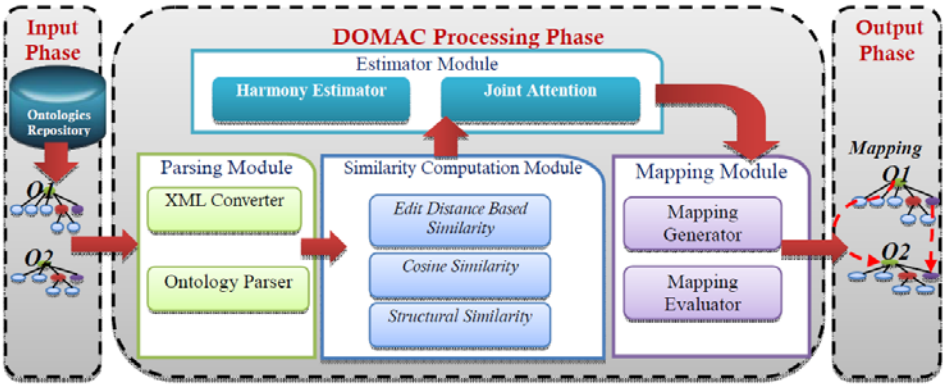

FIGURE 7. Dynamic ontology mapping system for agent communication (DOMAC)

among 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 [40] 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 message translator module in the MPCS, as showed in Figure 3. (3) The messages sent by external agents in form of OWL ontologies to the MPCS.

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 tokenizing. 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* [41] is calculated between the names of elements based on their Levenshtein distance. The similarity is defined as:

$$NameSim(e_{1i}, e_{2j}) = 1 - \frac{EditDist(e_{1i}, e_{2j})}{\max(l(e_{i1}), l(e_{2j}))} \tag{1}$$

  where the $EditDist(e_{1i}, e_{2j})$ is the Levenshtein distance between elements e_{1i} and e_{2j}, $l(e_{1i})$ and $l(e_{2j})$ are the string length of the name of e_{1i} and e_{2j} respectively.
- The *structural similarity* [41] between two elements calculated 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:

$$StructSim(e_{1i}, e_{2j}) = \frac{\sum_{k-1}^{n}(1 - diff_k(e_{1i}, e_{2j}))}{n} \tag{2}$$

where e_{1i} and e_{2j} are two class elements in ontology $O1$ and $O2$ respectively, n is the total number of structure features, and the $diff_k(e_{1i}, e_{2j})$ denotes the difference for feature k, and its defined as:

$$diff(e_{1i}, e_{2j}) = \frac{\mid sf(e_{1i}) - sf(e_{2j}) \mid}{\max(sf(e_{1i}), sf(e_{2j}))} \tag{3}$$

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

- *Cosine similarity* is a non-Euclidean distance measure between two vectors [42]; it is a common approach to compare documents in the field of text mining. Given two feature vectors c_i and c_j, the similarity score between concepts i and j is represented using the dot product as follows:

$$CosSim(i, j) = \frac{c_i \cdot c_j}{\mid c_i \mid \times \mid c_j \mid} \tag{4}$$

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, Agent 1 makes an announcement containing a unique representation of a concept and instance of the concept. After Agent 2 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 among the primitive concepts that make up the concept. The process of generating the 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 Agent 1 to Agent 2, the result of this process is called mapping:

A1. Node.Instructor.has.firstname $\leftrightarrow$ *A2. Node.lecturer.has.name.*

6. **Experimental Evaluation.** Several experiments were performed in two stages to validate the effectiveness of the proposed Multiplatform Communication System (MPCS) and the Dynamic Ontology Mapping System for Agent Communication (DOMAC).

6.1. **Stage1: Validation of the multiplatform communication system architecture (MPCS).** To investigate the effect of MPCS, a multi-agent system has been implemented to provide a testing platform. The whole system is implemented on a 5 Pc's with Intel Pentium 4 processor at 300GHz, with 2GB of Ram, connected with network Ethernet 512Mbps. A network of cooperative agents is designed, the number of agents ranges from 100 to 1000, 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. There are two experiments which 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 multi-agent system 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.

• **Experiment 1: Test scalability of local communication in MPCS**

In the local communication, when the sending and receiving agents are in the same platform, the MPCS are compared with two systems: JADE [15] and MOZART [5] system, as shown in Figure 8, and the computational time for the message delivery is measured when increasing the number of agents.

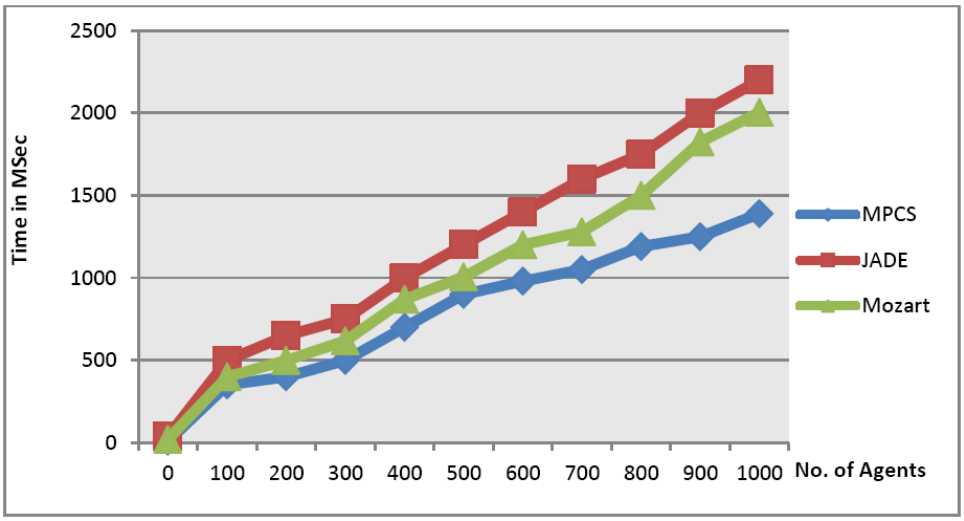

FIGURE 8. The local communication message delivery time

From the figure, 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.

• **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, as shown in Figure 9, and also the computational time for the message delivery is measured when increasing the number of agents.

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 gaining access to the directory service every time. Since the default directory service which employs LDAP has a slow response behavior, it is overwhelmed by a large number of concurrent requests [43]. This experiment shows that our MPCS architecture has the scalability advantage, as it behaves efficiently under full-load conditions comparing with recent systems.

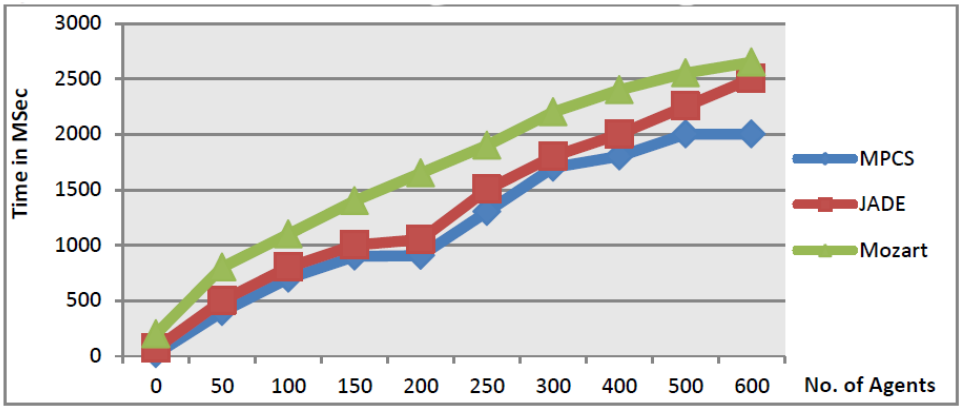

FIGURE 9. The global communication message delivery time

6.2. **Stage2: Validation of the dynamic ontology mapping system for agent communication (DOMAC).** A number of experiments were performed to validate the effectiveness of the proposed Dynamic Ontology Mapping System for Agent Communication (DOMAC). In each experiment, we used the Precision and recall to evaluate the experiment results which can be defined as follows:

$$\text{Precision: } P = \mid B \cap A \mid / \mid A \mid$$
$$\text{Recall: } R = \mid B \cap A \mid / \mid B \mid$$

In the formulas above, A presents the number of correct mappings recognized 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:

$$\text{F-Measure} = \frac{2.(\text{Percision} \times \text{Recall})}{\text{Percision} + \text{Recall}}$$

• **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 1000 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 [44] and KMS [45], as shown in Figure 10.

The experiment results showed 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 [46] to be the most effective and reliable than most of similarity methods.

• **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 (Ontology Alignment Evaluation Initiative), OAEI 2008 [47],

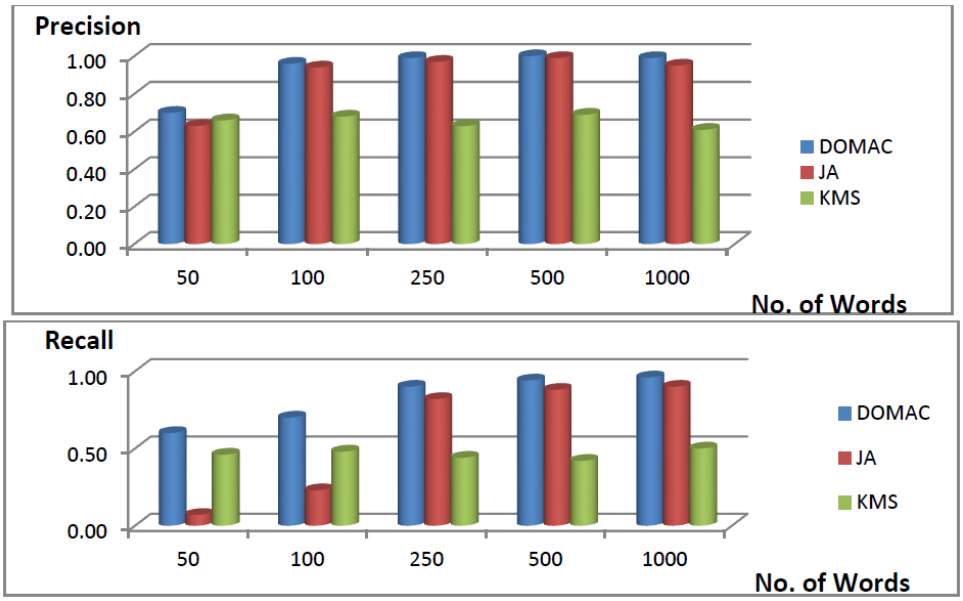

FIGURE 10. The precision and recall of DOMAC, JA and KMS

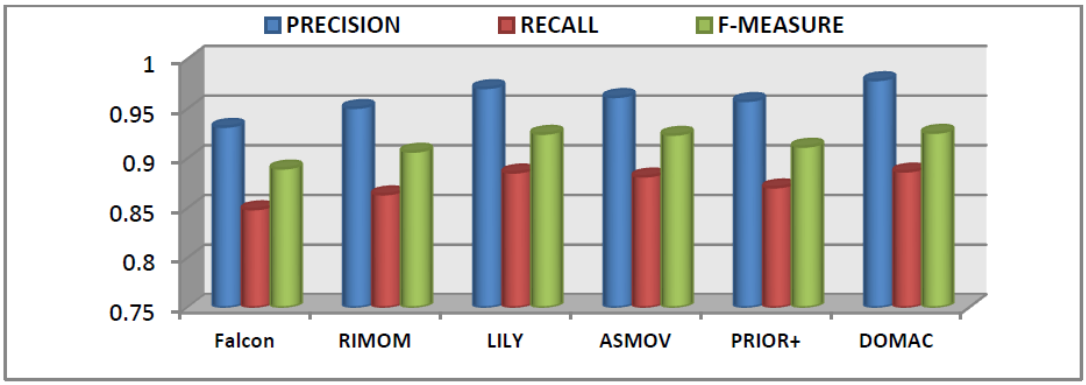

FIGURE 11. The comparison among DOMAC and top ranked systems

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 among different approaches are practical. (c) The ground truth of benchmark tests is open. Thus, we can use it to evaluate comprehensively 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 11 shows the comparison among the Precision, Recall and F-Measure of the DOMAC and top ranked systems on benchmark tests in OAEI campaign.

7. **Conclusion and Future Work.** In order to make possible interaction among agents in MAS, it is necessary to have a communication platform, a communication language and an ontology mapping system. In this sense, an outline of the communication among agents has been described by mean of the proposed communication layers, a Multiplatform communication system Architecture (MPCS) 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 Dynamic Ontology Mapping System for Agent Communication (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 uses of ontology researches 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 show 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.

The preliminary results obtained became a motivation to use our MPCS in the communication of Multi-Robot Systems. The communication between robots is essential for their cooperation in executing specific tasks. The agents used in MPCS can be simulated and put in group of robots that will be able to communicate with each other's even if their languages are different by using our DOMAC system that map different ontologies.

REFERENCES

[1] A. El-Desouky and S. El-Ghamrawy, A framework for distributed decision support multi-agent intelligent system, *Proc. of the 2008 International Conference on Artificial Intelligence*, Las Vegas, NV, USA, 2008.

[2] H. Farooq, Multi-agent systems: Overview of a new paradigm for distributed systems, *Proc. of the 7th IEEE International Symposium on High Assurance Systems Engineering*, Albuquerque, NM, USA, 2002.

[3] L. Wang and H. Zhao, Ontology for communication in distributed multi-agent system, *Proc.of the 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pp.588-592, Hong Kong, China, 2010.

[4] K. Sycara, K. S. Decker, A. Pannu, M. Williamson and D. Zeng, Distributed intelligent agents, *IEEE Expert*, vol.11, no.6, pp.36-46, 1996.

[5] J. A. Suarez-Romero, A. Alonso-Betanzos, B. Guijarro-Berdinas and C. Duran-Sanles, A tool for agent communication in Mozart/Oz, *Proc. of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp.706-710, France, 2005.

[6] H. Mellah, S. Hassas, I. Halilali, Z. Mesneb and H. Drias, Towards a self organizing protocol for a multi agents system (MASSOP), *Proc. of the 6th International Conference on Networking*, Martinique, pp.60, 2007.

[7] T. Fininand, R. Fritzson, D. McKay and R. McEntir, KQML as an agent communication Language, *Proc. of the 3rd International Conference on Information and Knowledge Management*, Gaithersburg, MD, USA, 1994.

[8] Y. Labrou, T. Finin and Y. Peng, Agent communication languages: The current landscape, *IEEE Intelligent Systems*, vol.14, no.2, pp.45-52, 1999.

[9] G. Weiss, *Multiagent systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, London, England, 1999.

[10] N. F. Noy and D. L. McGuinness, Ontology development 101: A guide to creating your first ontology, *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, 2001.

[11] G. Li, Y. Liu and B. Chen, Ontology and rule combined reasoning framework design, *ICIC Express Letters*, vol.4, no.5(B), pp.1753-1759, 2010.

[12] C. Trojahn, P. Quaresma and R. Vieira, Conjunctive queries for ontology based agent communication in MAS, *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, Estoril, Portugal, pp.829-836, 2008.

[13] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner, Ontology-based integration of information – A survey of existing approaches, *IJCAI 2001 Workshop on Ontologies and Information Sharing*, 2001.

[14] G. Tian and C. Cao, An ontology-driven multi-agent architecture for knowledge acquisition from text in NKI, *Proc. of CIMCA/IAWTIC*, pp.704-709, 2005.

[15] M. Laclavík, Z. Balogh, M. Babík and L. Hluchý, AgentOWL: Semantic knowledge model and agent architecture, *Computers and Artificial Intelligence*, vol.25, no.5, 2006.
[16] C. Şandru, V. Negru and D. Pop, A multi-agent problem solving architecture based on UPML, *Artificial Intelligence and Applications*, pp.597-602, 2005.
[17] D. Fensel et al., The unified problem-solving method development language, *Knowledge and Information Systems Journal*, vol.5, no.1, pp.83-131, 2003.
[18] M. Gómez et al., Domain-independent ontologies for cooperative information agents, *Proc. of the 5th Cooperative Information Agents Workshop*, pp.118-129, 2001.
[19] A. Hajnal, G. Pedone and L. Varga, Ontology-driven agent code generation for home care in protégé, *Proc. of the 10th International Protégé Conference*, pp.91-93, 2007.
[20] *K4CARE: Knowledge-Based Homecare EServices for An Ageing Europe*, http://www.k4care.net/.
[21] A. Doan, J. Madhaven, R. Dhamankar, P. Domingos and A. Halevy, Learning to match ontologies on the semantic web, *VLDB Journal*, vol.12, no.4, pp.303-319, 2002.
[22] M. Ehrig and S. Staab, QOM – Quick ontology mapping, *Proc. of International Semantic Web Conference*, pp.683-697, 2004.
[23] N. Noy and M. Musen, The PROMPT suite: Interactive tools for ontology merging and mapping, *International Journal of Human-Computer Studies*, vol.59, no.6, pp.983-1024, 2003.
[24] Y. Qu, W. Hu and G. Cheng, Constructing virtual documents for ontology matching, *Proc. of the 15th International Conference on World Wide Web*, 2006.
[25] J. Tang, J. Li, B. Liang, X. Huang, Y. Li and K. Wang, Using bayesian decision for ontology mapping, *Web Semantics: Science, Services and Agents on the World Wide Web*, vol.4, no.4, pp.243-262, 2006.
[26] P. Wang and B. Xu, LILY: The results for the ontology alignment contest OAEI 2007, *Proc. of ISWC 2007 Ontology Matching Workshop*, Busan, Korea, 2007.
[27] Y. R. Jean-Mary and M. R. Kabuka, ASMOV results for OAEI 2007, *Proc. of ISWC 2007 Ontology Matching Workshop*, Busan, Korea, 2007.
[28] M. Mao, Y. Peng and M. Spring, Integrating the IAC neural network in ontology mapping, *Proc. of the 17th International World Wide Web Conference*, 2008.
[29] Y. Labrou and T. Finin, *Semantics for an Agent Communication Language*, Ph.D. Thesis, University of Maryland, 1996.
[30] J. R. Searle, F. Kiefer and M. Bierwisch, *Speech Act Theory and Pragmatics*, Springer, 1980.
[31] M. Genesereth and R. Fikes, Knowledge interchange format, version3.0, reference manual, *Technical Report*, Computer Science Department, Stanford University, 1992.
[32] O. Lassila and R. R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification (W3C recommendation)*, http://www.w3.org/TR/REC-rdf-syntax/, 1999.
[33] *Rdf Schema*, http://www.w3.org/TR/1998/WD-rdf-schema-19980409/.
[34] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness and P. F. Patel-Schneider, OIL: An ontology infrastructure for the semantic web, *Intelligent Systems, IEEE*, vol.16, no.2, pp.38-45, 2001.
[35] Joint US/EU ad hoc Agent Markup Language Committee, *DAML+OIL (March 2001)*, http://www.daml.org/2001/03/daml+oil-index. Retrieved 2006-03-10, 2001.
[36] D. L. McGuinness and F. van Harmelen, *OWL Web Ontology Language Overview, W3C Recommendation*, www.w3.org/TR/2003/PR-owl-features-20031215/#ref-rdf-schema, 2004.
[37] *The Official Web Site of The World Wide Web Consortium*, www.W3C.org.
[38] H. S. Thompson, D. Beech, M. Maloney and N. Mendelsohn, *XML Schema (W3C Recommendation)*, http://www.w3.org/TR/xmlschema-1/, 2001.
[39] *FIPA Interaction Protocol Library*, http://www.fipa.org/repository/ips.html, 2001.
[40] W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu and M. Musen, Knowledge modelling at the millennium (The design and evolution of Protégé2000), *Proc. of the 12th Knowledge Acquisition, Modelling, and Management*, Ban, Canada, 1999.
[41] M. Mao, Y. Peng and M. Spring, A profile propagation and information retrieval based ontology mapping approach, *Proc. of SKG 2007*, Xi'an, China, 2007.
[42] J. Pan et al., Utilizing statistical semantic similarity techniques for ontology mapping – With applications to AEC standard models, *Tsinghua Science and Technology*, vol.13, no.S1, pp.217-222, 2008.
[43] X. Wang et al., Measurement and analysis of LDAP performance, *IEEE/ACM Transactions on Networking*, vol.16, no.1, 2008.
[44] W. Floris and R. Nico, Domain independent learning of ontology mappings, *Proc. of AAMAS'04*, New York, USA, 2004.

[45] R. Fu and Z. Xin, Research on electronic commerce KMS based on agent and ontology, *Proc. of the 1st International Workshop on Knowledge Discovery and Data Mining*, pp.190-195, 2008.
[46] M. Mao, Y. Peng and M. Spring, A harmony based adaptive ontology mapping approach, *Proc. of the 2008 International Conference on Semantic Web and Web Services*, 2008.
[47] *http://oaei.ontologymatching.org/2008/*.